# Введение в ASP.NET MVC

ASP.NET MVC 4.0

2013

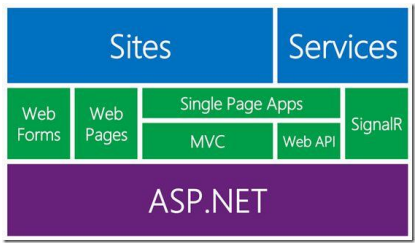
# История ASP.NET

- 1996 – ASP – Active Server Pages, построение страниц на сервере на основе шаблонов. Шаблоны сочетали код на VB с HTML-разметкой.

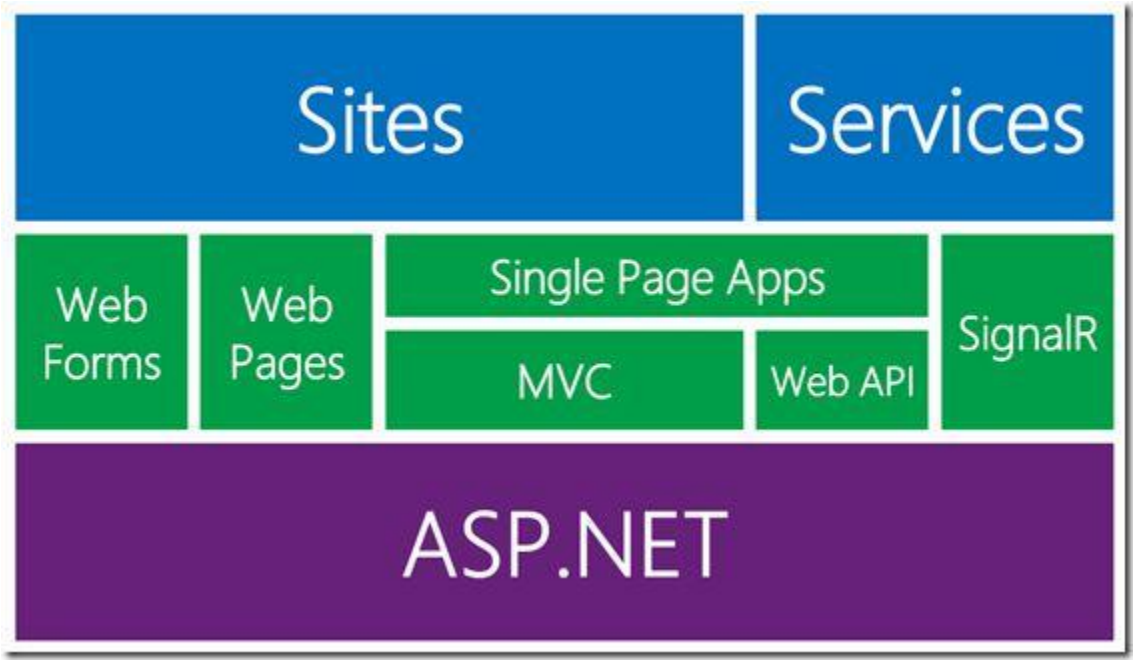

- 2001 – ASP.NET – Составная часть новой платформы .NET. Технология WebForms, по аналогии с WinForms.


- 2009 – ASP.NET MVC. Аналогична уже существующим на рынке подходам: Java Spring 2002, Python Jango 2003 и др.


- 2013 – ASP.NET MVC 5.0 – октябрь, последняя версия



# Фреймворки на базе ASP.NET




ASP.NET – бесплатный фреймворк для построения больших веб-приложений с использованием HTML, CSS и JavaScript.

WebForms – технология построение веб-приложений из стандартных управляющих элементов и обработчиков событий.

