

C#. ASP.NET MVC, загрузка файлов

лектор: Юрий Коноплев

07.08.2019

КОММУНИКАЦИЯ

- - telegram: @bimodaling
- - tele-group: t.me/PIITSchool (Phoenix Ingostrah IT School)
- - IRC chat.freenode.net (mIRC, HexChat): #itweeks
- - mail: yuri@silkmind.com
- - https://github.com/j0k/it_school_weeks

План на неделю

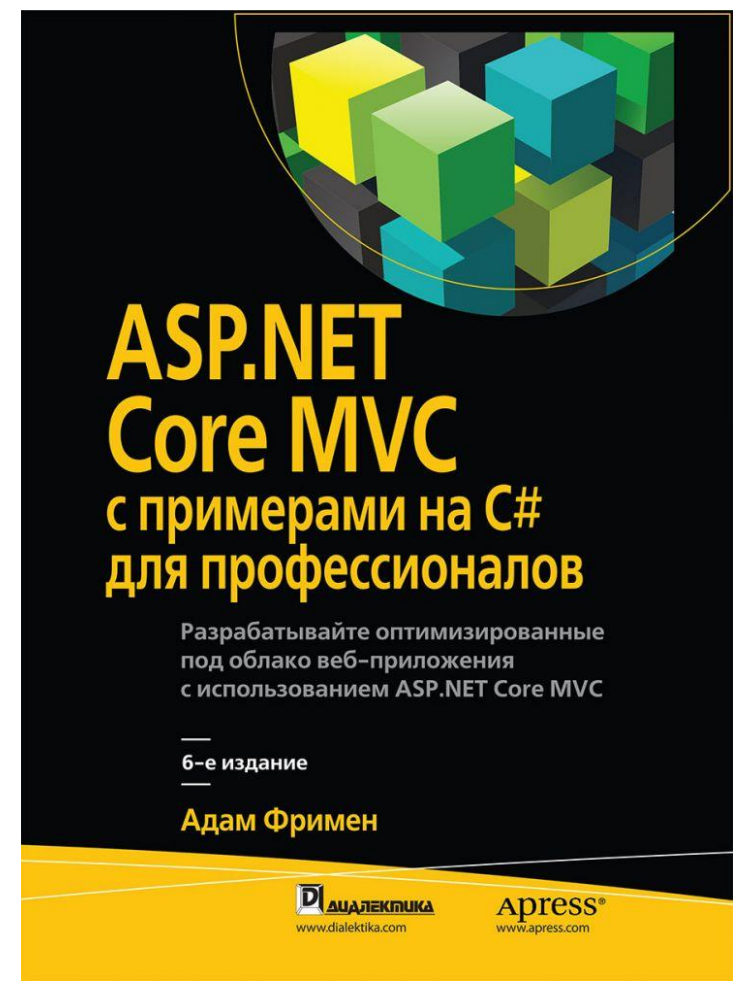
- MVC
- Взаимодействие фронта и бэка
- Динамический контент
- Model
- View
- Controller
- Загрузка файлов
- JSON и API
- Атрибуты

План на день

- Вчерашний день
- Паттерн MVC
- Загрузка файлов

Материалы

- Ссылки - https://github.com/j0k/it_school_weeks/



Вспоминая прошлую неделю
ООП, свойства, generics, lambda

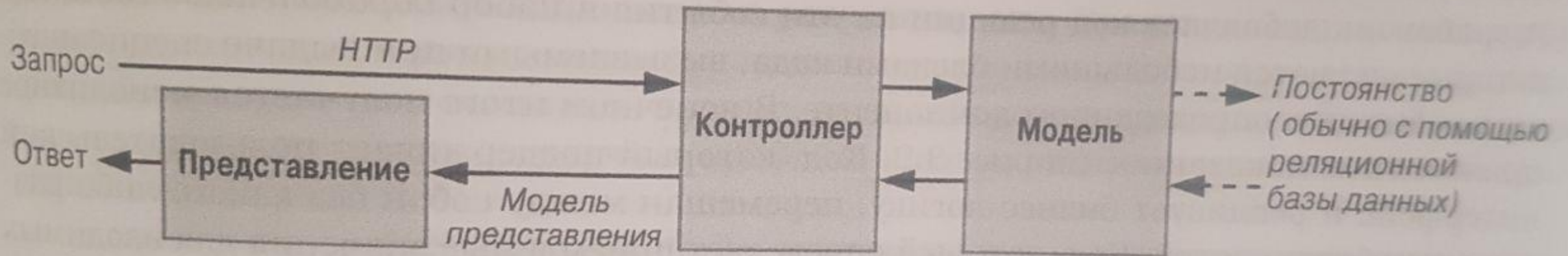


Рис. 3.1. Взаимодействия в приложении MVC

Паттерн MVC

MVC

- Термин **модель-представление-контроллер** был в употреблении с конца 1970-х годов и происходит из проекта Smalltalk в Xerox PARC.

Модель

- Модели содержат данные, с которыми работают пользователи.
- Модели - это определение "вселенной", в которой функционирует приложение. Например в банковском приложении модели представляют все аспекты банковской деятельности.

