



Mentoring: ASP.NET MVC

ASP.NET MVC: Controller

План занятия

- Методы действий (action methods)
- Результаты действий (action results)
- Связывание моделей
- Загрузка и выгрузка файлов

Контроллер

- Отвечает за взаимодействие с пользователем;
- Экземпляр контроллера создаётся на каждый запрос пользователя;
- Класс должен реализовать интерфейс `IController` или наследоваться от `Controller`;
- Имя должно заканчиваться на `Controller`;
- Состоит из методов действий.

Ручная реализация IController

```
public class ManualController : IController
{
    public void Execute(RequestContext requestContext)
    {
        var context = requestContext.HttpContext;
        var response = context.Response;
        var action = requestContext.RouteData.Values["action"] as string;
        var x = int.Parse(context.Request.Params["x"] as string);
        var y = int.Parse(context.Request.Params["y"] as string);

        int result;

        switch (action)
        {
            case "Add":
                result = x + y;
                break;

            case "Subtract":
                result = x - y;
                break;

            default:
                response.Write(string.Format("<p>Unknown action: {0}</p>", action));
                return;
        }

        response.Write(string.Format("<p>Result: {0}</p>", result));
    }
}
```

Наследование от Controller

```
public class MathController : Controller
{
    public string Add(int x, int y)
    {
        return "Result: " + (x + y);
    }

    public string Subtract(int x, int y)
    {
        return "Result: " + (x - y);
    }
}
```

- Метод должен быть открытым (public) и экземплярным;
- Рекомендуется возвращать ActionResult или его потомков;
- Атрибуты [HttpGet], [HttpPost] и др. позволяют явно указать HTTP-метод, для которого применяется данный метод действия.

Типы результатов методов действий

ActionResult :

- EmptyResult
- ContentResult
- FileContentResult
- FilePathResult
- FileStreamResult
- HttpStatusCodeResult
 - HttpNotFoundResult
 - HttpUnauthorizedResult
- JavaScriptResult
- JsonResult
- RedirectResult
- RedirectToRouteResult
- PartialViewResult
- ViewResult

EmptyResult

- Возвращается пустой ответ с кодом 200 (OK).
- Способы получения:
 - `new EmptyResult();`
 - `null;`
 - Метод действия, «возвращающий» `void`.
- Не рекомендуется к применению.

ContentResult

- Возвращается указанный контент.
- Способы получения:
 - Content(контент);
 - строка;
 - любой объект (будет приведён к строке).

ViewResult

- Формирует представление и возвращает страницу.
- Способ получения:
 - View(представление, модель)

PartialViewResult

- Формирует частичное представление и возвращает фрагмент страницы.
- Способ получения:
 - PartialView(представление, модель)

JavaScriptResult

- Возвращает скрипт на js для выполнения на клиентской стороне.
- Способ получения:
 - JavaScript(скрипт)

- `FileContentResult` — возвращает файл, заданный массивом байтов;
- `FilePathResult` — возвращает файл, заданный локальной ссылкой на файл на сервере;
- `FileStreamResult` — возвращает файл, заданный потоком;
- Способ получения:
 - `File(...)`
- Обязательно требуется указание типа содержимого файла (MIME-type).

- Возвращает произвольный объект в нотации JSON.
- Применяется, как правило, в сочетании с AJAX на клиентской стороне.
- Способ получения:
 - `Json(объект)`
- Также можно указать тип содержимого (MIME-type) и поведение при GET-запросе (по умолчанию запрещено).

