

# Веб-программирование

## Часть 2

Юрий Литвинов  
y.litvinov@spbu.ru

30.11.2022

# 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
- ▶ Итого, многостраничное приложение на Razor Pages
- ▶ Create a new project -> ASP.NET Core Web App

## Index.cshtml.cs

Немного обновим сгенерированный шаблон:

```
namespace ConferenceRegistration.Pages;
```

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

И уберём using в Program.cs

# Вид

@page

*<!DOCTYPE html>*

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

# Моделирование данных

```
namespace ConferenceRegistration.Pages;
```

```
[BindProperties]
```

```
public class RegistrationModel : PageModel
```

```
{
```

```
    public string Name { get; set; } = "";
```

```
    public string Email { get; set; } = "";
```

```
    public bool IsSpeaker { get; set; }
```

```
}
```

# Страница регистрации

@page

@model ConferenceRegistration.Pages.RegistrationModel

@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="IsSpeaker">
          <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

@page

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

```
<html>
  <head>
    <meta name="viewport" content="width=device-width" />
    <title>SEIM-2022 registration</title>
  </head>
  <body>
    <div>
      <p>SEIM-2022 conference will be held in April in St. Petersburg.</p>
      <a asp-page="Registration">Register now!</a>
    </div>
  </body>
</html>
```

## Вернёмся к странице регистрации

```
namespace ConferenceRegistration.Pages;
```

```
[BindProperties]
```

```
public class RegistrationModel : PageModel
```

```
{
```

```
    public string Name { get; set; } = "";
```

```
    public string Email { get; set; } = "";
```

```
    public bool IsSpeaker { get; set; }
```

```
    public void OnPost()
```

```
    {
```

```
        // TODO: Do something with registration info.
```

```
    }
```

```
}
```



# Работа с базой

Модель данных, Data.Participant.cs

```
namespace ConferenceRegistration.Data;
```

```
public class Participant
```

```
{  
    public string Name { get; set; } = "";  
  
    public string Email { get; set; } = "";  
  
    public bool IsSpeaker { get; set; }  
}
```

# Обновим модель

```
using ConferenceRegistration.Data;

namespace ConferenceRegistration.Pages;

[BindProperties]
public class RegistrationModel : PageModel
{
    public Participant Participant { get; set; } = new();

    public void OnPost()
    {
        // TODO: Do something with registration info.
    }
}
```

# И представление

@page

@model ConferenceRegistration.Pages.RegistrationModel

@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="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.IsSpeaker">
          <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>
```

# DbContext

```
namespace ConferenceRegistration.Data;
using Microsoft.EntityFrameworkCore;

public class ConferenceRegistrationDbContext: DbContext
{
    public ConferenceRegistrationDbContext(
        DbContextOptions<ConferenceRegistrationDbContext> options)
        : base(options)
    {
    }

    public DbSet<Participant> Participants => Set<Participant>();
}
```

# Модель с DbContext

```
namespace ConferenceRegistration.Pages;  
using ConferenceRegistration.Data;
```

```
[BindProperties]
```

```
public class RegistrationModel : PageModel
```

```
{
```

```
    private readonly ConferenceRegistrationDbContext _context;
```

```
    public RegistrationModel(ConferenceRegistrationDbContext context)  
        => _context = context;
```

```
    public Participant Participant { get; set; } = new();
```

```
    public async Task<IActionResult> OnPostAsync()
```

```
    {
```

```
        _context.Participants.Add(Participant);
```

```
        await _context.SaveChangesAsync();
```

```
        return RedirectToPage("./Index");
```

```
    }
```

```
}
```

# Конфигурация базы

```
global using Microsoft.AspNetCore.Mvc;  
global using Microsoft.AspNetCore.Mvc.RazorPages;
```

```
using ConferenceRegistration.Data;  
using Microsoft.EntityFrameworkCore;
```

```
var builder = WebApplication.CreateBuilder(args);
```

```
// Add services to the container.  
builder.Services.AddRazorPages();
```

```
builder.Services.AddDbContext<ConferenceRegistrationDbContext>(options =>  
    options.UseSqlite("Data Source=conferenceRegistration.db"));
```

```
var app = builder.Build();
```

## Добавим первичный ключ в модель данных

```
namespace ConferenceRegistration.Data;
```

```
public class Participant
```

```
{
```

```
    public int ParticipantId { get; set; }
```

```
    public string Name { get; set; } = "";
```

```
    public string Email { get; set; } = "";
```

```
    public bool IsSpeaker { get; set; }
```

```
}
```

# Миграции

- ▶ Миграции — механизм обеспечения эволюции схемы БД
- ▶ Генерируются автоматически по коду
- ▶ `dotnet tool install --global dotnet-ef`
- ▶ `dotnet add package Microsoft.EntityFrameworkCore.Design`
- ▶ `dotnet ef migrations add InitialCreate`
- ▶ `dotnet ef database update`
- ▶ Последний шаг надо применять каждый раз при создании базы



# Список участников

## ListParticipants.cshtml

@page

@model ConferenceRegistration.Pages.ListParticipantsModel

