Веб-программирование

Часть 2

Юрий Литвинов
y.litvinov@spbu.ru

30.11.2021

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

Контроллер

```
using Microsoft.AspNetCore.Mvc;

namespace ConferenceRegistration.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
            => View("MyView");
    }
}
```

Вид

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>MyView</title>
</head>
<body>
Hello, world!
</body>
</html>
```

Моделирование предметной области

```
namespace ConferenceRegistration.Models
{
    public class Participant
    {
        public string Name { get; set; }
        public string Email { get; set; }
        public bool? Speaker { get; set; }
    }
}
```

Страница регистрации

```
@model ConferenceRegistration.Models.Participant
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers

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

Титульная страница

@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers

```
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>SEIM-2021 registration</title>
</head>
<body>
    <div>
        <p>SEIM-2021 conference will be held in April in St. Petersburg.</p>
        <a asp-action="Register">Register now!</a>
    </div>
</body>
</html>
```

Подправим контроллер

```
public class HomeController : Controller
{
    public IActionResult Index()
        => View("MyView");

    public IActionResult Register()
        => View();
}
```

Подправим контроллер ещё раз

```
public class HomeController : Controller
{
    public IActionResult Index()
        => View("MyView");

    [HttpGet]
    public IActionResult Register()
        => View();

    [HttpPost]
    public IActionResult Register(Participant participant)
    {
        // TODO: Do something with registration info
        return View();
    }
}
```

Репозиторий

```
namespace ConferenceRegistration.Models
{
    public static class Repository
    {
        private static readonly IList<Participant> participants
            = new List<Participant>();

        public static IEnumerable<Participant> Participants
            => participants;

        public static void AddParticipant(Participant participant)
            => participants.Add(participant);
    }
}
```

Контроллер с репозиторием

```
public class HomeController : Controller
{
    public IActionResult Index()
        => View("MyView");

    [HttpGet]
    public IActionResult Register()
        => View();

    [HttpPost]
    public IActionResult Register(Participant participant)
    {
        Repository.AddParticipant(participant);
        return View();
    }
}
```

Страница подтверждения регистрации

```
@model ConferenceRegistration.Models.Participant

<html>
<head>
   <meta name="viewport" content="width=device-width" />
   <title>Thanks</title>
</head>
<body>
<p>
   <h1>Thank you, @Model.Name</h1>
</p>
<p>
   @if (Model.Speaker == true)
   {
      @:Please don't forget to submit your article!
   }
</p>
</body>
</html>
```

Страница со списком участников

```
@model IEnumerable<ConferenceRegistration.Models.Participant>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>ListParticipants</title>
</head>
<body>
<h2>List of conference participants:</h2>
<table>
    <thead>
    <tr>
        <th>Name</th>
        <th>Email</th>
        <th>Is speaker</th>
    </tr>
    </thead>
    <tbody>
    @foreach (ConferenceRegistration.Models.Participant p in Model) {
        <tr>
            <td>@p.Name</td>
            <td>@p.Email</td>
            <td>@(p.Speaker == true ? "Yes" : "No")</td>
        </tr>
    }
    </tbody>
</table>
</body>
</html>
```

Обновим контроллер

```
public class HomeController : Controller
{
    public IActionResult Index()
        => View("MyView");

    [HttpGet]
    public IActionResult Register()
        => View();

    [HttpPost]
    public IActionResult Register(Participant participant)
    {
        Repository.AddParticipant(participant);
        return View();
    }

    public IActionResult ListParticipants()
        => View(Repository.Participants);
}
```

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

Добавим server-side-валидацию

```
public class Participant
{
    [Required(ErrorMessage = "Please enter your name")]
    public string Name { get; set; }

    [Required(ErrorMessage = "Please enter your email")]
    [RegularExpression(".+\\@.+\\..+",
        ErrorMessage = "Please enter a valid email address")]
    public string Email { get; set; }

    [Required(ErrorMessage =
        "Please specify whether you'll be a speaker or just attending")]
    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

И снова обновим контроллер

```
public class HomeController : Controller
{
    ...
    [HttpPost]
    public IActionResult Register(Participant participant)
    {
        if (ModelState.IsValid)
        {
            Repository.AddParticipant(participant);
            return View("Thanks", participant);
        }

        return View();
    }
    ...
}
```

А теперь и вид

```
<form asp-action="Register" method="post">
    <div asp-validation-summary="All"></div>
    ...
</form>
```

Добавим немного CSS

```css
.field-validation-error {
    color: #f00;
}

.field-validation-valid {
    display: none;
}

.input-validation-error {
    border: 1px solid #f00;
    background-color: #fee;
}

.validation-summary-errors {
    font-weight: bold;
    color: #f00;
}

.validation-summary-valid {
    display: none;
}
```

И вставим .css-ку в форму

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

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
    <link rel="stylesheet" href="/lib/bootstrap/dist/css/bootstrap.css" />
</head>
<body>
    <div class="text-center">
        <h3>SEIM-2021 conference will be held in April in St. Petersburg.</h3>
        <a class="btn btn-primary" asp-action="Register">Register now!</a>
    </div>
</body>
</html>
```

То же с...

- формой регистрации
- страницей со списком участников
- страницей подтверждения регистрации
  - см. конспект

Добавим персистентность

- Добавим Microsoft.EntityFrameworkCore, Microsoft.EntityFrameworkCore.SqlServer

```
public class Repository : DbContext
{
    public DbSet<Participant> Participants { get; set; }

    protected override void OnConfiguring(
        DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer(
            @"Server=(localdb)\mssqllocaldb;"
            + @"Database=ConferenceRegistration;Trusted_Connection=True;");
    }
}
```

И модифицируем контроллер

```
public class HomeController : Controller
{
    ...
    [HttpPost]
    public IActionResult Register(Participant participant)
    {
        if (ModelState.IsValid)
        {
            using var repository = new Repository();
            repository.Participants.Add(participant);
            repository.SaveChanges();
            return View("Thanks", participant);
        }

        return View();
    }

    public IActionResult ListParticipants()
    {
        using var repository = new Repository();
        return View(repository.Participants.ToList());
    }
}
```

И модель

Добавим первичный ключ

```
public class Participant
{
    [Required(ErrorMessage = "Please enter your name")]
    public string Name { get; set; }

    [Required(ErrorMessage = "Please enter your email")]
    [RegularExpression(".+\\@.+\\..+",
        ErrorMessage = "Please enter a valid email address")]
    [Key]
    public string Email { get; set; }

    [Required(ErrorMessage =
        "Please specify whether you'll be a speaker or just attending")]
    public bool? Speaker { get; set; }
}
```

А теперь магически поднимем базу данных

EF Migrations

- Поставим Microsoft.EntityFrameworkCore.Tools
- Откроем NuGet Manager Console
- Напишем Add-Migration InitialCreate
- Напишем Update-Database

Проверим, что всё работает

- Запустим приложение, зарегистрируем пару участников
- Откроем Server Explorer
- Connect to Database -> Microsoft SQL Server
- Имя сервера -> (localdb)\mssqllocaldb
- Имя файла -> ConferenceRegistration
  - Как в Connection String
- Проверяем, что база имеет таблицу Participants
- Просматриваем данные
- Перезапускаем сервер, перезапускаем приложение, смотрим список участников