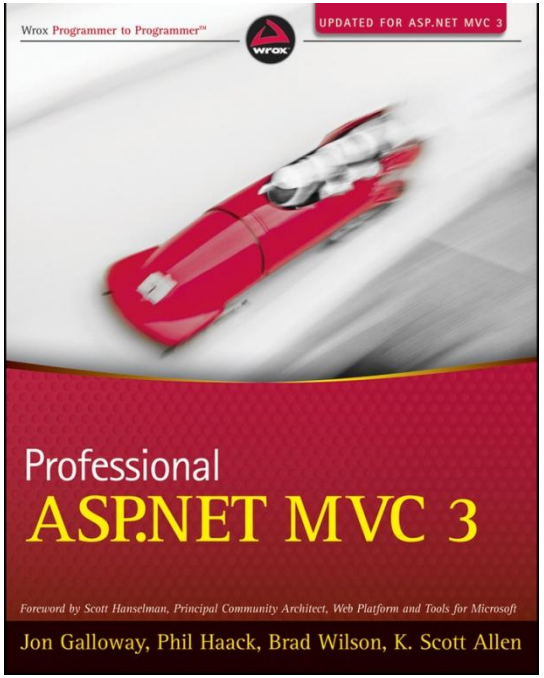
ASP.NET MVC – построение веб-приложений на базе шаблона MVC с разделением ответственности и полным контролем над HTML кодом страниц.

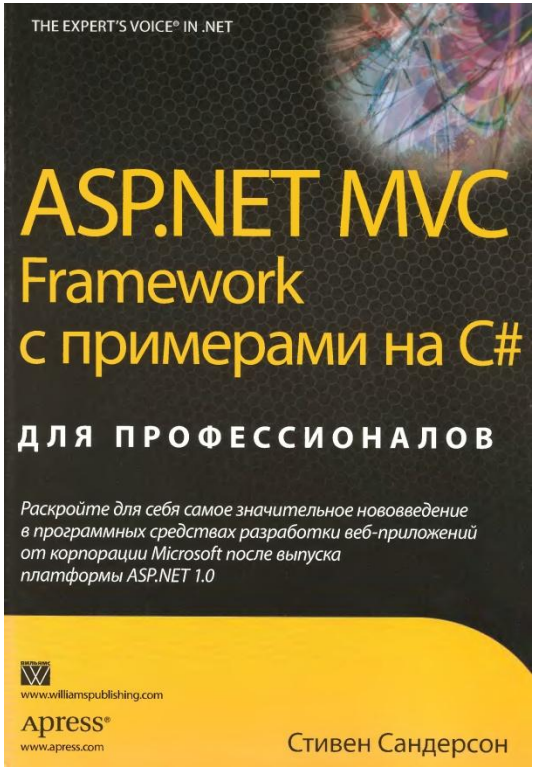
Web Pages – быстрая разработка веб-сайтов согласно современным веб-стандартам.

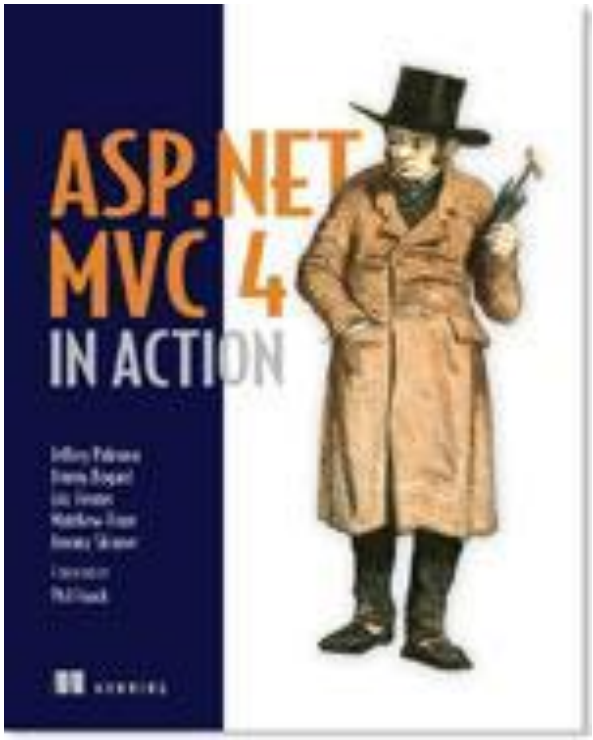
# План

1. Введение в ASP.NET MVC
2. Движок Razor
3. Модели
4. Доступ к данным
5. Контроллеры
6. Представления
7. Валидация ввода
8. Аутентификация и авторизация
9. Модульное тестирование
10. jQuery
11. AJAX