- Возвращает статусный код перенаправления — 301 (навсегда) или 302 (временно) — и URL-адрес назначения.
- Способ получения:
 - `Redirect(url)` — код 302
 - `RedirectPermanent(url)` — код 301
 - `RedirectToAction(метод, контроллер)`
 - `RedirectToActionPermanent(метод, контроллер)`

RedirectToRouteResult

- Возвращает статусный код перенаправления и маршрут назначения.
- Маршрут может быть задан либо именем, либо набором компонентов.
- Способ получения:
 - RedirectToRoute(маршрут);
 - RedirectToRoutePermanent(маршрут);

HttpStatusCodeResult

- Возвращает код статуса запроса.
- Существуют конкретизированные потомки:
 - `HttpUnauthorizedResult` — код 401
 - `HttpNotFoundResult` — код 404
- Способ получения:
 - `HttpNotFound()`
 - `new HttpStatusCodeResult(код, описание)`

```
@using (Html.BeginForm("Sum", "Home", FormMethod.Get))  
{  
    <p>  
        X: <input type="text" name="x" />  
    </p>  
    <p>  
        Y: <input type="text" name="y" />  
    </p>  
    <input type="submit" value="Send" />  
}
```

При формировании страницы будет сгенерирован URL для GET-запроса к методу Sum контроллера Home.

При отправке запроса будут переданы параметры x и y со значениями, равными тексту в соответствующих полях ввода.

- При получении запроса инфраструктура ASP.NET MVC пытается связать полученные параметры запроса с формальными аргументами метода действия.
- Способы связывания являются частью инфраструктуры, однако могут быть дополнены/изменены.
- Если для формального аргумента не было передано совместимого параметра, он получает значение по умолчанию.

Связывание простых значений

- Запрос: /Home/Sum?**x**=3&**y**=5
- Параметры: $x = 3$, $y = 5$.
- Варианты связывания:

– непосредственно в аргументы:

```
public string Sum(int x, int y)
{
    return (x + y).ToString();
}
```

– в поля объекта:

```
public string Sum(Point p)
{
    return (p.X + p.Y).ToString();
}
```

Чуть сложнее:

```
public class User
{
    public string Name { get; set; }
    public User Boss { get; set; }
}
```

- Запрос:

/Home/Get?name=John&boss.name=Alex

```
30 public string Get(User user)
31 {
32     return user.Name;
33 }
34
35
```

als	
ame	Value
this	{WebApplication4.Controllers.HomeController}
user	{WebApplication4.Models.User}
Boss	{WebApplication4.Models.User}
Boss	null
Name	"Alex"
Name	"John"

Применение загрузки файлов

- Отображение динамически подгружаемых изображений с сервера.
- Динамически формирующиеся отчёты/выгрузки.
- Загрузка обычных файлов, ранее загруженных на сервер пользователями.

Загрузка файла с сервера

- Для загрузки файла с сервера необходим метод действия, возвращающий `FileResult` (одного из его потомков).
- Как правило, для его создания применяется метод контроллера `File`.
- Браузер обязан знать тип содержимого файла.

```
public ActionResult GetImage(int userId)
{
    return File(Logic.LoadUserImage(userId), "image/jpeg");
}
```

Загрузка файла на сервер

- HTML-форма должна поддерживать передачу файлов:
 - Тип запроса: POST
 - Тип кодирования: `enctype="multipart/form-data"`
 - Поле для передачи файла: `<input type="file" ... >`

```
@using (Html.BeginForm("Upload", "File", FormMethod.Post,  
    new { enctype = "multipart/form-data" }))  
{  
    <input type="file" name="uploaded" />  
    <input type="submit" value="Upload" />  
}
```


- Связывание с типом `HttpPostedFileBase`

```
[HttpPost]
public ActionResult Upload(HttpPostedFileBase uploaded)
{
```

- Доступ через `Request.Files["имя параметра"]`

```
var uploaded = Request.Files["uploaded"];
var contentLength = uploaded.ContentLength;
var contentType = uploaded.ContentType;
var fileName = uploaded.FileName;

var inputStream = uploaded.InputStream;

uploaded.SaveAs("localFile");
```

Спасибо за внимание!

Контактная информация:

Дмитрий Верескун

Инструктор

EPAM Systems, Inc.

Адрес: Саратов, Рахова, 181

Email: Dmitry_Vereskun@epam.com

<http://www.epam.com>