Π.Σ.Α.Ε.Κ. ΠΑΤΡΑΣ

Σημειώσεις Στο μάθημα:

Γλώσσα Προγραμματισμού IV (ASP.NET, MVC, C#10)

Ειδικότητα: Τεχνικός λογισμικού Η/Υ

Τάξη: Δ’ εξάμηνο

Υπεύθυνη Καθηγήτρια:

Τζήμου Δέσποινα

ΠΑΤΡΑ ΜΑΡΤΙΟΣ 2026

Περιεχόμενα

Οι ενότητες ακολουθούν σπειροειδή πορεία: από τη γλώσσα C#, στον αντικειμενοστραφή προγραμματισμό, στο ASP.NET Core MVC, στις φόρμες, στο EF Core και τελικά σε ολοκληρωμένα mini projects.

1. Προσανατολισμός μαθήματος και τεχνολογική στοίβα

Η ενότητα «1. Προσανατολισμός μαθήματος και τεχνολογική στοίβα» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν τη σχέση C#, .NET, ASP.NET Core και MVC.
- Να αναγνωρίζουν το είδος εφαρμογών που θα αναπτύξουν.
- Να οργανώνουν σωστά το εργαστηριακό τους περιβάλλον.

Θεωρία και βασικές έννοιες

C# ως γλώσσα

Η έννοια «C# ως γλώσσα» είναι κομβική για την ενότητα «1. Προσανατολισμός μαθήματος και τεχνολογική στοίβα». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με τύπους, μεθόδους, κλάσεις και σύνταξη. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «C# ως γλώσσα» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η ίδια γλώσσα χρησιμοποιείται σε models, controllers, services και βοηθητικό κώδικα. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Ξεκινήστε με απλή Console εφαρμογή και κατόπιν δείξτε πώς η ίδια γλώσσα ζει σε MVC project.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «C# ως γλώσσα» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

.NET ως πλατφόρμα

Η έννοια «.NET ως πλατφόρμα» είναι κομβική για την ενότητα «1. Προσανατολισμός μαθήματος και τεχνολογική στοίβα». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με runtime, SDK, βιβλιοθήκες και εργαλεία. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «.NET ως πλατφόρμα» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Χωρίς το .NET δεν έχουμε περιβάλλον εκτέλεσης ούτε πρόσβαση στις απαραίτητες βιβλιοθήκες του framework. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δείξτε τις εκδόσεις SDK και εξηγήστε γιατί επιλέγουμε σταθερό περιβάλλον για όλο το εργαστήριο.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «.NET ως πλατφόρμα» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

ASP.NET Core

Η έννοια «ASP.NET Core» είναι κομβική για την ενότητα «1. Προσανατολισμός μαθήματος και τεχνολογική στοίβα». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με HTTP, browser, server και routing. Ο στόχος είναι να

περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «ASP.NET Core» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Το ASP.NET Core δίνει το web περιβάλλον στο οποίο η C# μετατρέπεται σε ολοκληρωμένη εφαρμογή με σελίδες και φόρμες. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Σχεδιάστε στον πίνακα τη διαδρομή browser -> server -> controller -> view.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «ASP.NET Core» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

MVC ως οργάνωση

Η έννοια «MVC ως οργάνωση» είναι κομβική για την ενότητα «1. Προσανατολισμός μαθήματος και τεχνολογική στοίβα». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με model, view, controller και διαχωρισμό ευθυνών. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «MVC ως οργάνωση» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Το MVC είναι ο τρόπος με τον οποίο διατηρούμε την εφαρμογή οργανωμένη όσο μεγαλώνει. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Ζητήστε από την τάξη να ταξινομήσει παραδείγματα σε Model, View ή Controller.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «MVC ως οργάνωση» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Πρώτη επαφή με C#

```
using System;

class Program
{
    static void Main()
    {
        Console.WriteLine("Καλωσορίσατε στο μάθημα ASP.NET Core MVC!");
        Console.WriteLine("Η C# είναι η βάση για όλο τον κώδικα που θα γράψουμε.");
    }
}
```

Το παρακάτω παράδειγμα για την ενότητα «1. Προσανατολισμός μαθήματος και τεχνολογική στοίβα» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Η Main είναι η αφετηρία σε απλή εφαρμογή κονσόλας.
- Η Console.WriteLine μας δίνει την πιο άμεση μορφή αλληλεπίδρασης.
- Το ίδιο πνεύμα μικρών βημάτων θα ακολουθήσουμε και όταν περάσουμε σε MVC έργα.

Σημεία προσοχής

- Να μη συγχέεται η C# με το ASP.NET Core.
- Να ξεκαθαριστεί ότι εδώ το GUI υλοποιείται κυρίως ως web διεπαφή.
- Να τονιστεί η αλυσίδα C# -> .NET -> ASP.NET Core -> MVC.

Μικρή εργαστηριακή δραστηριότητα

1. Εκτελέστε το πρώτο πρόγραμμα και αλλάξτε τα μηνύματα.
2. Συζητήστε ποια μέρη μιας web εφαρμογής πιστεύουν ότι θα γράφονται σε C#.

Ερωτήσεις κατανόησης

3. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «1. Προσανατολισμός μαθήματος και τεχνολογική στοίβα»;

4. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
5. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «1. Προσανατολισμός μαθήματος και τεχνολογική στοίβα» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

2. Εγκατάσταση εργαλείων και δημιουργία έργων

Η ενότητα «2. Εγκατάσταση εργαλείων και δημιουργία έργων» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να εγκαθιστούν Visual Studio 2022 Community και .NET 6 SDK.
- Να δημιουργούν Console και MVC έργα.
- Να χρησιμοποιούν τις βασικές εντολές του dotnet CLI.

Θεωρία και βασικές έννοιες

Visual Studio 2022

Η έννοια «Visual Studio 2022» είναι κομβική για την ενότητα «2. Εγκατάσταση εργαλείων και δημιουργία έργων». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με templates, debugger, solution explorer και NuGet. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Visual Studio 2022» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Το IDE επιτρέπει γρήγορη πλοήγηση στα αρχεία ενός MVC έργου και εύκολο debugging. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Περιηγηθείτε οργανωμένα στα τμήματα του IDE και αφήστε τους σπουδαστές να τα εντοπίσουν μόνοι τους.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Visual Studio 2022» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

dotnet CLI

Η έννοια «dotnet CLI» είναι κομβική για την ενότητα «2. Εγκατάσταση εργαλείων και δημιουργία έργων». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με dotnet new, dotnet run, dotnet build και add package. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «dotnet CLI» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Το CLI είναι ιδιαίτερα χρήσιμο για migrations και για εργασία εκτός του πλήρους IDE. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε έτοιμη μικρή λίστα εντολών και ζητήστε από τους σπουδαστές να τη δοκιμάσουν σε νέο φάκελο.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «dotnet CLI» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Solution και Project

Η έννοια «Solution και Project» είναι κομβική για την ενότητα «2. Εγκατάσταση εργαλείων και δημιουργία έργων». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με solution, project, dependencies και properties. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Solution και Project» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η σωστή κατανόηση της δομής λύσης και έργου διευκολύνει τη συντήρηση και την ομαδική συνεργασία. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Ξεκινήστε από μια κενή λύση και προσθέστε βήμα βήμα project ώστε να φαίνεται η σχέση τους.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Solution και Project» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Έκδοση C#10 και .NET 6

Η έννοια «Έκδοση C#10 και .NET 6» είναι κομβική για την ενότητα «2. Εγκατάσταση εργαλείων και δημιουργία έργων». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με συμβατότητα, templates, σταθερό περιβάλλον και εργαστήριο. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Έκδοση C#10 και .NET 6» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η σταθερή έκδοση μειώνει τις ασυμβατότητες και βοηθά την ομοιομορφία στο εργαστήριο. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δείξτε πώς ελέγχουμε εκδόσεις με dotnet --info και γιατί αυτό έχει πρακτική σημασία.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει

τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Έκδοση C#10 και .NET 6» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Βασικές εντολές dotnet

```
dotnet               new               console               -n               FirstConsoleApp
dotnet                                                                                 run
dotnet                 new                 mvc                 -n                 FirstMvcApp
dotnet                                                                                build
dotnet add package Microsoft.EntityFrameworkCore.SqlServer
```

Το παρακάτω παράδειγμα για την ενότητα «2. Εγκατάσταση εργαλείων και δημιουργία έργων» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Με το dotnet new δημιουργείται σκελετός έργου.
- Το dotnet run συνδυάζει build και εκτέλεση.
- Το dotnet add package μάς επιτρέπει να φέρνουμε βιβλιοθήκες όπως το EF Core.

Σημεία προσοχής

- Να αποφεύγονται πολλαπλές ασύμβατες εκδόσεις SDK χωρίς λόγο.
- Να υπάρχει σαφής οργάνωση φακέλων για κάθε project.
- Να διαχωρίζεται η έννοια της ανάπτυξης από τη δημοσίευση εφαρμογής.

Μικρή εργαστηριακή δραστηριότητα

6. Δημιουργήστε ένα Console App και ένα MVC App.
7. Εντοπίστε στον φάκελο του έργου ποια αρχεία παράγονται αυτόματα.

Ερωτήσεις κατανόησης

8. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «2. Εγκατάσταση εργαλείων και δημιουργία έργων»;
9. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
10. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «2. Εγκατάσταση εργαλείων και δημιουργία έργων» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα

διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

3. Βασική σύνταξη C# και πρώτη εφαρμογή

Η ενότητα «3. Βασική σύνταξη C# και πρώτη εφαρμογή» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να διαβάζουν και να γράφουν βασικές εντολές C#.
- Να κατανοούν τον ρόλο της Main.
- Να χρησιμοποιούν είσοδο και έξοδο στην κονσόλα.

Θεωρία και βασικές έννοιες

Δομή αρχείου C#

Η έννοια «Δομή αρχείου C#» είναι κομβική για την ενότητα «3. Βασική σύνταξη C# και πρώτη εφαρμογή». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με using, namespace, κλάση και Main. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Δομή αρχείου C#» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Οι ίδιες λογικές οργανωτικές μονάδες θα εμφανιστούν αργότερα σε controllers και άλλες κλάσεις. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αναλύστε ένα μικρό αρχείο γραμμή-γραμμή και ζητήστε να περιγράψουν τον ρόλο κάθε τμήματος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Δομή αρχείου C#» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε

μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Statements και blocks

Η έννοια «Statements και blocks» είναι κομβική για την ενότητα «3. Βασική σύνταξη C# και πρώτη εφαρμογή». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με εντολές, άγκιστρα, εσοχές και τερματικό ερωτηματικό. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Statements και blocks» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η σωστή μορφοποίηση από νωρίς μειώνει λάθη και βελτιώνει την αναγνωσιμότητα σε μεγαλύτερα έργα. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Χρησιμοποιήστε σκόπιμα λανθασμένο παράδειγμα για να βρουν και να διορθώσουν τα λάθη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Statements και blocks» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Console.ReadLine και WriteLine

Η έννοια «Console.ReadLine και WriteLine» είναι κομβική για την ενότητα «3. Βασική σύνταξη C# και πρώτη εφαρμογή». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με είσοδο, έξοδο, διάλογο και string. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Console.ReadLine και WriteLine» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η κονσόλα είναι παιδαγωγική γέφυρα πριν από τις φόρμες του MVC. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Μετατρέψτε ένα στατικό πρόγραμμα σε διαδραστικό με μία μόνο ερώτηση προς τον χρήστη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Console.ReadLine και WriteLine» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Σχόλια και ονοματοδοσία

Η έννοια «Σχόλια και ονοματοδοσία» είναι κομβική για την ενότητα «3. Βασική σύνταξη C# και πρώτη εφαρμογή». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με comments, ονόματα μεταβλητών, συντηρησιμότητα και καθαρότητα. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Σχόλια και ονοματοδοσία» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η ποιοτική γραφή κώδικα πρέπει να καλλιεργείται από την πρώτη εβδομάδα. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε καλές και κακές ονομασίες και συζητήστε πώς επηρεάζουν την κατανόηση.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει

τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Σχόλια και ονοματοδοσία» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Απλή αλληλεπίδραση με τον χρήστη

```
Console.Write("Δώσε το όνομά σου: ");
string? name = Console.ReadLine();
Console.WriteLine($"Γεια σου, {name}!");
```

Το παρακάτω παράδειγμα για την ενότητα «3. Βασική σύνταξη C# και πρώτη εφαρμογή» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Η ReadLine επιστρέφει string ή null.
- Η σύνταξη $"..." κάνει πιο καθαρή την ενσωμάτωση μεταβλητών.
- Το παράδειγμα είναι ιδανικό για μικρές παραλλαγές από τους ίδιους τους σπουδαστές.