# Литература по ASP.NET MVC



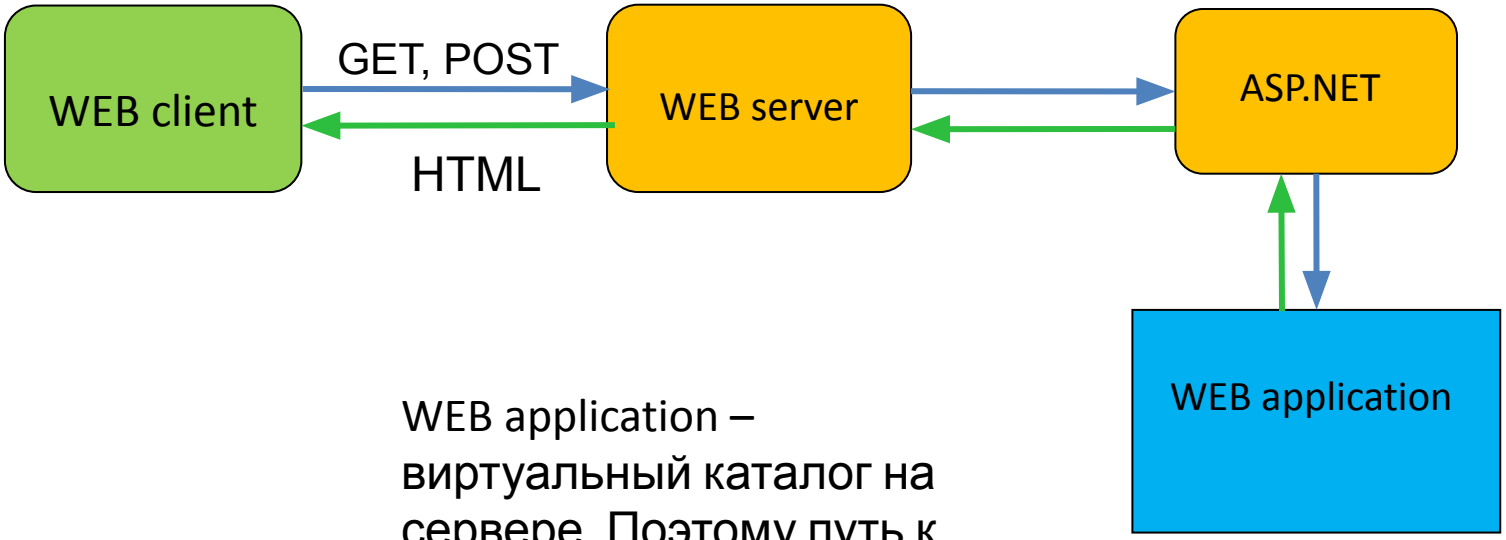




- Jon Galloway, Phil Haack, Brad Wilson, K. Scott Allen PROFESSIONAL ASP.NET MVC 3 (здесь перевод на русский)
- Стивен Сандерсон ASP.NET MVC Framework с примерами на C# для профессионалов
- Jeffrey Palermo, Jimmy Bogard, Eric Hexter, Matthew Hinze, and Jeremy Skinner ASP.NET MVC 4 in Action (есть на русском)

# Цель занятия

- Познакомиться с архитектурой WEB приложения.
- Вспомнить шаблон MVC.
- Написать приложение Hello ASP.NET MVC!

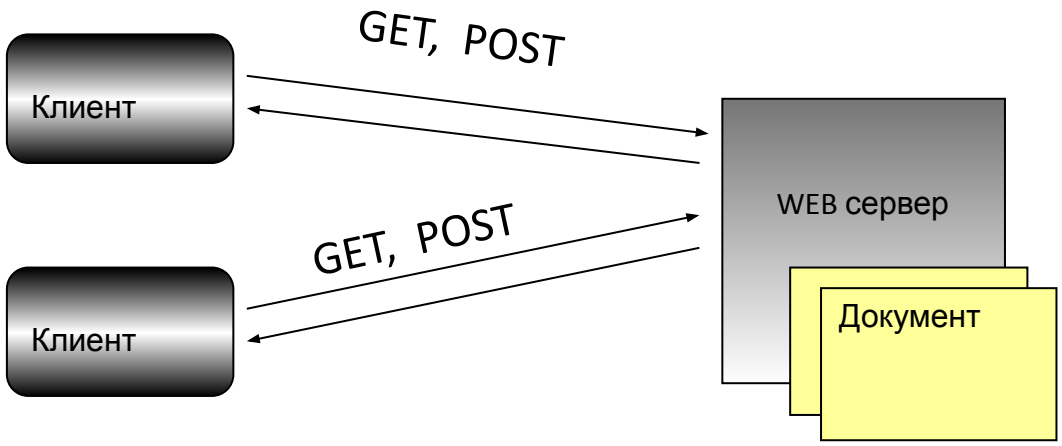
# WEB-приложение на платформе ASP.NET




WEB application – виртуальный каталог на сервере. Поэтому путь к ресурсу не обязан быть путем в файловой системе.