```
<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.Data.Participant p in Model.Participants) {
          <tr>
            <td>@p.Name</td>
            <td>@p.Email</td>
            <td>@(p.Speaker ? "Yes" : "No")</td>
          </tr>
        }
      </tbody>
    </table>
  </body>
</html>
```

# Модель

```
namespace ConferenceRegistration.Pages;  
using ConferenceRegistration.Data;
```

```
public class ListParticipantsModel : PageModel  
{  
    private readonly ConferenceRegistrationDbContext context;  
  
    public ListParticipantsModel(ConferenceRegistrationDbContext context)  
        => this.context = context;  
  
    public IList<Participant> Participants { get; private set; } = new List<Participant>();  
  
    public void OnGet()  
    {  
        Participants = context.Participants.OrderBy(p => p.ParticipantId).ToList();  
    }  
}
```

# Страница подтверждения регистрации

Thanks.cshtml

@page

@model ConferenceRegistration.Pages.ThanksModel

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

## И модель для неё

```
namespace ConferenceRegistration.Pages;  
using ConferenceRegistration.Data;
```

```
public class ThanksModel : PageModel  
{  
    public Participant Participant { get; set; } = new();  
  
    public void OnGet(Participant participant)  
    {  
        Participant = participant;  
    }  
}
```

## И редирект на страницу

```
...
public class RegistrationModel : PageModel
{
    ...
    public async Task<IActionResult> OnPostAsync()
    {
        context.Participants.Add(Participant);
        await context.SaveChangesAsync();

        return RedirectToPage("./Thanks", Participant);
    }
}
```

# Оформление

## Bootstrap

@page

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

```
<html>
  <head>
    <meta name="viewport" content="width=device-width" />
    <title>SEIM-2022 registration</title>
    <link rel="stylesheet" href="/lib/bootstrap/dist/css/bootstrap.css" />
  </head>
  <body>
    <div class="text-center">
      <h3>SEIM-2022 conference will be held in April in St. Petersburg.</h3>
      <a class="btn btn-primary" asp-page="Registration">Register now!</a>
    </div>
  </body>
</html>
```

# Форма регистрации

## Лейауты

```

<div class="row mb-3 text-center"><h4 class="col-sm-6">Registration form</h4></div>
<form asp-action="Register" method="post">
  <div class="row mb-3">
    <label class="col-sm-1 col-form-label col-form-label-lg"
      asp-for="Participant.Name">Your name:</label>
    <div class="col-sm-4">
      <input class="form-control form-control-lg"
        asp-for="Participant.Name" />
    </div>
  </div>
  <div class="row mb-3">
    <label class="col-sm-1 col-form-label col-form-label-lg"
      asp-for="Participant.Email">Your email:</label>
    <div class="col-sm-4">
      <input class="form-control form-control-lg"
        asp-for="Participant.Email" />
    </div>
  </div>
  ...
  <div class="row mb-3 mx-auto">
    <div class="col-sm-5 d-grid gap-2">
      <button class="btn btn-primary btn-lg" type="submit">
        Register!
      </button>
    </div>
  </div>
</form>

```

# Список участников

@page

@model ConferenceRegistration.Pages.ListParticipantsModel

```
<html>
  <head>
    <meta name="viewport" content="width=device-width" />
    <title>ListParticipants</title>
    <link rel="stylesheet" href="/lib/bootstrap/dist/css/bootstrap.css" />
  </head>
  <body>
    <div class="row mb-3 text-center"><h2 class="col-sm-6">List of conference participants</h2></div>
    <table class="table table-striped table-bordered">
      <thead>
        <tr>
          <th>Name</th>
          <th>Email</th>
          <th>Is speaker</th>
        </tr>
      </thead>
      <tbody>
        @foreach (ConferenceRegistration.Data.Participant p in Model.Participants) {
          <tr>
            <td>@p.Name</td>
            <td>@p.Email</td>
            <td>@(p.IsSpeaker ? "Yes" : "No")</td>
          </tr>
        }
      </tbody>
    </table>
  </body>
</html>
```



# Декларативная валидация

```
namespace ConferenceRegistration.Data;  
using System.ComponentModel.DataAnnotations;
```