Σημεία προσοχής

- Να μην ξεχνιούνται άγκιστρα και ερωτηματικά.
- Να μη χρησιμοποιούνται ασαφή ονόματα μεταβλητών.
- Να εξηγηθεί η διαφορά μεταξύ compile-time και run-time λαθών.

Μικρή εργαστηριακή δραστηριότητα

11. Προσθέστε ερώτηση για πόλη ή τμήμα σπουδών.
12. Αλλάξτε το πρόγραμμα ώστε να εμφανίζει δύο στοιχεία σε μία πρόταση.

Ερωτήσεις κατανόησης

13. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «3. Βασική σύνταξη C# και πρώτη εφαρμογή»;
14. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
15. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «3. Βασική σύνταξη C# και πρώτη εφαρμογή» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης

φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

4. Μεταβλητές, τύποι δεδομένων και μετατροπές

Η ενότητα «4. Μεταβλητές, τύποι δεδομένων και μετατροπές» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να δηλώνουν σωστά μεταβλητές και σταθερές.
- Να ξεχωρίζουν βασικούς τύπους δεδομένων.
- Να πραγματοποιούν ασφαλείς μετατροπές από string σε αριθμούς.

Θεωρία και βασικές έννοιες

Βασικοί τύποι

Η έννοια «Βασικοί τύποι» είναι κομβική για την ενότητα «4. Μεταβλητές, τύποι δεδομένων και μετατροπές». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με int, double, bool και char. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Βασικοί τύποι» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Οι ίδιοι τύποι θα εμφανίζονται στις ιδιότητες των models και στα πεδία της βάσης. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Χρησιμοποιήστε πίνακα τύπου, παραδείγματος τιμής και τυπικής χρήσης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Βασικοί τύποι» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε

μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Μεταβλητές και const

Η έννοια «Μεταβλητές και const» είναι κομβική για την ενότητα «4. Μεταβλητές, τύποι δεδομένων και μετατροπές». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με δήλωση, αρχικοποίηση, var και scope. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Μεταβλητές και const» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η σωστή δήλωση μεταβλητών επηρεάζει την αναγνωσιμότητα και την ασφάλεια των υπολογισμών. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δείξτε πότε το var κάνει τον κώδικα πιο καθαρό και πότε πιο ασαφή.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Μεταβλητές και const» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Parse και TryParse

Η έννοια «Parse και TryParse» είναι κομβική για την ενότητα «4. Μεταβλητές, τύποι δεδομένων και μετατροπές». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με μετατροπή, αποτυχία μετατροπής, validation και ασφάλεια. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Parse και TryParse» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η είσοδος από χρήστη και φόρμες έρχεται συνήθως ως string και πρέπει να ελέγχεται πριν χρησιμοποιηθεί. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε Parse με TryParse και ζητήστε να αιτιολογήσουν ποια επιλογή είναι ασφαλέστερη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Parse και TryParse» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Αριθμητική ακρίβεια

Η έννοια «Αριθμητική ακρίβεια» είναι κομβική για την ενότητα «4. Μεταβλητές, τύποι δεδομένων και μετατροπές». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με διαίρεση, στρογγυλοποίηση, εύρος τιμών και overflow. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Αριθμητική ακρίβεια» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Ο σωστός τύπος δεδομένων είναι σημαντικός όταν δουλεύουμε με μετρήσεις, βαθμούς ή οικονομικά στοιχεία. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε παραδείγματα ακέραιας και πραγματικής διαίρεσης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να

πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Αριθμητική ακρίβεια» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Ασφαλής μετατροπή ηλικίας

```
Console.Write("Δώσε                                    ηλικία:                                          ");
string?                      input                       =                        Console.ReadLine();
if                (int.TryParse(input,                 out                int                 age))
{
    Console.WriteLine($"Σε       5       χρόνια       θα       είσαι       {age       +       5}.");
}
else
{
    Console.WriteLine("Μη                              έγκυρη                                ηλικία.");
}
```

Το παρακάτω παράδειγμα για την ενότητα «4. Μεταβλητές, τύποι δεδομένων και μετατροπές» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Η TryParse αποφεύγει εξαίρεση σε κακή είσοδο.
- Η μεταβλητή age αποκτά τιμή μόνο όταν η μετατροπή πετύχει.
- Το ίδιο μοτίβο θα επανέλθει σε MVC φόρμες και validation.

Σημεία προσοχής

- Να μη χρησιμοποιείται Parse χωρίς έλεγχο σε δεδομένα χρήστη.
- Να μην παραβλέπεται η διαφορά int και double.
- Να συνδέεται πάντα ο τύπος με το νόημα του δεδομένου.

Μικρή εργαστηριακή δραστηριότητα

16. Υπολογίστε ΔΜΣ από βάρος και ύψος.
17. Δοκιμάστε επίτηδες μη έγκυρες τιμές και συζητήστε την αντίδραση του προγράμματος.

Ερωτήσεις κατανόησης

18. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «4. Μεταβλητές, τύποι δεδομένων και μετατροπές»;
19. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
20. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «4. Μεταβλητές, τύποι δεδομένων και μετατροπές» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

5. Τελεστές, εκφράσεις και αλφαριθμητικά

Η ενότητα «5. Τελεστές, εκφράσεις και αλφαριθμητικά» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να χρησιμοποιούν σωστά αριθμητικούς, σχεσιακούς και λογικούς τελεστές.
- Να χειρίζονται strings με βασικές μεθόδους.
- Να γράφουν σαφείς και αναγνώσιμες εκφράσεις.

Θεωρία και βασικές έννοιες

Αριθμητικοί τελεστές

Η έννοια «Αριθμητικοί τελεστές» είναι κομβική για την ενότητα «5. Τελεστές, εκφράσεις και αλφαριθμητικά». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με +, -, * και /. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Αριθμητικοί τελεστές» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Οι υπολογισμοί αποτελούν συχνό μέρος της επιχειρησιακής λογικής μέσα σε controllers ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Ζητήστε προσθήκη παρενθέσεων σε σύνθετες εκφράσεις ώστε να φανεί η σειρά αξιολόγησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Αριθμητικοί τελεστές» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε

μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Λογικοί και σχεσιακοί τελεστές

Η έννοια «Λογικοί και σχεσιακοί τελεστές» είναι κομβική για την ενότητα «5. Τελεστές, εκφράσεις και αλφαριθμητικά». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με ==, !=, && και ||. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Λογικοί και σχεσιακοί τελεστές» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η λογική ελέγχων είναι απαραίτητη για validation, επιλογές πλοήγησης και επιχειρησιακούς κανόνες. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Μετατρέψτε λεκτικούς κανόνες του προβλήματος σε λογικές συνθήκες.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Λογικοί και σχεσιακοί τελεστές» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

String interpolation

Η έννοια «String interpolation» είναι κομβική για την ενότητα «5. Τελεστές, εκφράσεις και αλφαριθμητικά». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με μορφοποίηση, συνένωση, αναγνωσιμότητα και αποφυγή θορύβου. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «String interpolation» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Τα views και τα μηνύματα σφάλματος ωφελούνται ιδιαίτερα από καθαρή διαχείριση συμβολοσειρών. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε τον ίδιο κώδικα με concatenation και με interpolation.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «String interpolation» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Βασικές string μέθοδοι

Η έννοια «Βασικές string μέθοδοι» είναι κομβική για την ενότητα «5. Τελεστές, εκφράσεις και αλφαριθμητικά». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με Trim, Length, Contains και ToUpper. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Βασικές string μέθοδοι» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Πολύ συχνά χρειάζεται να καθαρίζουμε είσοδο χρήστη πριν την επεξεργαστούμε. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Χρησιμοποιήστε παραδείγματα με κωδικούς μαθημάτων ή emails.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την

εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Βασικές string μέθοδοι» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Επεξεργασία string

```
string code = (Console.ReadLine() ?? "").Trim().ToUpper();
bool isValid = code.Length >= 4 && !code.Contains(" ");
Console.WriteLine($"Κωδικός: {code}");
Console.WriteLine($"Έγκυρος; {isValid}");
```

Το παρακάτω παράδειγμα για την ενότητα «5. Τελεστές, εκφράσεις και αλφαριθμητικά» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το Trim αφαιρεί κενά πριν και μετά το κείμενο.
- Η λογική έκφραση συνδυάζει δύο ελέγχους.
- Η προεπεξεργασία string είναι βασικό κομμάτι του validation.

Σημεία προσοχής

- Να αποφεύγονται περίπλοκες εκφράσεις χωρίς παρενθέσεις.
- Να δίνεται προσοχή στην πεζοκεφαλαιότητα.
- Να θυμόμαστε ότι string και αριθμός δεν είναι το ίδιο.

Μικρή εργαστηριακή δραστηριότητα

21. Ελέγξτε αν ένα email περιέχει @ και τελεία.
22. Μετασχηματίστε ονοματεπώνυμο σε μορφή με κεφαλαία αρχικά.

Ερωτήσεις κατανόησης

23. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «5. Τελεστές, εκφράσεις και αλφαριθμητικά»;
24. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
25. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «5. Τελεστές, εκφράσεις και αλφαριθμητικά» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης

φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

Διδακτική υπενθύμιση

- Πριν από κάθε εκτέλεση, ζητήστε πρόβλεψη αποτελέσματος από την τάξη.
- Μετά το βασικό παράδειγμα, δώστε μία μικρή παραλλαγή για ατομική ή ομαδική εργασία.
- Κλείστε με σύντομη ανακεφαλαίωση του πώς η έννοια επανέρχεται σε επόμενα κεφάλαια του MVC.

6. Δομές επιλογής: if, else, switch

Η ενότητα «6. Δομές επιλογής: if, else, switch» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να εκφράζουν σωστά εναλλακτικές ροές.
- Να επιλέγουν μεταξύ if και switch.
- Να μετατρέπουν λεκτικούς κανόνες σε συνθήκες.