# Протокол HTTP

Тим Бернерс-Ли, изобретатель
URI, URL, HTTP, HTML и Web




HTTP — протокол прикладного уровня, аналогичными ему являются FTP и SMTP. Обмен сообщениями идёт по обыкновенной схеме «запрос-ответ». Для идентификации ресурсов HTTP использует глобальные URI. В отличие от многих других протоколов, HTTP не сохраняет своего состояния.

# Примеры запроса и ответа

*http://www.wintellect.com/simple.html*

```
GET /simple.html HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
If-Modified-Since: Wed, 24 Oct 2011 14:12:36 GMT
User-Agent: Mozilla/4.0.(compatible; MSIE.6.0; Windows NT 5.1)
Host: www.wintellect.com
Connection: Keep-Alive
[blank line]
```

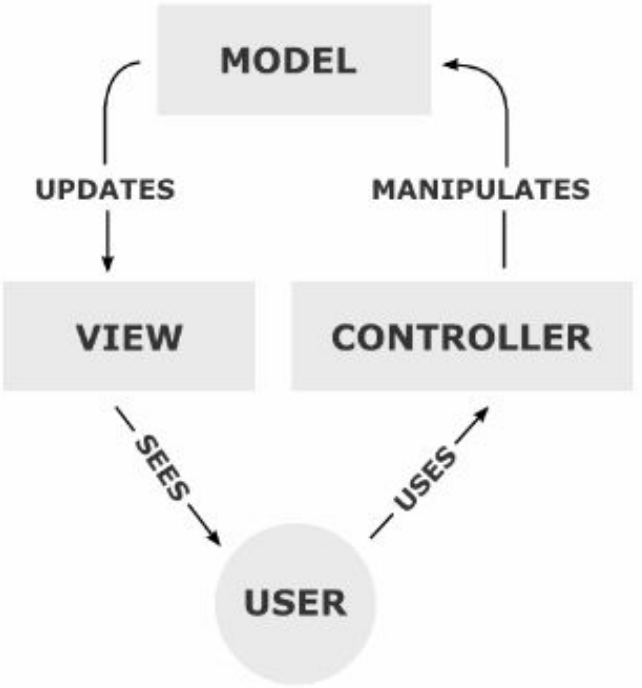
```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Wed, 24 Oct 2011 14:12:37 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Wed, 24 Oct 2001 14:00:53 GMT
Content-Length: 46

<html>
  <body>
    Hello, world
  </body>
</html>
```

Увидеть заголовки можно в окне разработчика браузера Chrome [F12, F5, Network, Headers].

# Шаблон MVC

Концепция MVC была описана в 1979 г. Трюгве Реенскаугом, тогда работающим над языком программирования Smalltalk в Xerox PARC.

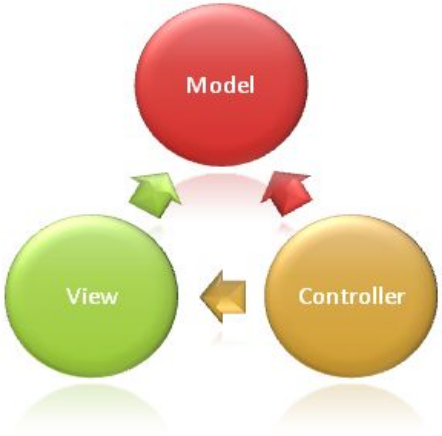



**Пассивная модель** — модель не имеет никаких способов воздействовать на представление или контроллер, и используется ими в качестве источника данных для отображения. Все изменения модели отслеживаются контроллером и он же отвечает за перерисовку представления, если это необходимо.

**Активная модель** — модель оповещает представление о том, что в ней произошли изменения, а представления, которые заинтересованы в оповещении, подписываются на эти сообщения. Это позволяет сохранить независимость модели как от контроллера, так и от представления.

Классической реализацией концепции MVC принято считать версию именно с активной моделью.

