Веб-программирование

Часть 2

Юрий Литвинов
yurii.litvinov@gmail.com

06.12.2019г

Попробуем написать что-нибудь “настоящее”

- Приложение для регистрации на конференцию
- Титульная страница конференции со ссылкой на форму регистрации
- Форма регистрации
  - Как слушатель или как докладчик
- Страница, на которой можно просмотреть всех зарегистрировавшихся

Моделирование предметной области

```
public class Participant
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    public bool? Speaker { get; set; }
}
```

Титульная страница

```
@page
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers

<html>
<body>
    <h1>SEIM-2019</h1>
    <p>Coolest conference ever</p>
    <p>Please register:</p>
    <a asp-page="Register">Register</a>
</body>
</html>
```

Страница регистрации

```
@page
@model WebApplication1.Pages.RegisterModel
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers

<html>
<body>
  <form asp-action="Register" method="post">
    <p>
      <label asp-for="Participant.Name">Your name:</label>
      <input asp-for="Participant.Name" />
    </p>
    <p>
      <label asp-for="Participant.Email">Your email:</label>
      <input asp-for="Participant.Email" />
    </p>
    <p>
      <label>Are you a speaker?</label>
      <select asp-for="Participant.Speaker">
        <option value="">Choose an option</option>
        <option value="true">Yes</option>
        7
        <option value="false">No</option>
      </select>
    </p>
    <button type="submit">Register!</button>
  </form>
</body>
</html>
```

Code behind

```
public class RegisterModel : PageModel
{
    public void OnGet()
    {
    }

    [BindProperty]
    public Data.Participant Participant { get; set; }

    public IActionResult OnPost()
    {
        if (!ModelState.IsValid)
        {
            return Page();
        }

        return RedirectToPage($"/Thanks", new { name = Participant.Name });
    }
}
```

Страница благодарности

```
@page "{name}"

<html>
<body>
   <h1>Thanks for registration, @RouteData.Values["name"]!</h1>
</body>
</html>
```

Добавим немного СУБД

Класс Data.AppDbContext

```
public class AppDbContext: DbContext
{
    public AppDbContext(DbContextOptions options)
        : base(options)
    {
    }

    public DbSet<Participant> Participants { get; set; }
}
```

Но где же Connection String?

- Dependency Injection — архитектурный паттерн, отделяющий создание и конфигурирование объектов от их логики
- За создание и конфигурирование/переконфигурирование отвечает DI-контейнер, обычно отдельная подсистема
- Мы лишь описываем ограничения на то, что должно получиться
- Конкретно в нашем случае:
  - Подключаем пакет Microsoft.EntityFrameworkCore.Sqlite
  - Пишем в Startup.ConfigureServices:

```
services.AddDbContext<Data.AppDbContext>(
    options => options.UseSqlite(@"Data source=Test.db;"));
```

- DI-контейнер создаёт и регистрирует контекст, передавая ему в конструктор опции

Инициализация базы данных

- Идём ВНЕЗАПНО в Tools -> Nuget Package Manager -> Package Manager Console
- Набираем в консоли Add-Migration Initial
  - Создаётся *миграция*, которая создаст базу
- Набираем в консоли Update-Database
  - Миграция применяется, в папке с проектом создаётся Test.db

Пришла пора воспользоваться СУБД

```
public class RegisterModel : PageModel
{
    private readonly AppDbContext db;

    public RegisterModel(AppDbContext db)
    {
        this.db = db;
    }

    [BindProperty]
    public Participant Participant { get; set; }

    public async Task<IActionResult> OnPostAsync()
    {
        if (!ModelState.IsValid)
        {
            return Page();
        }

        db.Participants.Add(Participant);
        await db.SaveChangesAsync();

        return RedirectToPage($"/Thanks", new { name = Participant.Name });
    }
}
```

Ну и последняя страница, просмотр участников

Code Behind

```
public class ViewModel : PageModel
{
    private readonly AppDbContext db;

    public ViewModel(AppDbContext db)
    {
        this.db = db;
    }

    public IList<Participant> Participants { get; private set; }

    public async Task OnGetAsync()
    {
        Participants = await db.Participants.AsNoTracking().ToListAsync();
    }
}
```

.cshtml

```
@page
@model WebApplication1.Pages.ViewModel

<html>
<body>
  <table class="table">
    <thead>
      <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Email</th>
        <th>Is Speaker</th>
      </tr>
    </thead>
    <tbody>
      @foreach (var participant in Model.Participants)
      {
      <tr>
        <td>@participant.Id</td>
        <td>@participant.Name</td>
        <td>@participant.Email</td>
        <td>@(participant.Speaker == true ? "Yes" : "No")</td>
      </tr>
      }
    </tbody>
  </table>
</body>
</html>
```

Но так даже в 90-е не верстали!

Добавим Bootstrap

```
@page
@model WebApplication1.Pages.ViewModel

<html>
<head>
  <link rel="stylesheet" href="/lib/bootstrap/dist/css/bootstrap.css" />
</head>
<body>
  <table class="table table-striped table-bordered">
    <thead>
      <tr>
        <th scope="col">ID</th>
        <th scope="col">Name</th>
        <th scope="col">Email</th>
        <th scope="col">Is Speaker</th>
      </tr>
    </thead>
    <tbody>
      @foreach (var participant in Model.Participants)
      {
      <tr>
        <td>@participant.Id</td>
        <td>@participant.Name</td>
        <td>@participant.Email</td>
        <td>@(participant.Speaker == true ? "Yes" : "No")</td>
      </tr>
      }
    </tbody>
  </table>
</body>
</html>
```

Валидация

- Клиентская
  - Работает с помощью jquery-validation
  - Только в браузере у клиента, без нужды связи с сервером
  - Только если там включён JavaScript
- Серверная
  - Проверка модели при Model Binding-е в контроллере

Атрибуты валидации

```
public class Participant
{
    public int Id { get; set; }

    [Required]
    public string Name { get; set; }

    [Required]
    [RegularExpression(@"^([a-zA-Z0-9_\-\.]+)"
        + @"@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$")]
    public string Email { get; set; }

    [Required(ErrorMessage = "Укажите Вашу роль")]
    public bool? Speaker { get; set; }
}
```

Что ещё бывает

- Required — для nullable-типов требует значение, для не-nullable не имеет смысла, они обязательны всегда
- StringLength — задаёт максимальную и минимальную длину строки
- RegularExpression
- Range — ограничивает значение заданным атрибутом

Что надо сделать на клиенте

```
<form asp-action="Register" method="post">
  <div asp-validation-summary="ModelOnly" class="text-danger"></div>
  <p>
    <label asp-for="Participant.Name">Your name:</label>
    <input asp-for="Participant.Name" />
    <span asp-validation-for="Participant.Name" class="text-danger" />
  </p>
  <p>
    <label asp-for="Participant.Email">Your email:</label>
    <input asp-for="Participant.Email" />
    <span asp-validation-for="Participant.Email" class="text-danger" />
  </p>
  <p>
    <label>Are you a speaker?</label>
    <select asp-for="Participant.Speaker">
      <option value="">Choose an option</option>
      <option value="true">Yes</option>
      <option value="false">No</option>
    </select>
    <span asp-validation-for="Participant.Speaker" class="text-danger" />
  </p>
  <button type="submit">Register!</button>
</form>
...
@section Scripts {
  @{await Html.RenderPartialAsync("_ValidationScriptsPartial"); }
}
```