Θεωρία και βασικές έννοιες

if και else

Η έννοια «if και else» είναι κομβική για την ενότητα «6. Δομές επιλογής: if, else, switch». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με συνθήκη, διακλάδωση, εκτέλεση μπλοκ και λογική. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «if και else» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Το if είναι βασικό σε validation, σε απόφαση πλοήγησης και σε επεξεργασία δεδομένων εισόδου. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Ξεκινήστε με μονογραμμική if και κλιμακώστε σταδιακά σε πιο σύνθετα σενάρια.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «if και else» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της

εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

else-if αλυσίδες

Η έννοια «else-if αλυσίδες» είναι κομβική για την ενότητα «6. Δομές επιλογής: if, else, switch». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με σειρά ελέγχων, κατηγοριοποίηση, προτεραιότητα και fallback. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «else-if αλυσίδες» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όταν πολλές περιπτώσεις είναι αμοιβαία αποκλειόμενες, η σειρά των ελέγχων έχει καθοριστική σημασία. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Ζητήστε να εξηγήσουν γιατί μία διαφορετική σειρά αλλάζει το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «else-if αλυσίδες» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

switch

Η έννοια «switch» είναι κομβική για την ενότητα «6. Δομές επιλογής: if, else, switch». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με διακριτές τιμές, καθαρότητα, περιπτώσεις και readability. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν

αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «switch» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Το switch προσφέρει σαφέστερη οργάνωση όταν έχουμε συγκεκριμένες εναλλακτικές τιμές. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε το ίδιο πρόβλημα λυμένο και με if-else και με switch.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «switch» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Επιχειρησιακοί κανόνες

Η έννοια «Επιχειρησιακοί κανόνες» είναι κομβική για την ενότητα «6. Δομές επιλογής: if, else, switch». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με εύρος τιμών, μη έγκυρες περιπτώσεις, μηνύματα και κανόνες προβλήματος. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Επιχειρησιακοί κανόνες» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Ο σκοπός είναι να εκφράζουμε κανόνες του πραγματικού προβλήματος και όχι απλώς να γράφουμε δομές επιλογής μηχανικά. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Ξεκινήστε από λεκτικό κανόνα και ζητήστε συλλογικά μετατροπή του σε λογική συνθήκη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Επιχειρησιακοί κανόνες» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Κατηγοριοποίηση βαθμού

```
if          (double.TryParse(Console.ReadLine(),          out          double          grade))
{
    if          (grade          <          0          ||          grade          >          10)
        Console.WriteLine("Μη                          έγκυρος                          βαθμός");
    else                    if                  (grade                  <                  5)
        Console.WriteLine("Αποτυχία");
    else                    if                  (grade                  <                  8)
        Console.WriteLine("Καλώς");
    else
        Console.WriteLine("Άριστα");
}
```

Το παρακάτω παράδειγμα για την ενότητα «6. Δομές επιλογής: if, else, switch» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Ελέγχουμε πρώτα ότι η είσοδος είναι αριθμός.
- Ακολουθεί έλεγχος εγκυρότητας πριν από την κατηγοριοποίηση.
- Η σειρά των else-if είναι κρίσιμη.

Σημεία προσοχής

- Να αποφεύγεται υπερβολική εμφώλευση.
- Να ελέγχονται ακραίες τιμές.
- Να αιτιολογείται πάντα η επιλογή μεταξύ if και switch.

Μικρή εργαστηριακή δραστηριότητα

26. Μετατρέψτε το παράδειγμα σε switch expression.
27. Σχεδιάστε λογικό κανόνα για φόρμα εισαγωγής βαθμού σε MVC.

Ερωτήσεις κατανόησης

28. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «6. Δομές επιλογής: if, else, switch»;
29. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
30. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «6. Δομές επιλογής: if, else, switch» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

7. Δομές επανάληψης: for, while, foreach

Η ενότητα «7. Δομές επανάληψης: for, while, foreach» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να χρησιμοποιούν σωστά επαναλήψεις.
- Να υπολογίζουν αθροίσματα, μέσους όρους και αναζητήσεις.
- Να αποφεύγουν άπειρους βρόχους και λάθη δεικτών.

Θεωρία και βασικές έννοιες

for

Η έννοια «for» είναι κομβική για την ενότητα «7. Δομές επανάληψης: for, while, foreach». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με μετρητή, αρχικοποίηση, συνθήκη και βήμα. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «for» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η for είναι κατάλληλη όταν γνωρίζουμε τον αριθμό επαναλήψεων ή όταν δουλεύουμε με δείκτες. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Ζητήστε πίνακα τιμών για την εξέλιξη του μετρητή.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «for» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της

εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

while

Η έννοια «while» είναι κομβική για την ενότητα «7. Δομές επανάληψης: for, while, foreach». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με άγνωστος αριθμός επαναλήψεων, sentinel, εισροή δεδομένων και τερματισμός. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «while» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η while ταιριάζει όταν ζητούμε επαναληπτικά έγκυρη είσοδο ή όταν η επανάληψη εξαρτάται από συνθήκη. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δείξτε παράδειγμα όπου ο χρήστης συνεχίζει μέχρι να γράψει ΤΕΛΟΣ.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «while» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

foreach

Η έννοια «foreach» είναι κομβική για την ενότητα «7. Δομές επανάληψης: for, while, foreach». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με συλλογές, ανάγνωση στοιχείων, καθαρότητα και ασφάλεια. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν

αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «foreach» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η foreach είναι θεμελιώδης για την εμφάνιση λιστών τόσο σε C# όσο και σε Razor views. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συνδέστε foreach σε List<T> και αντίστοιχο foreach σε .cshtml.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «foreach» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Συσσωρευτές

Η έννοια «Συσσωρευτές» είναι κομβική για την ενότητα «7. Δομές επανάληψης: for, while, foreach». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με sum, count, average και pattern. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Συσσωρευτές» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Πολλά μικρά προβλήματα εργαστηρίου λύνονται με το μοτίβο μετρητή και συσσωρευτή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Ζητήστε προφορική περιγραφή της μεταβολής των μεταβλητών σε κάθε βήμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Συσσωρευτές» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Μέσος όρος με for

```
double                          sum                          =                          0;
int                            count                         =                          5;
for         (int         i         =         1;         i         <=         count;         i++)
{
    if          (double.TryParse(Console.ReadLine(),          out          double          grade))
        sum                                      +=                                      grade;
}
Console.WriteLine($"Μέσος όρος: {sum / count:F2}");
```

Το παρακάτω παράδειγμα για την ενότητα «7. Δομές επανάληψης: for, while, foreach» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Η μεταβλητή sum λειτουργεί ως συσσωρευτής.
- Το count καθορίζει τον αριθμό επαναλήψεων.
- Το pattern αυτό επεκτείνεται εύκολα σε στατιστικές επεξεργασίες μικρών λιστών.

Σημεία προσοχής

- Να ελέγχεται πάντα η συνθήκη τερματισμού.
- Να προσέχονται τα όρια δεικτών.
- Να μη χρησιμοποιείται foreach όταν χρειάζεται χειροκίνητη αλλαγή index.

Μικρή εργαστηριακή δραστηριότητα

31. Υλοποιήστε ανάγνωση ονομάτων μέχρι ο χρήστης να γράψει ΤΕΛΟΣ.
32. Δημιουργήστε πίνακα τιμών για μία επανάληψη for.

Ερωτήσεις κατανόησης

33. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «7. Δομές επανάληψης: for, while, foreach»;
34. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
35. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «7. Δομές επανάληψης: for, while, foreach» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης

φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

8. Μέθοδοι, παράμετροι και επιστρεφόμενες τιμές

Η ενότητα «8. Μέθοδοι, παράμετροι και επιστρεφόμενες τιμές» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να διασπούν ένα πρόβλημα σε μικρές μεθόδους.
- Να ξεχωρίζουν παραμέτρους από επιστρεφόμενες τιμές.
- Να σχεδιάζουν επαναχρησιμοποιήσιμη λογική.

Θεωρία και βασικές έννοιες

Ρόλος των μεθόδων

Η έννοια «Ρόλος των μεθόδων» είναι κομβική για την ενότητα «8. Μέθοδοι, παράμετροι και επιστρεφόμενες τιμές». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με αποσύνθεση, επαναχρησιμοποίηση, τεστ και καθαρότητα. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Ρόλος των μεθόδων» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC, controllers και services αποτελούνται από μεθόδους με σαφείς ευθύνες. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Ξεκινήστε με μεγάλο μονολιθικό παράδειγμα και σπάστε το σε μικρότερες μεθόδους.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Ρόλος των μεθόδων» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράμετροι

Η έννοια «Παράμετροι» είναι κομβική για την ενότητα «8. Μέθοδοι, παράμετροι και επιστρεφόμενες τιμές». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με είσοδοι, τύποι, scope και ορίσματα. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Παράμετροι» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η κατανόηση παραμέτρων είναι βασική για action methods και constructors. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Σχεδιάστε τη ροή των τιμών από την κλήση προς τη μέθοδο και πίσω.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Παράμετροι» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Επιστρεφόμενες τιμές

Η έννοια «Επιστρεφόμενες τιμές» είναι κομβική για την ενότητα «8. Μέθοδοι, παράμετροι και επιστρεφόμενες τιμές». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με return, void, σύνθεση και λογική. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Επιστρεφόμενες τιμές» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Οι επιστρεφόμενες τιμές επιτρέπουν να χτίζουμε λογική βήμα-βήμα με καθαρό τρόπο. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δείξτε μέθοδο που επιστρέφει αποτέλεσμα και χρησιμοποιείται μέσα σε άλλη έκφραση.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Επιστρεφόμενες τιμές» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Σχεδίαση API μεθόδων

Η έννοια «Σχεδίαση API μεθόδων» είναι κομβική για την ενότητα «8. Μέθοδοι, παράμετροι και επιστρεφόμενες τιμές». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με ονόματα, μία ευθύνη, μικρό μέγεθος και σαφήνεια. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Σχεδίαση API μεθόδων» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η ποιότητα των μεθόδων επηρεάζει άμεσα τη συντηρησιμότητα ενός MVC project. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συζητήστε γιατί ένα όνομα όπως DoWork δεν βοηθά έναν τρίτο αναγνώστη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την

εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Σχεδίαση API μεθόδων» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Μέθοδος υπολογισμού τελικού βαθμού

```
static        double        CalculateFinalGrade(double        theory,        double        lab)
{
    return          theory          *          0.6          +          lab          *          0.4;
}
Console.WriteLine(CalculateFinalGrade(7.5, 8.0));
```

Το παρακάτω παράδειγμα για την ενότητα «8. Μέθοδοι, παράμετροι και επιστρεφόμενες τιμές» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Η μέθοδος έχει σαφείς εισόδους και μία έξοδο.
- Το όνομά της περιγράφει ακριβώς τον σκοπό της.
- Η ίδια λογική μπορεί να χρησιμοποιηθεί σε πολλά σημεία χωρίς αντιγραφή κώδικα.

Σημεία προσοχής

- Να αποφεύγονται μέθοδοι που τα κάνουν όλα.
- Να προσέχεται ο τύπος επιστροφής.
- Να συνδέεται η μέθοδος με ένα σαφές και μικρό καθήκον.

Μικρή εργαστηριακή δραστηριότητα

36. Διασπάστε πρόγραμμα εισαγωγής στοιχείων φοιτητή σε τρεις μεθόδους.
37. Προσθέστε έλεγχο εγκυρότητας στους βαθμούς.

Ερωτήσεις κατανόησης

38. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «8. Μέθοδοι, παράμετροι και επιστρεφόμενες τιμές»;
39. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
40. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «8. Μέθοδοι, παράμετροι και επιστρεφόμενες τιμές» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό

σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

9. Πίνακες, List<T> και Dictionary<TKey, TValue>

Η ενότητα «9. Πίνακες, List<T> και Dictionary<TKey, TValue>» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να αποθηκεύουν συλλογές δεδομένων.
- Να ξεχωρίζουν array, list και dictionary.
- Να επαναλαμβάνουν, να αναζητούν και να ταξινομούν δεδομένα.