# Шаблон MVC для Web




**M** – классы, которые представляют данные приложения и бизнес-правила, которым должны удовлетворять эти данные

**V** – файлы шаблонов, по которым генерируется динамический HTML-ответ

**C** – классы, которые обрабатывают запросы пользователя, получают данные от модели и выбирают представление для формирования ответа пользователю

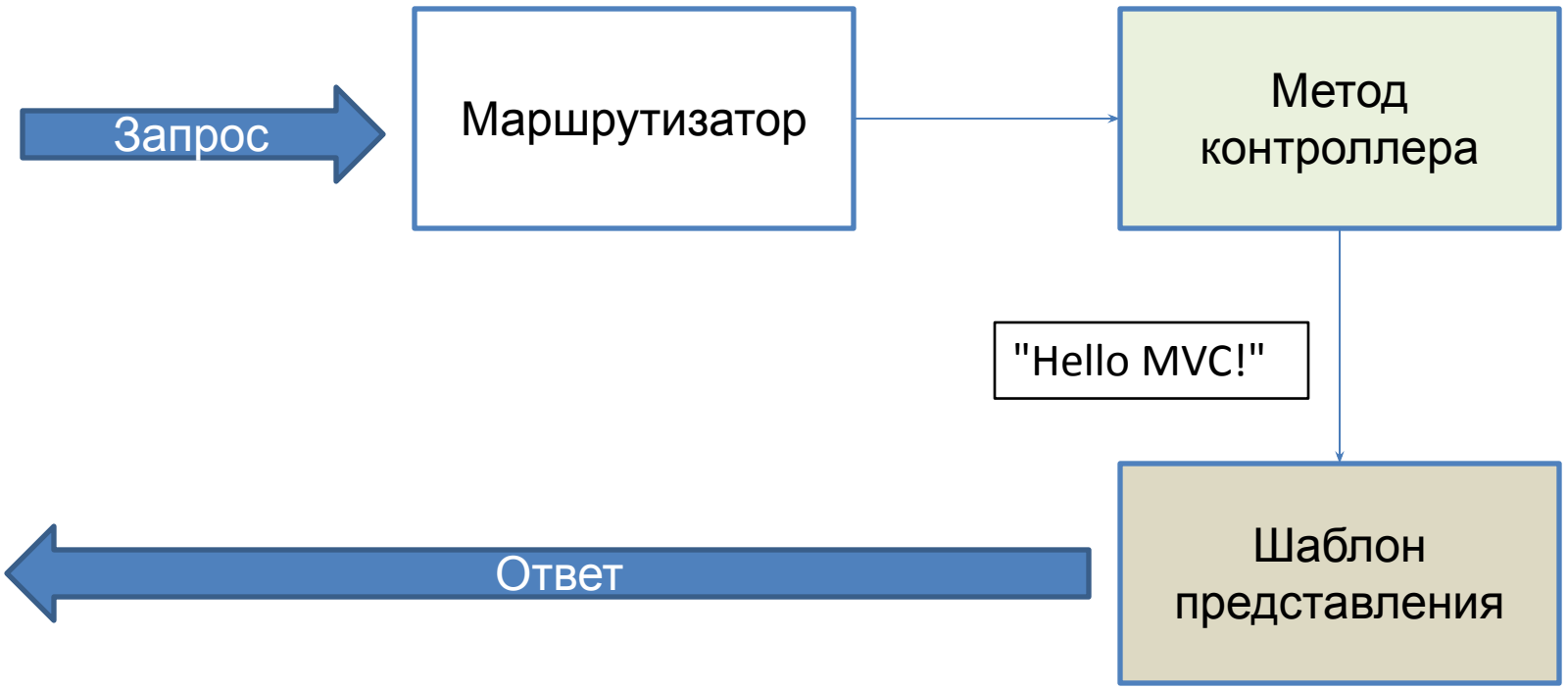
# Приложение Hello MVC!

1. Создать проект ASP.NET MVC 4, вид проекта – пустой (Empty).

2. Добавить HomeController, который передаст в представление слова "Hello MVC!"

3. Создать представление, которое получит от контроллера слова "Hello MVC!" и покажет их на странице.

# Траектория запроса



# Маршрутизатор

```
namespace MvcApplication4
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index",
                                id = UrlParameter.Optional }
            );
        }
    }
}
```

В файле ~/App_Start/RouteConfig.cs корректируется таблица маршрутов.

