Projektowanie aplikacji ADO.NET + ASP.NET
Zestaw 4

Routing, kontener usług, konfiguracja

2024-10-29

Liczba punktów do zdobycia: **6/32**
Zestaw ważny do: 2024-11-12

1. (**1p**) Powtórzyć zaprezentowany na wykładzie i przedstawiony w notatkach przykład własnej klasy obsługującej Routing w .NET Framework. Przeczytać i rozumieć przedstawione na wykładzie pojęcie **Multitenant Application**.

2. (**1p**) Wzorując się na przedstawionym w notatkach do wykładu przykładzie dynamicznego routingu dla .NET Core, przygotować aplikację która mapuje punkt końcowy dla ścieżki

   - z parametrem
   - nakładającej ograniczenie typu na parametr
   - nakładającej ograniczenie długości na parametr
   - nakładającej ograniczenie wymagalności na parametr
   - opisanej wyrażeniem regularnym

   Wynikiem żądania powinno być wypisanie ścieżki która jest parametrem żądania oraz wartość dopasowanego parametru/parametrów.

   `https://learn.microsoft.com/en-us/aspnet/core/fundamentals/routing?view=aspnetcore-8.0#route-constraints`

   Uwaga, jeśli ścieżki zazębiają się, to wywołanie może zakończyć się wyjątkiem **AmbiguousMatchException**. Pokazać taki przypadek.

   Pokazać że istnieje możliwość określenia własnego selektora punktów końcowych, który rozwiązuje problem.

   (`https://www.wiktorzychla.com/2023/11/ambiguousmatchexception-on-simple.html`)

3. (**2p**) Przedstawiony na wykładzie sposób definiowania usługi (service) dostępu do danych (usługa `IDapperRepository`) zademonstrować na przykładowej aplikacji .NET Core w której obiekt usługi dostępu do danych jest automatycznie wstrzyknięty jako parametr do handlera zmapowanego na jakiś punkt końcowy:

```
app.MapGet( "/", ( ... ? ... ) => {

repository....; // tu chcemy mieć dostęp do danych za pomocą referencji typu IDapperRepository

} );
```

Zwrócić uwagę na prawidłowe zwalnianie zasobów (metoda `Dispose`).

4. (**2p**) Pokazać jak korzystać z plików i parametrów konfiguracyjnych w ASP.NET.

   W szczególności, dla .NET Framework:

   - Pokazać jak definiować dane w sekcjach `appSettings` i `connectionStrings`
   - Pokazać jak odczytywać dane za pomocą klasy `ConfigurationManager`
   - Pokazać jak z poziomu `web.config` wynieść sekcje `appSettings` i `connectionStrings` do osobnych plików (atrybut `ConfigSource`)

   Dla .NET Core:

   - Pokazać jak korzystać z dodatkowych plików konfiguracyjnych (`AddJsonFile`, `AddXmlFile`)
   - Pokazać jak odczytywać dane z plików konfiguracyjnych za pomocą `IConfiguration` i indeksera (`config["foo"]`)
   - Pokazać jak odczytywać dane z plików konfiguracyjnych za pomocą `IConfiguration` i metody `GetSection`
   - Pokazać jak odczytywać dane z plików konfiguracyjnych za pomocą wzorca `IOptions`

Wiktor Zychla