Θεωρία και βασικές έννοιες

Πίνακες

Η έννοια «Πίνακες» είναι κομβική για την ενότητα «9. Πίνακες, List<T> και Dictionary<TKey, TValue>». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με σταθερό μέγεθος, indexes, Length και σειρά στοιχείων. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Πίνακες» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Οι πίνακες βοηθούν στην κατανόηση της έννοιας της συλλογής και του δείκτη. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δείξτε οπτικά ότι το πρώτο στοιχείο έχει index 0.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Πίνακες» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της

εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

List<T>

Η έννοια «List<T>» είναι κομβική για την ενότητα «9. Πίνακες, List<T> και Dictionary<TKey, TValue>». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με δυναμικό μέγεθος, Add, Remove και Count. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «List<T>» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Οι λίστες είναι η πιο συχνή μορφή συλλογής για δεδομένα που εμφανίζονται στα views. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συνδέστε List<Student> με μία σελίδα Index σε MVC εφαρμογή.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «List<T>» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Dictionary

Η έννοια «Dictionary» είναι κομβική για την ενότητα «9. Πίνακες, List<T> και Dictionary<TKey, TValue>». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με κλειδί, τιμή, αναζήτηση και ContainsKey. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Dictionary» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Τα λεξικά είναι χρήσιμα σε αντιστοιχίσεις κωδικού-περιγραφής ή ρυθμίσεων. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε παράδειγμα κωδικού μαθήματος προς τίτλο μαθήματος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Dictionary» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Περιήγηση σε συλλογές

Η έννοια «Περιήγηση σε συλλογές» είναι κομβική για την ενότητα «9. Πίνακες, List<T> και Dictionary<TKey, TValue>». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με for, foreach, αναζήτηση και συγκέντρωση στοιχείων. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Περιήγηση σε συλλογές» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η διαχείριση συλλογών είναι γέφυρα προς EF Core και λίστες δεδομένων της βάσης. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δείξτε πώς μία List περνά από controller σε view.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Περιήγηση σε συλλογές» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της

εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Λίστα φοιτητών

```
List<string> students = new List<string>();
students.Add("Άννα");
students.Add("Γιώργος");
students.Add("Μαρία");
foreach (string student in students)
{
    Console.WriteLine(student);
}
```

Το παρακάτω παράδειγμα για την ενότητα «9. Πίνακες, List<T> και Dictionary<TKey, TValue>» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Η λίστα ξεκινά κενή και εμπλουτίζεται με Add.
- Η foreach διατρέχει όλα τα στοιχεία με καθαρό τρόπο.
- Το ίδιο μοτίβο θα χρησιμοποιηθεί σε Razor για προβολή λιστών.

Σημεία προσοχής

- Να μη συγχέεται το Count με το τελευταίο index.
- Να αποφεύγεται πρόσβαση σε ανύπαρκτο στοιχείο.
- Να εξηγείται πότε συμφέρει κάθε συλλογή.

Μικρή εργαστηριακή δραστηριότητα

41. Δημιουργήστε λίστα μαθημάτων και εμφανίστε τα αριθμημένα.
42. Χρησιμοποιήστε Dictionary για αντιστοίχιση βαθμού σε χαρακτηρισμό.

Ερωτήσεις κατανόησης

43. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «9. Πίνακες, List<T> και Dictionary<TKey, TValue>»;
44. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
45. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «9. Πίνακες, List<T> και Dictionary<TKey, TValue>» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

Σύγκριση βασικών συλλογών

Δομή	**Χρήση**	**Πλεονεκτήματα**	**Σημεία προσοχής**
Array	Σταθερό πλήθος στοιχείων	Απλότητα, άμεση πρόσβαση με index	Δεν αλλάζει μέγεθος
List<T>	Δυναμική συλλογή αντικειμένων	Ευελιξία, πολλές μέθοδοι	Προσοχή σε index και null τιμές
Dictionary<TKey,TValue>	Αντιστοίχιση κλειδιού-τιμής	Γρήγορη αναζήτηση	Μοναδικότητα κλειδιών

10. Εξαιρέσεις και debugging στο Visual Studio

Η ενότητα «10. Εξαιρέσεις και debugging στο Visual Studio» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν τι είναι η εξαίρεση.
- Να χρησιμοποιούν try-catch όπου χρειάζεται.
- Να αξιοποιούν breakpoints και watch windows.

Θεωρία και βασικές έννοιες

Run-time errors

Η έννοια «Run-time errors» είναι κομβική για την ενότητα «10. Εξαιρέσεις και debugging στο Visual Studio». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με FormatException, DivideByZero, null και λάθη εκτέλεσης. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Run-time errors» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Οι εξαιρέσεις είναι αναπόφευκτες σε πραγματικές εφαρμογές και πρέπει να αντιμετωπίζονται μεθοδικά. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δείξτε πρώτα ένα πρόγραμμα που αποτυγχάνει και ύστερα τη διορθωμένη εκδοχή.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Run-time errors» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

try-catch-finally

Η έννοια «try-catch-finally» είναι κομβική για την ενότητα «10. Εξαιρέσεις και debugging στο Visual Studio». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με προστασία κώδικα, τύποι εξαιρέσεων, ανάκτηση και καθαρισμός. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «try-catch-finally» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο web, ο σωστός χειρισμός λαθών προστατεύει την εμπειρία του χρήστη και τη σταθερότητα της εφαρμογής. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Τονίστε ότι το exception handling δεν αντικαθιστά το καλό validation.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «try-catch-finally» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Debugger

Η έννοια «Debugger» είναι κομβική για την ενότητα «10. Εξαιρέσεις και debugging στο Visual Studio». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με breakpoints, step over, watch και locals. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Debugger» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Το debugging είναι δεξιότητα κατανόησης και όχι τυφλής δοκιμής τυχαίων αλλαγών. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Σταματήστε την εκτέλεση πριν από κρίσιμη γραμμή και αναλύστε την κατάσταση των μεταβλητών.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Debugger» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Μηνύματα σφαλμάτων

Η έννοια «Μηνύματα σφαλμάτων» είναι κομβική για την ενότητα «10. Εξαιρέσεις και debugging στο Visual Studio». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με compiler messages, stack trace, αιτία και σύμπτωμα. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Μηνύματα σφαλμάτων» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Οι φοιτητές πρέπει να μάθουν να διαβάζουν τα μηνύματα με ψυχραιμία και σειρά. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Ζητήστε να εντοπίσουν στο μήνυμα τη γραμμή και το είδος του προβλήματος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την

εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Μηνύματα σφαλμάτων» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Απλός χειρισμός εξαίρεσης

```
try
{
    int                  x                  =                  int.Parse(Console.ReadLine()!);
    Console.WriteLine(100                                  /                                  x);
}
catch                                                                                  (FormatException)
{
    Console.WriteLine("Η           είσοδος           δεν           είναι           ακέραιος.");
}
catch                                                                                       (DivideByZeroException)
{
    Console.WriteLine("Δεν       επιτρέπεται       διαίρεση       με       το       μηδέν.");
}
```

Το παρακάτω παράδειγμα για την ενότητα «10. Εξαιρέσεις και debugging στο Visual Studio» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Δείχνουμε δύο διαφορετικά είδη εξαιρέσεων.
- Η δομή catch επιτρέπει διαφορετική αντίδραση ανάλογα με το σφάλμα.
- Το παράδειγμα είναι κατάλληλο για εξάσκηση με debugger.

Σημεία προσοχής

- Να μη χρησιμοποιείται κενό catch.
- Να μη χρησιμοποιούνται exceptions εκεί που αρκεί η TryParse.
- Να υπάρχει στρατηγική debugging και όχι τυχαίες αλλαγές.

Μικρή εργαστηριακή δραστηριότητα

46. Τοποθετήστε breakpoint πριν από την Parse.
47. Κάντε το πρόγραμμα να συνεχίζει μέχρι να δοθεί έγκυρος αριθμός.

Ερωτήσεις κατανόησης

48. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «10. Εξαιρέσεις και debugging στο Visual Studio»;

49. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
50. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «10. Εξαιρέσεις και debugging στο Visual Studio» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

Διδακτική υπενθύμιση

- Πριν από κάθε εκτέλεση, ζητήστε πρόβλεψη αποτελέσματος από την τάξη.
- Μετά το βασικό παράδειγμα, δώστε μία μικρή παραλλαγή για ατομική ή ομαδική εργασία.
- Κλείστε με σύντομη ανακεφαλαίωση του πώς η έννοια επανέρχεται σε επόμενα κεφάλαια του MVC.

11. Κλάσεις και αντικείμενα

Η ενότητα «11. Κλάσεις και αντικείμενα» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «κλάση ως πρότυπο».
- Να εφαρμόζουν πρακτικά τις έννοιες στιγμιότυπο και πεδία και μέθοδοι.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

κλάση ως πρότυπο

Η έννοια «κλάση ως πρότυπο» είναι κομβική για την ενότητα «11. Κλάσεις και αντικείμενα». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με κλάση ως πρότυπο, στιγμιότυπο, πεδία και μέθοδοι και μοντελοποίηση οντοτήτων. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «κλάση ως πρότυπο» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «κλάση ως πρότυπο» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

στιγμιότυπο

Η έννοια «στιγμιότυπο» είναι κομβική για την ενότητα «11. Κλάσεις και αντικείμενα». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με στιγμιότυπο, πεδία και μέθοδοι, μοντελοποίηση οντοτήτων και 11. Κλάσεις και αντικείμενα. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «στιγμιότυπο» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «στιγμιότυπο» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

πεδία και μέθοδοι

Η έννοια «πεδία και μέθοδοι» είναι κομβική για την ενότητα «11. Κλάσεις και αντικείμενα». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με πεδία και μέθοδοι, μοντελοποίηση οντοτήτων, Student και 11. Κλάσεις και αντικείμενα. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «πεδία και μέθοδοι» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «πεδία και μέθοδοι» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

μοντελοποίηση οντοτήτων

Η έννοια «μοντελοποίηση οντοτήτων» είναι κομβική για την ενότητα «11. Κλάσεις και αντικείμενα». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με μοντελοποίηση οντοτήτων, κλάση ως πρότυπο, στιγμιότυπο και Student. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «μοντελοποίηση οντοτήτων» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την

εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «μοντελοποίηση οντοτήτων» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Student

```
public                  string                  Name                    =                   "";
public                                    int                                         Semester;
public                                   void                                       PrintInfo()
{
    Console.WriteLine($"{Name}                          -                         {Semester}");
}
```

Το παρακάτω παράδειγμα για την ενότητα «11. Κλάσεις και αντικείμενα» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «11. Κλάσεις και αντικείμενα» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

51. Τροποποιήστε το παράδειγμα «Student» προσθέτοντας μία μικρή επιπλέον λειτουργία.
52. Συζητήστε πώς το θέμα «11. Κλάσεις και αντικείμενα» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

53. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «11. Κλάσεις και αντικείμενα»;
54. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
55. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «11. Κλάσεις και αντικείμενα» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

12. Properties, ενθυλάκωση και constructors

Η ενότητα «12. Properties, ενθυλάκωση και constructors» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «ενθυλάκωση».
- Να εφαρμόζουν πρακτικά τις έννοιες properties και constructors.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

ενθυλάκωση

Η έννοια «ενθυλάκωση» είναι κομβική για την ενότητα «12. Properties, ενθυλάκωση και constructors». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με ενθυλάκωση, properties, constructors και ασφαλής αρχικοποίηση. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «ενθυλάκωση» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «ενθυλάκωση» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

properties

Η έννοια «properties» είναι κομβική για την ενότητα «12. Properties, ενθυλάκωση και constructors». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με properties, constructors, ασφαλής αρχικοποίηση και 12. Properties, ενθυλάκωση και constructors. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «properties» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «properties» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

constructors

Η έννοια «constructors» είναι κομβική για την ενότητα «12. Properties, ενθυλάκωση και constructors». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με constructors, ασφαλής αρχικοποίηση, Course και 12. Properties, ενθυλάκωση και constructors. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «constructors» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «constructors» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

ασφαλής αρχικοποίηση

Η έννοια «ασφαλής αρχικοποίηση» είναι κομβική για την ενότητα «12. Properties, ενθυλάκωση και constructors». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με ασφαλής αρχικοποίηση, ενθυλάκωση, properties και Course. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «ασφαλής αρχικοποίηση» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την

εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «ασφαλής αρχικοποίηση» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Course

```
public           string           Title           {           get;           set;           }
public           int           Semester           {           get;           set;           }
public              Course(string              title,              int              semester)
{
    Title                                    =                                    title;
    Semester                                 =                                    semester;
}
```

Το παρακάτω παράδειγμα για την ενότητα «12. Properties, ενθυλάκωση και constructors» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «12. Properties, ενθυλάκωση και constructors» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

56. Τροποποιήστε το παράδειγμα «Course» προσθέτοντας μία μικρή επιπλέον λειτουργία.
57. Συζητήστε πώς το θέμα «12. Properties, ενθυλάκωση και constructors» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

58. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «12. Properties, ενθυλάκωση και constructors»;
59. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
60. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «12. Properties, ενθυλάκωση και constructors» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

13. Υπερφόρτωση, static μέλη και this

Η ενότητα «13. Υπερφόρτωση, static μέλη και this» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «overloading».
- Να εφαρμόζουν πρακτικά τις έννοιες constructors και static.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

overloading

Η έννοια «overloading» είναι κομβική για την ενότητα «13. Υπερφόρτωση, static μέλη και this». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με overloading, constructors, static και this. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «overloading» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «overloading» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της

εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

constructors

Η έννοια «constructors» είναι κομβική για την ενότητα «13. Υπερφόρτωση, static μέλη και this». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με constructors, static, this και 13. Υπερφόρτωση, static μέλη και this. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «constructors» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «constructors» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

static

Η έννοια «static» είναι κομβική για την ενότητα «13. Υπερφόρτωση, static μέλη και this». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με static, this, Student και 13. Υπερφόρτωση, static μέλη και this. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν

αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «static» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «static» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

this

Η έννοια «this» είναι κομβική για την ενότητα «13. Υπερφόρτωση, static μέλη και this». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με this, overloading, constructors και Student. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «this» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «this» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε

μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Student

```
public Student() { Name = "Άγνωστος"; Semester = 1; }
public Student(string name, int semester)
{
    this.Name = name;
    this.Semester = semester;
}
```

Το παρακάτω παράδειγμα για την ενότητα «13. Υπερφόρτωση, static μέλη και this» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «13. Υπερφόρτωση, static μέλη και this» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

61. Τροποποιήστε το παράδειγμα «Student» προσθέτοντας μία μικρή επιπλέον λειτουργία.
62. Συζητήστε πώς το θέμα «13. Υπερφόρτωση, static μέλη και this» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

63. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «13. Υπερφόρτωση, static μέλη και this»;
64. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
65. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «13. Υπερφόρτωση, static μέλη και this» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

14. Κληρονομικότητα, polymorphism, abstract και interfaces

Η ενότητα «14. Κληρονομικότητα, polymorphism, abstract και interfaces» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «κληρονομικότητα».
- Να εφαρμόζουν πρακτικά τις έννοιες virtual/override και abstract.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

κληρονομικότητα

Η έννοια «κληρονομικότητα» είναι κομβική για την ενότητα «14. Κληρονομικότητα, polymorphism, abstract και interfaces». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με κληρονομικότητα, virtual/override, abstract και interfaces. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «κληρονομικότητα» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «κληρονομικότητα» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

virtual/override

Η έννοια «virtual/override» είναι κομβική για την ενότητα «14. Κληρονομικότητα, polymorphism, abstract και interfaces». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με virtual/override, abstract, interfaces και 14. Κληρονομικότητα, polymorphism, abstract και interfaces. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «virtual/override» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «virtual/override» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

abstract

Η έννοια «abstract» είναι κομβική για την ενότητα «14. Κληρονομικότητα, polymorphism, abstract και interfaces». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με abstract, interfaces, Person και 14. Κληρονομικότητα,

polymorphism, abstract και interfaces. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «abstract» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «abstract» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

interfaces

Η έννοια «interfaces» είναι κομβική για την ενότητα «14. Κληρονομικότητα, polymorphism, abstract και interfaces». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με interfaces, κληρονομικότητα, virtual/override και Person. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «interfaces» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «interfaces» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Person

```
public abstract string GetRole();
```

Το παρακάτω παράδειγμα για την ενότητα «14. Κληρονομικότητα, polymorphism, abstract και interfaces» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «14. Κληρονομικότητα, polymorphism, abstract και interfaces» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

66. Τροποποιήστε το παράδειγμα «Person» προσθέτοντας μία μικρή επιπλέον λειτουργία.
67. Συζητήστε πώς το θέμα «14. Κληρονομικότητα, polymorphism, abstract και interfaces» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

68. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «14. Κληρονομικότητα, polymorphism, abstract και interfaces»;
69. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
70. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «14. Κληρονομικότητα, polymorphism, abstract και interfaces» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

15. ASP.NET Core και κύκλος HTTP αιτήματος

Η ενότητα «15. ASP.NET Core και κύκλος HTTP αιτήματος» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «client-server».
- Να εφαρμόζουν πρακτικά τις έννοιες HTTP και Program.cs.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

client-server

Η έννοια «client-server» είναι κομβική για την ενότητα «15. ASP.NET Core και κύκλος HTTP αιτήματος». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με client-server, HTTP, Program.cs και middleware. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «client-server» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «client-server» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε

μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

HTTP

Η έννοια «HTTP» είναι κομβική για την ενότητα «15. ASP.NET Core και κύκλος HTTP αιτήματος». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με HTTP, Program.cs, middleware και 15. ASP.NET Core και κύκλος HTTP αιτήματος. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «HTTP» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «HTTP» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Program.cs

Η έννοια «Program.cs» είναι κομβική για την ενότητα «15. ASP.NET Core και κύκλος HTTP αιτήματος». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με Program.cs, middleware, Program.cs και 15. ASP.NET Core και κύκλος HTTP αιτήματος. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν

αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Program.cs» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Program.cs» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

middleware

Η έννοια «middleware» είναι κομβική για την ενότητα «15. ASP.NET Core και κύκλος HTTP αιτήματος». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με middleware, client-server, HTTP και Program.cs. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «middleware» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «middleware» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Program.cs

```
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddControllersWithViews();
var app = builder.Build();
app.UseStaticFiles();
app.UseRouting();
app.MapControllerRoute(name: "default", pattern: "{controller=Home}/{action=Index}/{id?}");
app.Run();
```

Το παρακάτω παράδειγμα για την ενότητα «15. ASP.NET Core και κύκλος HTTP αιτήματος» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «15. ASP.NET Core και κύκλος HTTP αιτήματος» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

71. Τροποποιήστε το παράδειγμα «Program.cs» προσθέτοντας μία μικρή επιπλέον λειτουργία.
72. Συζητήστε πώς το θέμα «15. ASP.NET Core και κύκλος HTTP αιτήματος» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

73. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «15. ASP.NET Core και κύκλος HTTP αιτήματος»;
74. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
75. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «15. ASP.NET Core και κύκλος HTTP αιτήματος» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

Διδακτική υπενθύμιση

- Πριν από κάθε εκτέλεση, ζητήστε πρόβλεψη αποτελέσματος από την τάξη.
- Μετά το βασικό παράδειγμα, δώστε μία μικρή παραλλαγή για ατομική ή ομαδική εργασία.
- Κλείστε με σύντομη ανακεφαλαίωση του πώς η έννοια επανέρχεται σε επόμενα κεφάλαια του MVC.

16. MVC αρχιτεκτονική και δομή έργου

Η ενότητα «16. MVC αρχιτεκτονική και δομή έργου» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «model».
- Να εφαρμόζουν πρακτικά τις έννοιες view και controller.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

model

Η έννοια «model» είναι κομβική για την ενότητα «16. MVC αρχιτεκτονική και δομή έργου». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με model, view, controller και δομή φακέλων. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «model» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «model» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της

εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

view

Η έννοια «view» είναι κομβική για την ενότητα «16. MVC αρχιτεκτονική και δομή έργου». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με view, controller, δομή φακέλων και 16. MVC αρχιτεκτονική και δομή έργου. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «view» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «view» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

controller

Η έννοια «controller» είναι κομβική για την ενότητα «16. MVC αρχιτεκτονική και δομή έργου». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με controller, δομή φακέλων, HelloController και 16. MVC αρχιτεκτονική και δομή έργου. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν

αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «controller» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «controller» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

δομή φακέλων

Η έννοια «δομή φακέλων» είναι κομβική για την ενότητα «16. MVC αρχιτεκτονική και δομή έργου». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με δομή φακέλων, model, view και HelloController. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «δομή φακέλων» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «δομή φακέλων» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - HelloController

```
public class HelloController : Controller
{
    public IActionResult Index()
    {
        return Content("Γεια από MVC");
    }
}
```

Το παρακάτω παράδειγμα για την ενότητα «16. MVC αρχιτεκτονική και δομή έργου» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «16. MVC αρχιτεκτονική και δομή έργου» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

76. Τροποποιήστε το παράδειγμα «HelloController» προσθέτοντας μία μικρή επιπλέον λειτουργία.
77. Συζητήστε πώς το θέμα «16. MVC αρχιτεκτονική και δομή έργου» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

78. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «16. MVC αρχιτεκτονική και δομή έργου»;
79. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
80. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «16. MVC αρχιτεκτονική και δομή έργου» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης

φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

Σύγκριση στοιχείων MVC

Στοιχείο	Ευθύνη	Παράδειγμα
Model	Δεδομένα και κανόνες	Student, Course, ContactViewModel
View	Παρουσίαση περιεχομένου	Index.cshtml, Create.cshtml
Controller	Χειρισμός αιτημάτων	StudentsController, HomeController

17. Controllers, routing και action methods

Η ενότητα «17. Controllers, routing και action methods» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «actions».
- Να εφαρμόζουν πρακτικά τις έννοιες routes και parameters.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

actions

Η έννοια «actions» είναι κομβική για την ενότητα «17. Controllers, routing και action methods». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με actions, routes, parameters και IActionResult. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «actions» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «actions» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της

εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

routes

Η έννοια «routes» είναι κομβική για την ενότητα «17. Controllers, routing και action methods». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με routes, parameters, IActionResult και 17. Controllers, routing και action methods. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «routes» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «routes» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

parameters

Η έννοια «parameters» είναι κομβική για την ενότητα «17. Controllers, routing και action methods». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με parameters, IActionResult, StudentsController και 17. Controllers, routing και action methods. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν

αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «parameters» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «parameters» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

IActionResult

Η έννοια «IActionResult» είναι κομβική για την ενότητα «17. Controllers, routing και action methods». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με IActionResult, actions, routes και StudentsController. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «IActionResult» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «IActionResult» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - StudentsController

```
public IActionResult Details(int id)
{
    return Content($"Προβολή φοιτητή με id = {id}");
}
```

Το παρακάτω παράδειγμα για την ενότητα «17. Controllers, routing και action methods» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «17. Controllers, routing και action methods» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

81. Τροποποιήστε το παράδειγμα «StudentsController» προσθέτοντας μία μικρή επιπλέον λειτουργία.
82. Συζητήστε πώς το θέμα «17. Controllers, routing και action methods» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

83. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «17. Controllers, routing και action methods»;
84. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
85. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «17. Controllers, routing και action methods» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

18. Views, Razor και layouts

Η ενότητα «18. Views, Razor και layouts» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «Razor».
- Να εφαρμόζουν πρακτικά τις έννοιες strongly typed views και layouts.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

Razor

Η έννοια «Razor» είναι κομβική για την ενότητα «18. Views, Razor και layouts». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με Razor, strongly typed views, layouts και partials. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Razor» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Razor» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της

εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

strongly typed views

Η έννοια «strongly typed views» είναι κομβική για την ενότητα «18. Views, Razor και layouts». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με strongly typed views, layouts, partials και 18. Views, Razor και layouts. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «strongly typed views» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «strongly typed views» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

layouts

Η έννοια «layouts» είναι κομβική για την ενότητα «18. Views, Razor και layouts». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με layouts, partials, Student View και 18. Views, Razor και layouts. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν

αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «layouts» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «layouts» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

partials

Η έννοια «partials» είναι κομβική για την ενότητα «18. Views, Razor και layouts». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με partials, Razor, strongly typed views και Student View. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «partials» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «partials» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε

μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Student View

```
@model Student
<h2>Στοιχεία φοιτητή</h2>
<p>Όνομα: @Model.Name</p>
<p>Εξάμηνο: @Model.Semester</p>
```

Το παρακάτω παράδειγμα για την ενότητα «18. Views, Razor και layouts» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «18. Views, Razor και layouts» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