Модель

- Модель в приложении, построенном с использованием паттерна MVC, **должна**
 - Содержать данные предметной области
 - Содержать логику для создания, управления и модификации данных предметной области
 - Представлять чистый API-интерфейс, который открывает доступ к данным модели и операциям с ними

Модель

- Модель **не должна**:
 - Показывать детали того, как осуществляется получение или управление данными модели (подробности механизма хранения данных не должны быть видны контроллерам и представлениям)
 - Содержать логику, которая трансформирует модель на основе взаимодействия с пользователем (это работа контроллера)
 - Содержать логику для отображения данных пользователю (это работа отображения)

FileUpload

- <https://docs.microsoft.com/en-us/aspnet/core/mvc/models/file-uploads?view=aspnetcore-2.1>
- https://www.youtube.com/watch?v=9LPlu_BvatE
- <https://gunnarpeipman.com/aspnet/aspnet-core-content-web-root/>

FileUpload: Model (Picture.cs)

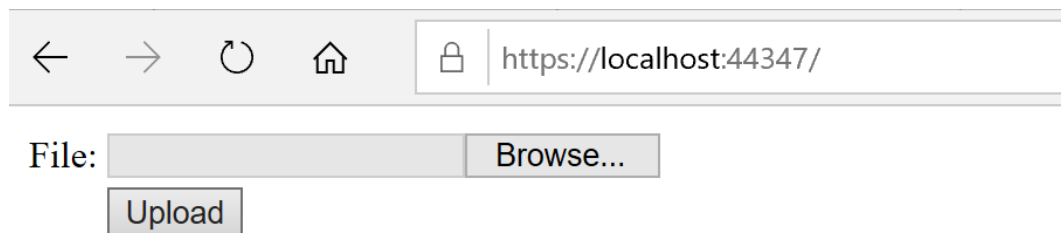
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;

namespace FileUpload.Models
{
    4 references
    public class Picture
    {
        0 references | 0 exceptions
        public IFormFile File { get; set; }
    }
}
```

FileUpload: View (Index.cshtml)

```
Index.cshtml* Picture.cs FileUpload
1 @model Picture
2 @{
3     ViewData["Title"] = "Index";
4 }
5
6 @using (Html.BeginForm(null, null, FormMethod.Post, new { enctype = "multipart/form-data" }))
7 {
8     <table>
9     <tr>
10         <td>File:</td>
11         <td>
12             <input type="file" name="File" id="file"/>
13         </td>
14     </tr>
15     <tr>
16         <td>&nbsp;</td>
17         <td><input type="submit" name="submit" value="Upload"/></td>
18     </tr>
19 </table>
20 }
```

FileUpload



A mockup of a web interface for file uploads. It features a browser-like header with navigation icons (back, forward, refresh, home) and a lock icon followed by the URL 'https://localhost:44347/'. Below the header is a file selection area with the label 'File:', a text input field, and a 'Browse...' button. Underneath the input field is an 'Upload' button. The entire interface is framed by a dark gray L-shaped border on the left and bottom.

← → ↻ 🏠 🔒 https://localhost:44347/

File: Browse...

Upload

© 2019 - FileUpload

FileUpload: Controller (HomeController.cshtml)

```
HomeController.cs  _Layout.cshtml  Index.cshtml  Picture.cs  FileUpload
FileUpload
11 namespace FileUpload.Controllers
12 {
13     1 reference
14     public class HomeController : Controller
15     {
16         private readonly IHostingEnvironment _host;
17         0 references | 0 exceptions
18         public HomeController(IHostingEnvironment host)
19         {
20             _host = host;
21         }
22         0 references | 0 requests | 0 exceptions
23         public IActionResult Index()
24         {
25             return View();
26         }
27
28         [HttpPost]
29         0 references | 0 requests | 0 exceptions
30         public async Task<IActionResult> Index(Picture picture)
31         {
32             if (picture.File.Length > 0)
33             {
34                 var filename = Path.GetFileName(picture.File.FileName);
35                 var path = Path.Combine(_host.WebRootPath + "/upload", filename);
36                 using (var stream = new FileStream(path, FileMode.Create))
37                 {
38                     await picture.File.CopyToAsync(stream);
39                 }
40             }
41             return View();
42         }
43     }
44 }
```

```
HomeController.cs  X  _Layout.cshtml  Index.cshtml
FileUpload
1  using System;
2      using System.Collections.Generic;
3      using System.Diagnostics;
4      using System.Linq;
5      using System.Threading.Tasks;
6      using Microsoft.AspNetCore.Mvc;
7      using FileUpload.Models;
8      using System.IO;
9      using Microsoft.AspNetCore.Hosting;
10
```

FileUpload: Controller
(HomeController.cshtml)

ASP.NET

<https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.2>

Introduction to ASP.NET Core

04/07/2019 • 5 minutes to read •  +10

By [Daniel Roth](#), [Rick Anderson](#), and [Shaun Luttin](#)

ASP.NET Core is a cross-platform, high-performance, [open-source](#) framework for building modern, cloud-based, Internet-connected applications. With ASP.NET Core, you can:

- Build web apps and services, [IoT](#) apps, and mobile backends.
- Use your favorite development tools on Windows, macOS, and Linux.
- Deploy to the cloud or on-premises.
- Run on [.NET Core](#) or [.NET Framework](#).

Create Empty Web Application

```
namespace WebApplication4
{
    1 reference
    public class Startup
    {
        public static Random rnd = new Random();
        // This method gets called by the runtime. Use this method to add services to the container.
        // For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?linkid=213254
        0 references | 0 exceptions
        public void ConfigureServices(IServiceCollection services)
        {
        }

        // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
        0 references | 0 exceptions
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.Run(async (context) =>
            {
                int i = rnd.Next(10);
                await context.Response.WriteAsync("Hello random world: " + i.ToString());
            });
        }
    }
}
```

Сессия livecode: qsort в бэж

- https://github.com/j0k/it_school_weeks/tree/master/week4/WebQSort/WebQSort

