Работен лист

Коментари и документация в програмирането (ASP.NET Core MVC) - Уроци 1-2

Клас: 12 Модул: 3 Език: C# Технология: ASP.NET Core MVC

Име: _____ Дата: _____

Цели за двата урока

• Да различаваш: коментар (в кода) vs документация (README/архитектура).
• Да прилагаш правилото: кодът показва как, а документацията казва защо.
• Да подобряваш четимост чрез именуване и рефакторинг преди да добавяш коментари.
• Да разпознаваш полезни и вредни коментари и да ги пренаписваш правилно.

Инструкция: Работи чисто и конкретно. Където се иска “рефакторинг”, целта е по-добри имена и по-ясна структура, а не добавяне на много коментари.

Урок 1: Четим код vs коментари

Основна идея: Преди да пишеш коментар, опитай да направиш кода по-ясен чрез имена, структури и разделяне на отговорности.

Упражнение 1: Рефакторинг на лоши имена (без добавяне на нови коментари)

```
public class BController : Controller
{
private readonly AppDbContext _db;

public BController(AppDbContext db)
{
_db = db;
}

public IActionResult I()
{
var x = _db.Books.ToList(); // get list
return View(x); // return view
}

public IActionResult D(int id)
{
var b = _db.Books.FirstOrDefault(z => z.Id == id); // find
if (b == null) return RedirectToAction("I"); // not found
return View(b);
}
}
```

Задача: Опиши как би рефакторирал кода, така че да стане самообясним (примерни промени):

1) Преименуване на Controller-а и action-ите (смислени имена).
1) Преименуване на променливите (x, b, z) и премахване на излишни коментари.
1) По-добър резултат при липсващ запис (напр. NotFound/Details).

Упражнение 2: “Коментар или по-добро име?”

За всяка ситуация избери най-доброто решение: (A) добавям коментар, (B) сменям име/рефакторирам, (C) и двете (с кратко обяснение).

1. Методът се казва Process(), но реално записва книга в базата.

2. Имаш число 30 в кода за “макс дни заемане”, но никъде не е обяснено.

3. В Service има сложна проверка по бизнес правило (не е очевидна от кода).

4. В Controller има 15 реда логика за заемане на книга.

Урок 2: Видове коментари и стил

Работим с: // (еднoредов), /* ... */ (многоредов, рядко), както и маркери TODO и NOTE.

Правило: Коментарът трябва да добавя информация, която кодът сам не казва (например “защо”, “ограничение”, “бизнес правило”, “edge case”).

Упражнение 3: Класифицирай коментарите

Маркирай всеки коментар като: П (полезен), В (вреден), Н (нуждае се от пренаписване). Ако е Н - пренапиши го по-добре.

1.

```
// return view
```

Категория (П/В/Н): _____ Ако е Н - пренаписан коментар:

2.

```
// gets books from database
```

Категория (П/В/Н): _____ Ако е Н - пренаписан коментар:

3.

```
// Business rule: a book can be borrowed only when IsBorrowed is false.
```

Категория (П/В/Н): _____ Ако е Н - пренаписан коментар:

4.

```
// TODO: fix later
```

Категория (П/В/Н): _____ Ако е Н - пренаписан коментар:

5.

```
// NOTE: We call SaveChangesAsync here to persist IsBorrowed before creating the
BorrowRecord.
```

Категория (П/В/Н): _____ Ако е Н - пренаписан коментар:

6.

```
// HACK: Using AsNoTracking because the view must not modify entities.
```

Категория (П/В/Н): _____ Ако е Н - пренаписан коментар:

7.

```
// This is important
```

Категория (П/В/Н): _____ Ако е Н - пренаписан коментар:

8.

```
/* old code below */
/* var temp = ... */
```

Категория (П/В/Н): _____ Ако е Н - пренаписан коментар:

9.

```
// Trim input to avoid duplicate student names with trailing spaces.
```

Категория (П/В/Н): _____ Ако е Н - пренаписан коментар:

10.

```
// 30 means max borrow days
```

Категория (П/В/Н): _____ Ако е Н - пренаписан коментар:

Упражнение 4: Напиши добри TODO и NOTE

Пренапиши следните “лоши” маркери така, че да са конкретни: какво липсва, защо и кога е важно.

```
// TODO: fix
// NOTE: check this
```

Пример за добър TODO: // TODO: Add validation for StudentName (required) before release - prevents empty borrow records.

Твоите версии:

Мини задачи: Рефакторинг + коментари (5-10 мин)

Мини задача A: Премахни излишните коментари, като подобриш имената и структурата.

```
public async Task<IActionResult> B(int id, string n)
{
// find book
var b = await _db.Books.FindAsync(id);
if (b == null) return RedirectToAction("I"); // not found

// if borrowed do nothing
if (b.IsBorrowed) return RedirectToAction("I");

// set borrowed
b.IsBorrowed = true;

// save
await _db.SaveChangesAsync();

return RedirectToAction("I"); // done
}
```

Опиши 3 конкретни промени (имена/извличане в service/резултати при грешка):

Мини задача B: Добави само един смислен коментар (NOTE или Business rule) на най-подходящото място.

```
public async Task<bool> ReturnAsync(int bookId)
{
var book = await _db.Books.FindAsync(bookId);
if (book is null) return false;

if (!book.IsBorrowed) return false;

book.IsBorrowed = false;
await _db.SaveChangesAsync();
return true;
}
```

Къде би добавил коментар и какъв точно? (1-2 изречения)

Бележка за учителя: Ако ще раздаваш листа на ученици, не е нужно да има “верни отговори” тук. Проверявай за: смислени имена, премахнати излишни коментари, конкретни TODO/NOTE, и коментари, които обясняват “защо”.