NobleProg
The World's Local Training Provider

Plan szkolenia – Tworzenie aplikacji internetowych w technologii ASP.NET Core MVC- kurs praktyczny

1. **Analiza wymagań**

Na podstawie przygotowanej makiety (wireframe) studenci analizują wymagania dotyczące aplikacji internetowej którą będą budować w trakcie kury. Określają listę potrzebnych stron, czy strony będą zawierały formularze, czy będzie potrzebny magazyn danych, czy będzie potrzebna identyfikacja użytkownika.

2. **Przygotowanie kodu html/css**

Studenci poznają podstawy frameworku css'owego **Bootstrap**. Następnie przygotowują statyczny kod stron w layout'cie typu **grid**. Na tą chwilę treść jest ustawiana „na sztywno".

3. **Wprowadzenie do ASP.NET Core MVC**

Omówienie wzorca projektowego MVC. Studenci tworzą w Visual Studio startową aplikację typu ASP.NET Core Web App (Model-View-Controller). Identyfikowane są w niej poszczególne elementy wzorca MVC oraz elementy specyficzne dla ASP.NET Core MVC np.

- **Areas, Shared Views**

- **routing** w aplikacjach ASP.NET Core MVC
Omówiona zostaje struktura katalogów w solucji.

4. **Przygotowanie widoków**

Studenci przenoszą statyczny kod html do widoków **cshtml**.

5. **Przygotowanie klas modelu**

Na podstawie makiety studenci budują model danych potrzebny do zaimplementowania oczekiwanej funkcjonalności. Na tym etapie określane są:

- typy danych
- relacje pomiędzy elementami modelu
- reguły szczegółowe dla wybranych danych (**data annotations**)

6. **Programowanie kontrolerów, akcji i dostosowanie widoków aplikacji ASP.NET Core MVC**

7. **Akcje zwracające dane [HttpGet]**

Studenci uczą się jak zasilić statyczny widok **cshtml** przy pomocy danych zwracanych przez akcję. Na tym etapie dane są zakodowane „na sztywno" jako pole klasy kontrolera. Następuje omówienie użycia składni **Razor** do budowania widoków i wyświetlania danych.

8. **Akcje pobierające dane [HttpPost]**

Studenci identyfikują akcje które będą pobierały dane. Określają typ pobieranych danych i budują logikę walidacji. Następuje omówienie **kodów odpowiedzi HTTP** (200,404,500). W zależności od wyniku walidacji, zwracany jest właściwy **action result** (i właściwy kod HTTP).

9. **Rozbudowa aplikacji o warstwę serwisów**

Studenci identyfikują miejsca w kodzie, gdzie dane są ustawione na sztywno a powinny być zwracane dynamicznie oraz miejsca w kodze, gdzie dane pobierane od użytkownika powinny zostać przetworzone. Na tej podstawie budują klasy serwisów (implementujące interface logiki biznesowej). Następuje omówienie mechanizmu **Dependency Injection** w ASP. NET Core MVC.

10. **Wykorzystanie środowiska Entity Framework Core**

11. **Podstawowa konfiguracja EF Core**

Studenci tworzą **DbContext** na podstawie modelu danych. Poznają sposób określania relacji pomiędzy elementami modelu przy użyciu **fluent API**. Omówiony zostaje sposób konfiguracji **wstrzykiwania DbContextu**. Studenci konfigurują połączenie do bazy **Ms Sql**. Omówione zostaje pojęcie **migracji.** i na tej podstawie studenci tworzą bazę danych odzwierciedlającą model.

12. **Wykorzystanie obiektu DbCotext w serwisach**

Studenci dodają **DbCotext** do urzednio stworzonych serwisów. Implementują podstawowe operacje **CRUD**. Uczą się korzystać z **LINQ** (fluent syntax) do filtrowania dancyh.

13. **Zarządzanie uprawnieniami**

Studenci poznają pakiet **ASP.NET Core Identity**. Omówiony zostaje model użytkownika i struktura tabel. Studenci implementują mechanizm rejestracji nowego użytkownika i logowania do serwisu. Identyfikują, które kontrolery/akcje powinny być dostępne tylko dla zalogowanych użytkowników i poznają sposób wymuszenie takiego zachowania aplikacji.

14. **Wykorzystanie Web API**

Studenci identyfikują funkcjonalność aplikacji web'owej, która mogłaby być realizowana asynchronicznie. Poznają sposób na użycie wywołań **AJAX**'owych. Dane do wywołania zwracane są w formacie **JSON** przy użyciu akcji Web API.

15. **Zarządzanie wydajnością i wymianą informacji**

Wprowadzenie **ResponseCache**. Omówienie powodów dla których można z tego mechanizmu korzystać. Określenie w jakich sytuacjach można cach'ować odpowiedź servera a w jakich należy tego kategorycznie unikać.

16. **Testowanie i rozwiązywanie problemów**

Ogólne omówienie rodzajów testów (jednostkowe, integracyjne, wydajnościowe, smoke testy itp.)

17. **Testy jednostkowe**

Studenci tworzą osobny projekt z testami. Poznają sposób pisania testu jednostkowego (arrange/act/assert). Omówiony zostaje framework MsTest. Studenci piszą test do wybranych akcji.

18. **Wdrażanie aplikacji na system hostujący**

W zależności od dostępu studentów do platformy Azure, następuje prezentacja bądź deployment indywidualnych aplikacji na platformę Azure.