86. Τροποποιήστε το παράδειγμα «Student View» προσθέτοντας μία μικρή επιπλέον λειτουργία.
87. Συζητήστε πώς το θέμα «18. Views, Razor και layouts» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

88. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «18. Views, Razor και layouts»;
89. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
90. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «18. Views, Razor και layouts» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

19. Models, ViewModels και model binding

Η ενότητα «19. Models, ViewModels και model binding» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «domain model».
- Να εφαρμόζουν πρακτικά τις έννοιες view model και model binding.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

domain model

Η έννοια «domain model» είναι κομβική για την ενότητα «19. Models, ViewModels και model binding». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με domain model, view model, model binding και ασφαλής είσοδος. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «domain model» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «domain model» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

view model

Η έννοια «view model» είναι κομβική για την ενότητα «19. Models, ViewModels και model binding». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με view model, model binding, ασφαλής είσοδος και 19. Models, ViewModels και model binding. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «view model» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «view model» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

model binding

Η έννοια «model binding» είναι κομβική για την ενότητα «19. Models, ViewModels και model binding». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με model binding, ασφαλής είσοδος, StudentRegisterViewModel και 19. Models, ViewModels και model binding. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «model binding» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «model binding» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

ασφαλής είσοδος

Η έννοια «ασφαλής είσοδος» είναι κομβική για την ενότητα «19. Models, ViewModels και model binding». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με ασφαλής είσοδος, domain model, view model και StudentRegisterViewModel. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «ασφαλής είσοδος» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την

εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «ασφαλής είσοδος» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - StudentRegisterViewModel

```
public                              class                                StudentRegisterViewModel
{
    public       string       FullName       {       get;       set;       }       =       "";
    public       string       Email       {       get;       set;       }       =       "";
    public          int          Semester          {          get;          set;          }
}
```

Το παρακάτω παράδειγμα για την ενότητα «19. Models, ViewModels και model binding» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «19. Models, ViewModels και model binding» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

91. Τροποποιήστε το παράδειγμα «StudentRegisterViewModel» προσθέτοντας μία μικρή επιπλέον λειτουργία.
92. Συζητήστε πώς το θέμα «19. Models, ViewModels και model binding» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

93. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «19. Models, ViewModels και model binding»;
94. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
95. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «19. Models, ViewModels και model binding» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

20. Φόρμες, GET/POST και validation

Η ενότητα «20. Φόρμες, GET/POST και validation» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «GET».
- Να εφαρμόζουν πρακτικά τις έννοιες POST και DataAnnotations.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

GET

Η έννοια «GET» είναι κομβική για την ενότητα «20. Φόρμες, GET/POST και validation». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με GET, POST, DataAnnotations και ModelState. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «GET» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «GET» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της

εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

POST

Η έννοια «POST» είναι κομβική για την ενότητα «20. Φόρμες, GET/POST και validation». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με POST, DataAnnotations, ModelState και 20. Φόρμες, GET/POST και validation. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «POST» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «POST» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

DataAnnotations

Η έννοια «DataAnnotations» είναι κομβική για την ενότητα «20. Φόρμες, GET/POST και validation». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με DataAnnotations, ModelState, ContactViewModel και 20. Φόρμες, GET/POST και validation. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση

του «DataAnnotations» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «DataAnnotations» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

ModelState

Η έννοια «ModelState» είναι κομβική για την ενότητα «20. Φόρμες, GET/POST και validation». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με ModelState, GET, POST και ContactViewModel. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «ModelState» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «ModelState» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε

μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - ContactViewModel

```
[Required]
public string Name { get; set; } = "";
[Required, EmailAddress]
public string Email { get; set; } = "";
[StringLength(500)]
public string Message { get; set; } = "";
```

Το παρακάτω παράδειγμα για την ενότητα «20. Φόρμες, GET/POST και validation» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «20. Φόρμες, GET/POST και validation» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

96. Τροποποιήστε το παράδειγμα «ContactViewModel» προσθέτοντας μία μικρή επιπλέον λειτουργία.
97. Συζητήστε πώς το θέμα «20. Φόρμες, GET/POST και validation» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

98. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «20. Φόρμες, GET/POST και validation»;
99. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
100. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «20. Φόρμες, GET/POST και validation» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

Διδακτική υπενθύμιση

- Πριν από κάθε εκτέλεση, ζητήστε πρόβλεψη αποτελέσματος από την τάξη.
- Μετά το βασικό παράδειγμα, δώστε μία μικρή παραλλαγή για ατομική ή ομαδική εργασία.
- Κλείστε με σύντομη ανακεφαλαίωση του πώς η έννοια επανέρχεται σε επόμενα κεφάλαια του MVC.

GET και POST

Στοιχείο	GET	POST
Σκοπός	Ανάκτηση ή εμφάνιση	Υποβολή και μεταβολή δεδομένων
Δεδομένα	Συνήθως στο URL	Στο σώμα αιτήματος ή φόρμας
Χρήση	Άνοιγμα φόρμας	Υποβολή φόρμας
Προσοχή	Όχι ευαίσθητα δεδομένα	Πάντα validation και anti-forgery

21. Entity Framework Core και DbContext

Η ενότητα «21. Entity Framework Core και DbContext» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «ORM».
- Να εφαρμόζουν πρακτικά τις έννοιες DbContext και DbSet.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

ORM

Η έννοια «ORM» είναι κομβική για την ενότητα «21. Entity Framework Core και DbContext». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με ORM, DbContext, DbSet και connection string. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «ORM» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «ORM» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της

εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

DbContext

Η έννοια «DbContext» είναι κομβική για την ενότητα «21. Entity Framework Core και DbContext». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με DbContext, DbSet, connection string και 21. Entity Framework Core και DbContext. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «DbContext» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «DbContext» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

DbSet

Η έννοια «DbSet» είναι κομβική για την ενότητα «21. Entity Framework Core και DbContext». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με DbSet, connection string, AppDbContext και 21. Entity Framework Core και DbContext. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν

αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «DbSet» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «DbSet» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

connection string

Η έννοια «connection string» είναι κομβική για την ενότητα «21. Entity Framework Core και DbContext». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με connection string, ORM, DbContext και AppDbContext. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «connection string» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «connection string» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - AppDbContext

```
public class AppDbContext : DbContext
{
    public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }
    public DbSet<Student> Students { get; set; }
}
```

Το παρακάτω παράδειγμα για την ενότητα «21. Entity Framework Core και DbContext» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «21. Entity Framework Core και DbContext» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

101. Τροποποιήστε το παράδειγμα «AppDbContext» προσθέτοντας μία μικρή επιπλέον λειτουργία.
102. Συζητήστε πώς το θέμα «21. Entity Framework Core και DbContext» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

103. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «21. Entity Framework Core και DbContext»;
104. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
105. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «21. Entity Framework Core και DbContext» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης

φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

22. Migrations και CRUD

Η ενότητα «22. Migrations και CRUD» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «migrations».
- Να εφαρμόζουν πρακτικά τις έννοιες create και edit/delete.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

migrations

Η έννοια «migrations» είναι κομβική για την ενότητα «22. Migrations και CRUD». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με migrations, create, edit/delete και redirect μετά από POST. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «migrations» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «migrations» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της

εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

create

Η έννοια «create» είναι κομβική για την ενότητα «22. Migrations και CRUD». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με create, edit/delete, redirect μετά από POST και 22. Migrations και CRUD. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «create» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «create» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

edit/delete

Η έννοια «edit/delete» είναι κομβική για την ενότητα «22. Migrations και CRUD». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με edit/delete, redirect μετά από POST, Create action και 22. Migrations και CRUD. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν

αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «edit/delete» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «edit/delete» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

redirect μετά από POST

Η έννοια «redirect μετά από POST» είναι κομβική για την ενότητα «22. Migrations και CRUD». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με redirect μετά από POST, migrations, create και Create action. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «redirect μετά από POST» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «redirect μετά από POST» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Create action

```
[HttpPost]
public                  IActionResult                 Create(Student                  student)
{
    if                (!ModelState.IsValid)               return                View(student);
    _context.Students.Add(student);
    _context.SaveChanges();
    return                                                    RedirectToAction(nameof(Index));
}
```

Το παρακάτω παράδειγμα για την ενότητα «22. Migrations και CRUD» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «22. Migrations και CRUD» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

106. Τροποποιήστε το παράδειγμα «Create action» προσθέτοντας μία μικρή επιπλέον λειτουργία.
107. Συζητήστε πώς το θέμα «22. Migrations και CRUD» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

108. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «22. Migrations και CRUD»;
109. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
110. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «22. Migrations και CRUD» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

23. LINQ, σχέσεις οντοτήτων και Include

Η ενότητα «23. LINQ, σχέσεις οντοτήτων και Include» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «LINQ».
- Να εφαρμόζουν πρακτικά τις έννοιες σχέσεις και Include.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

LINQ

Η έννοια «LINQ» είναι κομβική για την ενότητα «23. LINQ, σχέσεις οντοτήτων και Include». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με LINQ, σχέσεις, Include και σύνθετα δεδομένα. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «LINQ» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «LINQ» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της

εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

σχέσεις

Η έννοια «σχέσεις» είναι κομβική για την ενότητα «23. LINQ, σχέσεις οντοτήτων και Include». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με σχέσεις, Include, σύνθετα δεδομένα και 23. LINQ, σχέσεις οντοτήτων και Include. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «σχέσεις» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «σχέσεις» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Include

Η έννοια «Include» είναι κομβική για την ενότητα «23. LINQ, σχέσεις οντοτήτων και Include». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με Include, σύνθετα δεδομένα, LINQ query και 23. LINQ, σχέσεις οντοτήτων και Include. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν

αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «Include» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «Include» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

σύνθετα δεδομένα

Η έννοια «σύνθετα δεδομένα» είναι κομβική για την ενότητα «23. LINQ, σχέσεις οντοτήτων και Include». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με σύνθετα δεδομένα, LINQ, σχέσεις και LINQ query. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «σύνθετα δεδομένα» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «σύνθετα δεδομένα» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - LINQ query

```
var students = _context.Students
    .Where(s => s.Semester >= 4)
    .OrderBy(s => s.Name)
    .ToList();
```

Το παρακάτω παράδειγμα για την ενότητα «23. LINQ, σχέσεις οντοτήτων και Include» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «23. LINQ, σχέσεις οντοτήτων και Include» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

111. Τροποποιήστε το παράδειγμα «LINQ query» προσθέτοντας μία μικρή επιπλέον λειτουργία.
112. Συζητήστε πώς το θέμα «23. LINQ, σχέσεις οντοτήτων και Include» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

113. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «23. LINQ, σχέσεις οντοτήτων και Include»;
114. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
115. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «23. LINQ, σχέσεις οντοτήτων και Include» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

24. Upload αρχείων και στατικά αρχεία

Η ενότητα «24. Upload αρχείων και στατικά αρχεία» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «wwwroot».
- Να εφαρμόζουν πρακτικά τις έννοιες IFormFile και path.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

wwwroot

Η έννοια «wwwroot» είναι κομβική για την ενότητα «24. Upload αρχείων και στατικά αρχεία». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με wwwroot, IFormFile, path και έλεγχοι ασφαλείας. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «wwwroot» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «wwwroot» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε

μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

IFormFile

Η έννοια «IFormFile» είναι κομβική για την ενότητα «24. Upload αρχείων και στατικά αρχεία». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με IFormFile, path, έλεγχοι ασφαλείας και 24. Upload αρχείων και στατικά αρχεία. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «IFormFile» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «IFormFile» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

path

Η έννοια «path» είναι κομβική για την ενότητα «24. Upload αρχείων και στατικά αρχεία». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με path, έλεγχοι ασφαλείας, Upload action και 24. Upload αρχείων και στατικά αρχεία. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν

αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «path» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «path» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