```
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; }
```

```
}
```

# Модель

```
...  
public class RegistrationModel : PageModel  
{  
    ...  
    public async Task<IActionResult> OnPostAsync()  
    {  
        if (!ModelState.IsValid)  
        {  
            return Page();  
        }  
  
        context.Participants.Add(Participant);  
        await context.SaveChangesAsync();  
  
        return RedirectToPage("./Thanks", Participant);  
    }  
}
```

# Представление

```

...
<div class="row mb-3 text-center"><h4 class="col-sm-6">Registration form</h4></div>
<form asp-action="Register" method="post">
  <div class="row mb-3">
    <span asp-validation-for="Participant.Name" class="text-danger"></span>
    ...
  </div>
  <div class="row mb-3">
    <span asp-validation-for="Participant.Email" class="text-danger"></span>
    ...
  </div>
  <div class="row mb-3">
    <span asp-validation-for="Participant.IsSpeaker" class="text-danger"></span>
    ...
  </div>
  <div class="row mb-3 mx-auto">
    <div class="col-sm-5 d-grid gap-2">
      <button class="btn btn-primary btn-lg" type="submit">
        Register!
      </button>
    </div>
  </div>
</form>
...

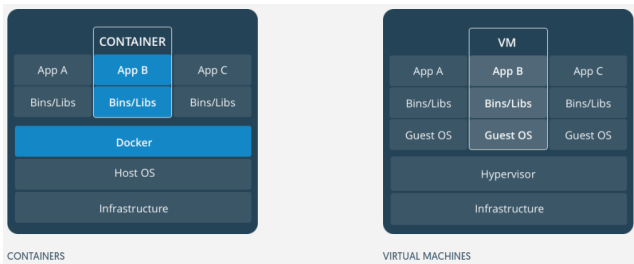
```

# Client-side-валидация

```
...  
<head>  
  <meta name="viewport" content="width=device-width" />  
  <title>Register</title>  
  <link rel="stylesheet" href="/lib/bootstrap/dist/css/bootstrap.css" />  
  <script src="/lib/jquery/dist/jquery.js"></script>  
  <script src="/lib/jquery-validation/dist/jquery.validate.js"></script>  
  <script src="/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js" />  
  </script>  
</head>  
...
```

# Docker

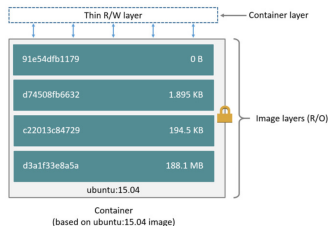
- ▶ Средство для “упаковки” приложений в изолированные контейнеры
- ▶ Что-то вроде легковесной виртуальной машины
- ▶ DSL для описания образов
- ▶ Публичный репозиторий
- ▶ Стандарт де-факто для деплоя веб-приложений



© <https://www.docker.com>

# Docker Image

- ▶ Окружение и приложение
- ▶ Состоит из слоёв
  - ▶ Все слои read-only
  - ▶ Образы делят слои между собой как процессы делят динамические библиотеки
- ▶ На основе одного образа можно создать другой



# Dockerfile

```
FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443
```

```
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
WORKDIR /src
COPY ["ConferenceRegistration/ConferenceRegistration.csproj", "ConferenceRegistration/"]
RUN dotnet restore "ConferenceRegistration/ConferenceRegistration.csproj"
COPY . .
WORKDIR "/src/ConferenceRegistration"
RUN dotnet build "ConferenceRegistration.csproj" -c Release -o /app/build
```

```
FROM build AS publish
RUN dotnet publish "ConferenceRegistration.csproj" -c Release -o /app/publish
```

```
FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "ConferenceRegistration.dll"]
```

# Пушим на Docker Hub

- ▶ Регистрируемся на Docker Hub
- ▶ `docker images ls`
- ▶ `docker image tag conferenceregistration:latest <ваш юзернейм на Docker Hub>/conferenceregistration:latest`
- ▶ `docker push <ваш юзернейм на Docker Hub>/conferenceregistration:latest`
- ▶ `docker run -d -p 80:80 <ваш юзернейм на Docker Hub>/conferenceregistration:latest`



# Azure

- ▶ Облачный хостинг с в том числе бесплатным планом
- ▶ Регистрируемся на <https://azure.microsoft.com/>
- ▶ Логинимся и переходим по ссылке на <https://portal.azure.com/>
- ▶ App Services -> Create
- ▶ Pay-As-You-Go, Resource Group -> Create New, имя приложения, Publish — Docker Container, Operating System — Linux
- ▶ Region — East US, Sku and size — бесплатный

## Azure (2)

- ▶ Single container, Image Source -> Docker Hub, Access Type -> Public
- ▶ Image and Tag — имя образа на Docker Hub
- ▶ Review + create, Create
- ▶ Идём на домашнюю страницу Azure, находим там приложение
- ▶ Browse