έλεγχοι ασφαλείας

Η έννοια «έλεγχοι ασφαλείας» είναι κομβική για την ενότητα «24. Upload αρχείων και στατικά αρχεία». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με έλεγχοι ασφαλείας, wwwroot, IFormFile και Upload action. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «έλεγχοι ασφαλείας» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «έλεγχοι ασφαλείας» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Upload action

```
[HttpPost]
public async Task<IActionResult> Upload(IFormFile file)
{
    if (file == null || file.Length == 0) return BadRequest();
    string uploadsFolder = Path.Combine(_environment.WebRootPath, "uploads");
    Directory.CreateDirectory(uploadsFolder);
    string filePath = Path.Combine(uploadsFolder, file.FileName);
    using var stream = new FileStream(filePath, FileMode.Create);
    await file.CopyToAsync(stream);
    return Content("OK");
}
```

Το παρακάτω παράδειγμα για την ενότητα «24. Upload αρχείων και στατικά αρχεία» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «24. Upload αρχείων και στατικά αρχεία» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

116. Τροποποιήστε το παράδειγμα «Upload action» προσθέτοντας μία μικρή επιπλέον λειτουργία.
117. Συζητήστε πώς το θέμα «24. Upload αρχείων και στατικά αρχεία» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

118. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «24. Upload αρχείων και στατικά αρχεία»;
119. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
120. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «24. Upload αρχείων και στατικά αρχεία» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

25. Services, dependency injection, configuration και logging

Η ενότητα «25. Services, dependency injection, configuration και logging» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «services».
- Να εφαρμόζουν πρακτικά τις έννοιες interfaces και DI.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

services

Η έννοια «services» είναι κομβική για την ενότητα «25. Services, dependency injection, configuration και logging». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με services, interfaces, DI και logging. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «services» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «services» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε

μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

interfaces

Η έννοια «interfaces» είναι κομβική για την ενότητα «25. Services, dependency injection, configuration και logging». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με interfaces, DI, logging και 25. Services, dependency injection, configuration και logging. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «interfaces» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «interfaces» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

DI

Η έννοια «DI» είναι κομβική για την ενότητα «25. Services, dependency injection, configuration και logging». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με DI, logging, IStudentService και 25. Services, dependency injection, configuration και logging. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν

αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «DI» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «DI» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

logging

Η έννοια «logging» είναι κομβική για την ενότητα «25. Services, dependency injection, configuration και logging». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με logging, services, interfaces και IStudentService. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «logging» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «logging» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - IStudentService

```
public                                  interface                                   IStudentService
{
    IEnumerable<Student>                                                        GetActiveStudents();
}
```

Το παρακάτω παράδειγμα για την ενότητα «25. Services, dependency injection, configuration και logging» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «25. Services, dependency injection, configuration και logging» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

121. Τροποποιήστε το παράδειγμα «IStudentService» προσθέτοντας μία μικρή επιπλέον λειτουργία.
122. Συζητήστε πώς το θέμα «25. Services, dependency injection, configuration και logging» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

123. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «25. Services, dependency injection, configuration και logging»;
124. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
125. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «25. Services, dependency injection, configuration και logging» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό

σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

Διδακτική υπενθύμιση

- Πριν από κάθε εκτέλεση, ζητήστε πρόβλεψη αποτελέσματος από την τάξη.
- Μετά το βασικό παράδειγμα, δώστε μία μικρή παραλλαγή για ατομική ή ομαδική εργασία.
- Κλείστε με σύντομη ανακεφαλαίωση του πώς η έννοια επανέρχεται σε επόμενα κεφάλαια του MVC.

Πλάνο 13 εργαστηριακών συναντήσεων

Εβδομάδα	Θέμα	Παραδοτέο
1	Περιβάλλον και πρώτη εφαρμογή C#	Console πρόγραμμα εισόδου/εξόδου
2	Τύποι, επιλογές, επαναλήψεις	Μικρά προβλήματα validation
3	Μέθοδοι και συλλογές	Άσκηση με λίστες
4	OOP, κλάσεις, properties	Model classes
5	ASP.NET Core και MVC	Πρώτο MVC project
6	Controllers και routing	Νέος controller με actions
7	Views και Razor	Λίστα δεδομένων σε view
8	Forms και validation	Create/Edit φόρμα
9	EF Core και DbContext	Σύνδεση βάσης
10	Migrations και CRUD	CRUD λειτουργίες
11	LINQ και σχέσεις	Σύνθετη προβολή
12	Upload και polishing	Media ή Student project

Εβδομάδα	Θέμα	Παραδοτέο
13	Παρουσίαση έργων	Επίδειξη και αναστοχασμός

26. Χειρισμός σφαλμάτων και βασικές αρχές ασφάλειας

Η ενότητα «26. Χειρισμός σφαλμάτων και βασικές αρχές ασφάλειας» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «error pages».
- Να εφαρμόζουν πρακτικά τις έννοιες ασφάλεια και anti-forgery.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

error pages

Η έννοια «error pages» είναι κομβική για την ενότητα «26. Χειρισμός σφαλμάτων και βασικές αρχές ασφάλειας». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με error pages, ασφάλεια, anti-forgery και ποιοτικός κώδικας. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «error pages» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «error pages» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε

μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

ασφάλεια

Η έννοια «ασφάλεια» είναι κομβική για την ενότητα «26. Χειρισμός σφαλμάτων και βασικές αρχές ασφάλειας». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με ασφάλεια, anti-forgery, ποιοτικός κώδικας και 26. Χειρισμός σφαλμάτων και βασικές αρχές ασφάλειας. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «ασφάλεια» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «ασφάλεια» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

anti-forgery

Η έννοια «anti-forgery» είναι κομβική για την ενότητα «26. Χειρισμός σφαλμάτων και βασικές αρχές ασφάλειας». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με anti-forgery, ποιοτικός κώδικας, Error handling και 26. Χειρισμός σφαλμάτων και βασικές αρχές ασφάλειας. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν

αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «anti-forgery» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «anti-forgery» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

ποιοτικός κώδικας

Η έννοια «ποιοτικός κώδικας» είναι κομβική για την ενότητα «26. Χειρισμός σφαλμάτων και βασικές αρχές ασφάλειας». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με ποιοτικός κώδικας, error pages, ασφάλεια και Error handling. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «ποιοτικός κώδικας» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «ποιοτικός κώδικας» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Error handling

```
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}
```

Το παρακάτω παράδειγμα για την ενότητα «26. Χειρισμός σφαλμάτων και βασικές αρχές ασφάλειας» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «26. Χειρισμός σφαλμάτων και βασικές αρχές ασφάλειας» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

126. Τροποποιήστε το παράδειγμα «Error handling» προσθέτοντας μία μικρή επιπλέον λειτουργία.
127. Συζητήστε πώς το θέμα «26. Χειρισμός σφαλμάτων και βασικές αρχές ασφάλειας» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

128. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «26. Χειρισμός σφαλμάτων και βασικές αρχές ασφάλειας»;
129. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
130. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «26. Χειρισμός σφαλμάτων και βασικές αρχές ασφάλειας» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

27. Mini project A - Σύστημα διαχείρισης φοιτητών

Η ενότητα «27. Mini project A - Σύστημα διαχείρισης φοιτητών» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «οθόνες λίστας».
- Να εφαρμόζουν πρακτικά τις έννοιες form create/edit και search.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

οθόνες λίστας

Η έννοια «οθόνες λίστας» είναι κομβική για την ενότητα «27. Mini project A - Σύστημα διαχείρισης φοιτητών». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με οθόνες λίστας, form create/edit, search και project evaluation. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «οθόνες λίστας» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «οθόνες λίστας» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

form create/edit

Η έννοια «form create/edit» είναι κομβική για την ενότητα «27. Mini project A - Σύστημα διαχείρισης φοιτητών». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με form create/edit, search, project evaluation και 27. Mini project A - Σύστημα διαχείρισης φοιτητών. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «form create/edit» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «form create/edit» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

search

Η έννοια «search» είναι κομβική για την ενότητα «27. Mini project A - Σύστημα διαχείρισης φοιτητών». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με search, project evaluation, Student model και 27. Mini project A -

Σύστημα διαχείρισης φοιτητών. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «search» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «search» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

project evaluation

Η έννοια «project evaluation» είναι κομβική για την ενότητα «27. Mini project A - Σύστημα διαχείρισης φοιτητών». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με project evaluation, οθόνες λίστας, form create/edit και Student model. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «project evaluation» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «project evaluation» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - Student model

```
public                                    class                                        Student
{
    public            int            Id           {           get;            set;           }
    public       string       FullName       {       get;       set;       }       =       "";
    public       string       Email       {       get;       set;       }       =       "";
    public           int           Semester           {           get;           set;           }
}
```

Το παρακάτω παράδειγμα για την ενότητα «27. Mini project A - Σύστημα διαχείρισης φοιτητών» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «27. Mini project A - Σύστημα διαχείρισης φοιτητών» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

131. Τροποποιήστε το παράδειγμα «Student model» προσθέτοντας μία μικρή επιπλέον λειτουργία.
132. Συζητήστε πώς το θέμα «27. Mini project A - Σύστημα διαχείρισης φοιτητών» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

133. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «27. Mini project A - Σύστημα διαχείρισης φοιτητών»;

134. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
135. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «27. Mini project Α - Σύστημα διαχείρισης φοιτητών» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

28. Mini project B - Κατάλογος πολυμεσικού υλικού

Η ενότητα «28. Mini project B - Κατάλογος πολυμεσικού υλικού» έχει διπλό ρόλο: αφενός εισάγει ή επεκτείνει μια τεχνική έννοια της C# ή του ASP.NET Core, αφετέρου εντάσσει τη γνώση αυτή μέσα στη συνολική λογική ανάπτυξης μιας εφαρμογής MVC. Σε εργαστηριακό μάθημα δεν αρκεί ο θεωρητικός ορισμός. Απαιτείται συνεχής μετάβαση από την εξήγηση στην επίδειξη και από την επίδειξη στη μικρή αυτόνομη άσκηση.

Η συγκεκριμένη θεματική προσφέρεται για ζωντανό coding, για σύντομες προφορικές ερωτήσεις κατανόησης και για προσεκτική αναφορά στα συνηθισμένα λάθη που εμφανίζονται όταν ο/η εκπαιδευόμενος/η δοκιμάζει μόνος/η του/της νέες παραλλαγές. Η διδασκαλία γίνεται πιο αποτελεσματική όταν αναδεικνύεται όχι μόνο το πώς γράφεται μια δομή, αλλά και το γιατί επιλέγεται στη συγκεκριμένη περίπτωση.

Μαθησιακοί στόχοι

- Να κατανοούν οι εκπαιδευόμενοι το θέμα «media catalog».
- Να εφαρμόζουν πρακτικά τις έννοιες upload εικόνας και cards.
- Να συνδέουν τη γνώση αυτή με ολοκληρωμένη εφαρμογή MVC και με το ευρύτερο πλαίσιο του μαθήματος.

Θεωρία και βασικές έννοιες

media catalog

Η έννοια «media catalog» είναι κομβική για την ενότητα «28. Mini project B - Κατάλογος πολυμεσικού υλικού». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με media catalog, upload εικόνας, cards και extensions. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «media catalog» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Στο MVC η έννοια αυτή επηρεάζει άμεσα τον τρόπο που οργανώνουμε models, controllers, views ή services. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Δώστε αρχικά μικρό παράδειγμα και στη συνέχεια ζητήστε μία ελεγχόμενη παραλλαγή από την τάξη.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «media catalog» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

upload εικόνας

Η έννοια «upload εικόνας» είναι κομβική για την ενότητα «28. Mini project Β - Κατάλογος πολυμεσικού υλικού». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με upload εικόνας, cards, extensions και 28. Mini project Β - Κατάλογος πολυμεσικού υλικού. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «upload εικόνας» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Η έννοια αυτή μεταφράζεται πρακτικά σε αποφάσεις σχεδίασης, naming και καθαρή ροή δεδομένων μέσα στην εφαρμογή. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Συγκρίνετε μία καλή και μία προβληματική υλοποίηση για να φανεί το όφελος.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «upload εικόνας» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

cards

Η έννοια «cards» είναι κομβική για την ενότητα «28. Mini project Β - Κατάλογος πολυμεσικού υλικού». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με cards, extensions, MediaItem και 28. Mini project Β - Κατάλογος

πολυμεσικού υλικού. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «cards» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Εδώ αναδεικνύεται η γέφυρα ανάμεσα στη θεωρία της C# και στη λειτουργικότητα που τελικά εμφανίζεται στον browser. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Αφήστε τους σπουδαστές να τροποποιήσουν μία γραμμή και να παρατηρήσουν το αποτέλεσμα.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «cards» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

extensions

Η έννοια «extensions» είναι κομβική για την ενότητα «28. Mini project B - Κατάλογος πολυμεσικού υλικού». Δεν αρκεί να θυμόμαστε ποια σύνταξη γράφεται στο πληκτρολόγιο. Χρειάζεται να αντιλαμβανόμαστε ποιο πρόβλημα λύνει, πότε συμφέρει να τη χρησιμοποιήσουμε και πώς επηρεάζει τη δομή του κώδικα. Στο συγκεκριμένο μάθημα, η έννοια αυτή συναντάται όταν εργαζόμαστε με extensions, media catalog, upload εικόνας και MediaItem. Ο στόχος είναι να περάσουμε από την επιφανειακή γνώση της σύνταξης σε ουσιαστική κατανόηση της συμπεριφοράς του προγράμματος.

Διδακτικά είναι χρήσιμο να προηγείται ένα απλό παράδειγμα κονσόλας και στη συνέχεια να ακολουθεί ένα πιο ρεαλιστικό σενάριο μέσα σε web εφαρμογή. Με αυτόν τον τρόπο ο/η εκπαιδευόμενος/η βλέπει ότι η θεωρία δεν αποτελεί αποκομμένο κεφάλαιο αλλά εργαλείο επίλυσης πραγματικών προβλημάτων. Στην πράξη, η σωστή χρήση του «extensions» οδηγεί σε κώδικα πιο ευανάγνωστο, λιγότερο επιρρεπή σε λάθη και ευκολότερο στη συντήρηση.

Η σύνδεση με ASP.NET Core MVC είναι σημαντική. Όσο το project μεγαλώνει, αυτή η έννοια λειτουργεί ως μηχανισμός οργάνωσης, ποιότητας και επεκτασιμότητας. Όταν οι φοιτητές κατανοούν αυτή τη γέφυρα, αντιλαμβάνονται ότι οι βασικές έννοιες της C# δεν χάνονται όταν περνάμε στο web, αλλά αντιθέτως αποτελούν το θεμέλιο πάνω στο οποίο χτίζονται controllers, models, views και services.

Ως πρακτική διδασκαλίας, προτείνεται η εξής ροή: Κλείστε την ενότητα με μικρή άσκηση ή προφορική ερώτηση κατανόησης.. Η επανάληψη με μικρές παραλλαγές είναι ιδιαίτερα αποτελεσματική, γιατί αναγκάζει τον/την εκπαιδευόμενο/η να πειραματιστεί ενεργά και να εντοπίσει σχέσεις αιτίας-αποτελέσματος ανάμεσα στον κώδικα και στο τελικό αποτέλεσμα.

Συχνή παρανόηση των αρχάριων είναι ότι το «extensions» αποτελεί μεμονωμένη τεχνική χωρίς συνέχεια. Στην πραγματικότητα επανεμφανίζεται σε πολλές διαφορετικές καταστάσεις: σε επεξεργασία εισόδου, σε μοντελοποίηση οντοτήτων, σε επικύρωση δεδομένων, σε ροή πλοήγησης και στη διαχείριση της κατάστασης της εφαρμογής. Η επισήμανση αυτής της συνέχειας βοηθά τους εκπαιδευόμενους να χτίσουν ενιαίο νοητικό μοντέλο και όχι ασύνδετες γνώσεις.

Παράδειγμα κώδικα - MediaItem

```
public                                  class                                       MediaItem
{
    public           int           Id          {           get;           set;           }
    public       string       Title       {       get;       set;       }       =       "";
    public       string       Category       {       get;       set;       }       =       "";
    public       string       Description       {       get;       set;       }       =       "";
    public          string?          ImagePath          {          get;          set;          }
}
```

Το παρακάτω παράδειγμα για την ενότητα «28. Mini project Β - Κατάλογος πολυμεσικού υλικού» είναι σκόπιμα μικρό αλλά ουσιαστικό. Ο εκπαιδευτής μπορεί να το εκτελέσει αυτούσιο και στη συνέχεια να ζητήσει από την τάξη να προτείνει μία ή δύο στοχευμένες τροποποιήσεις.

Πριν από την εκτέλεση αξίζει να ζητηθεί πρόβλεψη αποτελέσματος. Η πρόβλεψη αυτή μετατρέπει το παράδειγμα από απλή παρακολούθηση σε ενεργό νοητική συμμετοχή και συχνά αποκαλύπτει παρεξηγήσεις που αλλιώς θα παρέμεναν κρυφές.

- Το παράδειγμα είναι μικρό ώστε να αναλύεται γρήγορα στο εργαστήριο.
- Η δομή του επιτρέπει μικρές αλλαγές χωρίς να χάνεται η βασική ιδέα.
- Με κατάλληλη συζήτηση μπορεί να συνδεθεί τόσο με κονσόλα όσο και με MVC περιβάλλον.

Σημεία προσοχής

- Να μη μείνει το θέμα «28. Mini project Β - Κατάλογος πολυμεσικού υλικού» μόνο σε ορισμούς χωρίς πρακτική εφαρμογή.
- Να αποφεύγονται υπερβολικά μεγάλα βήματα που κουράζουν ή μπερδεύουν τους αρχάριους.
- Να αναδεικνύεται κάθε φορά τι ανήκει στη γλώσσα C#, τι στο framework και τι στην αρχιτεκτονική MVC.

Μικρή εργαστηριακή δραστηριότητα

136. Τροποποιήστε το παράδειγμα «MediaItem» προσθέτοντας μία μικρή επιπλέον λειτουργία.
137. Συζητήστε πώς το θέμα «28. Mini project Β - Κατάλογος πολυμεσικού υλικού» θα φαινόταν μέσα σε ένα μεγαλύτερο project του εργαστηρίου.

Ερωτήσεις κατανόησης

138. Ποιο είναι το βασικό πρόβλημα που λύνει η ενότητα «28. Mini project Β - Κατάλογος πολυμεσικού υλικού»;
139. Πώς θα εξηγούσατε τη συγκεκριμένη έννοια σε συμφοιτητή που μπερδεύτηκε στο εργαστήριο;
140. Με ποιον τρόπο επανεμφανίζεται η έννοια αυτή μέσα σε μια εφαρμογή ASP.NET Core MVC;

Σύνδεση με μεγαλύτερο project

Το περιεχόμενο της ενότητας «28. Mini project Β - Κατάλογος πολυμεσικού υλικού» δεν πρέπει να μείνει αποκομμένο. Κάθε νέα τεχνική αποκτά πραγματική αξία όταν εντάσσεται σε μεγαλύτερο project, όπως ένα μικρό σύστημα διαχείρισης φοιτητών ή ένας κατάλογος πολυμεσικού υλικού. Ο εκπαιδευτής μπορεί να κλείνει κάθε μάθημα με σύντομη υπενθύμιση του πού ακριβώς θα ξανασυναντήσουμε τη γνώση αυτή στο επόμενο βήμα της εφαρμογής.

Με αυτόν τον τρόπο, οι εκπαιδευόμενοι κατανοούν ότι το μάθημα δεν είναι σειρά αποσπασματικών θεμάτων αλλά διαδοχική κατασκευή ενός ενιαίου τεχνολογικού οικοσυστήματος. Η επιστροφή σε προηγούμενες έννοιες μέσα σε νέο πλαίσιο αποτελεί βασικό μοχλό εμπέδωσης, ειδικά σε περιβάλλον εργαστηρίου όπου η πράξη προηγείται συχνά της πλήρους θεωρητικής ωρίμανσης.

Παράρτημα Β - Ενδεικτικές ασκήσεις εργαστηρίου

141. Να δημιουργηθεί Console εφαρμογή που ζητά τρία ονόματα μαθημάτων και εμφανίζει το μεγαλύτερο σε μήκος.
142. Να υλοποιηθεί κλάση Book με properties Title, Author, Year και μέθοδο PrintCard().
143. Να γραφτεί πρόγραμμα που διαβάζει βαθμούς σε List<double> και εμφανίζει μέσο όρο και μέγιστο.
144. Να κατασκευαστεί controller ProductsController με actions Index και Details(int id).
145. Να δημιουργηθεί strongly typed view που εμφανίζει στοιχεία Course.
146. Να σχεδιαστεί φόρμα επικοινωνίας με validation Name, Email και Message.
147. Να υλοποιηθεί CRUD για οντότητα Category σε βάση δεδομένων μέσω EF Core.
148. Να προστεθεί αναζήτηση σε λίστα Students με query parameter q.
149. Να επεκταθεί mini project με upload εικόνας και προβολή thumbnail.
150. Να δημιουργηθεί service που επιστρέφει μόνο ενεργά αντικείμενα από τη βάση.
151. Να σχεδιαστεί ViewModel που συνδυάζει λίστα κατηγοριών και στοιχεία προϊόντος για φόρμα edit.
152. Να προστεθεί basic logging σε δύο κρίσιμες actions.
153. Να δημιουργηθεί αναφορά συνηθισμένων compile-time και run-time λαθών.
154. Να σχεδιαστεί rubric αξιολόγησης mini project.
155. Να προστεθεί ταξινόμηση ή σελιδοποίηση σε λίστα δεδομένων.
156. Να προστεθεί φίλτρο κατηγορίας σε mini project πολυμέσων.
157. Να σχεδιαστεί φόρμα εγγραφής φοιτητή με dropdown εξαμήνου.
158. Να υλοποιηθεί απλό interface IPrintable σε δύο διαφορετικές κλάσεις.
159. Να δημιουργηθεί σχέση Student-Department και προβολή σχετικών δεδομένων.
160. Να βελτιωθεί μια φόρμα με μηνύματα σφάλματος και μήνυμα επιτυχίας.

Παράρτημα Δ - Σύντομες ερωτήσεις θεωρίας για επανάληψη

161. Τι διαφορά έχει η C# ως γλώσσα από το ASP.NET Core ως framework;
162. Πότε χρησιμοποιούμε TryParse αντί για Parse;
163. Γιατί η string interpolation θεωρείται πιο ευανάγνωστη;
164. Πότε προτιμάται η for και πότε η foreach;
165. Τι σημαίνει ότι μια μέθοδος έχει μία μόνο ευθύνη;
166. Ποια είναι η διαφορά ανάμεσα σε κλάση και αντικείμενο;
167. Γιατί προτιμούμε properties αντί για public πεδία;
168. Πότε έχει νόημα η κληρονομικότητα και πότε ένα interface;
169. Τι ρόλο παίζει το Program.cs σε ASP.NET Core MVC;
170. Πώς συνεργάζονται Model, View και Controller;
171. Τι είναι το routing και γιατί είναι σημαντικό;
172. Ποια είναι η διαφορά ανάμεσα σε domain model και ViewModel;
173. Γιατί χρειαζόμαστε validation και server-side έλεγχο;
174. Τι είναι το DbContext και τι τα DbSet;
175. Γιατί χρησιμοποιούμε migrations;

176. Τι μας προσφέρει η LINQ;
177. Ποιοι βασικοί κίνδυνοι υπάρχουν σε file upload;
178. Γιατί προτιμούμε services και dependency injection σε πιο οργανωμένο έργο;
179. Τι σημαίνει καθαρός controller;
180. Ποια βήματα θα ακολουθούσατε για να εντοπίσετε ένα bug σε MVC εφαρμογή;