Эта коррекция означает, что запрос "сервер/приложение/C/M/" вызовет метод M класса C, а запрос "сервер/приложение/" вызовет метод Index класса Home.

Шаблон маршрута содержит текст «как есть» и параметры. Имена параметров заключены в фигурные скобки.

# Контроллер

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
    ViewBag.Info = "Hello MVC!";
        return View();
    }
}
```

В файле ~/Controllers/HomeController.cs находится класс контроллера.

Открытые методы контроллера вызываются по http-запросу.

Возвращаемое значение метода ссылается на шаблон представления. По умолчанию дается ссылка на представление, одноименное с методом.

Данные, которые нужно показать, закладываются в динамический объект `ViewBag`.

# Представление

```
<h2>@ViewBag.Info</h2>
```

В файле ~/Views/Home/Index.cshtml находится шаблон представления (Home – класс контроллера, Index – метод контроллера).

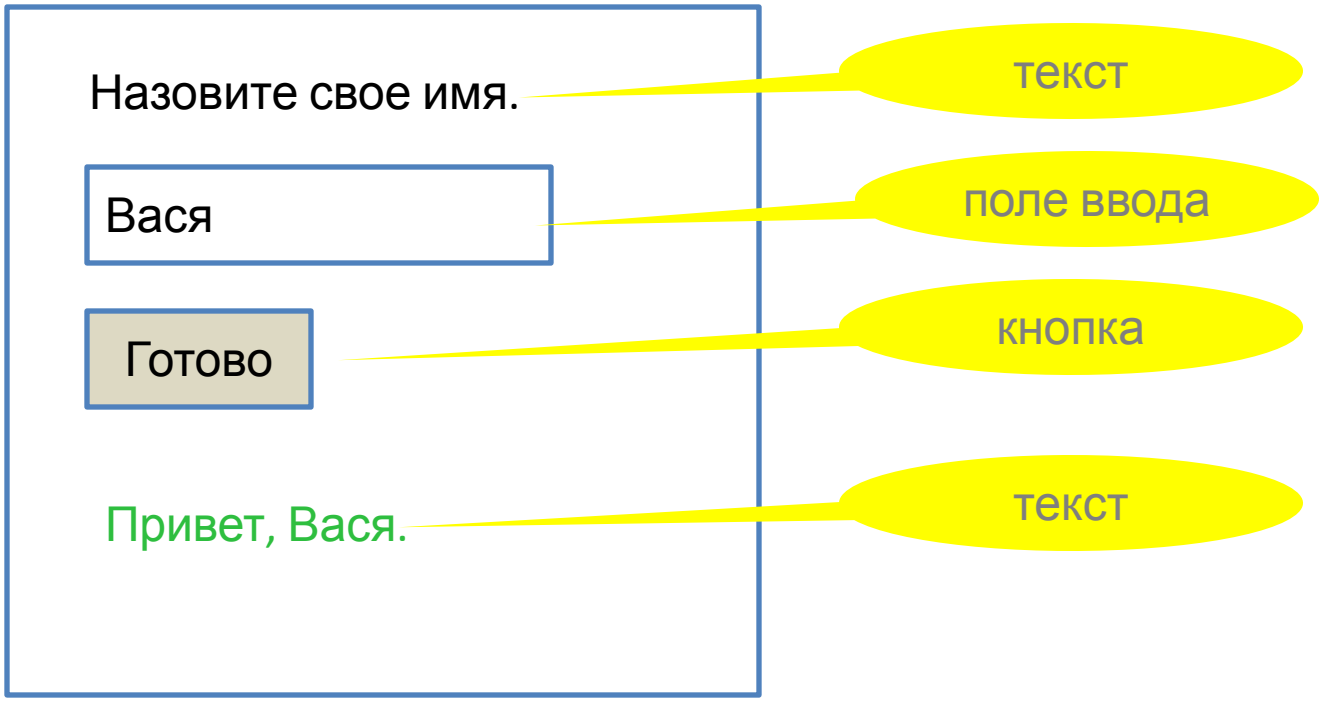
Данные, которые передал контроллер, извлекаются из динамического объекта `ViewBag`.

Данные вставляются в шаблон страницы с использованием нотации Razor.

# Реализовать диалог

**Сервер**: Назовите свое имя.
**Клиент**: Вася.
**Сервер**: Привет, Вася.

Вид веб-формы

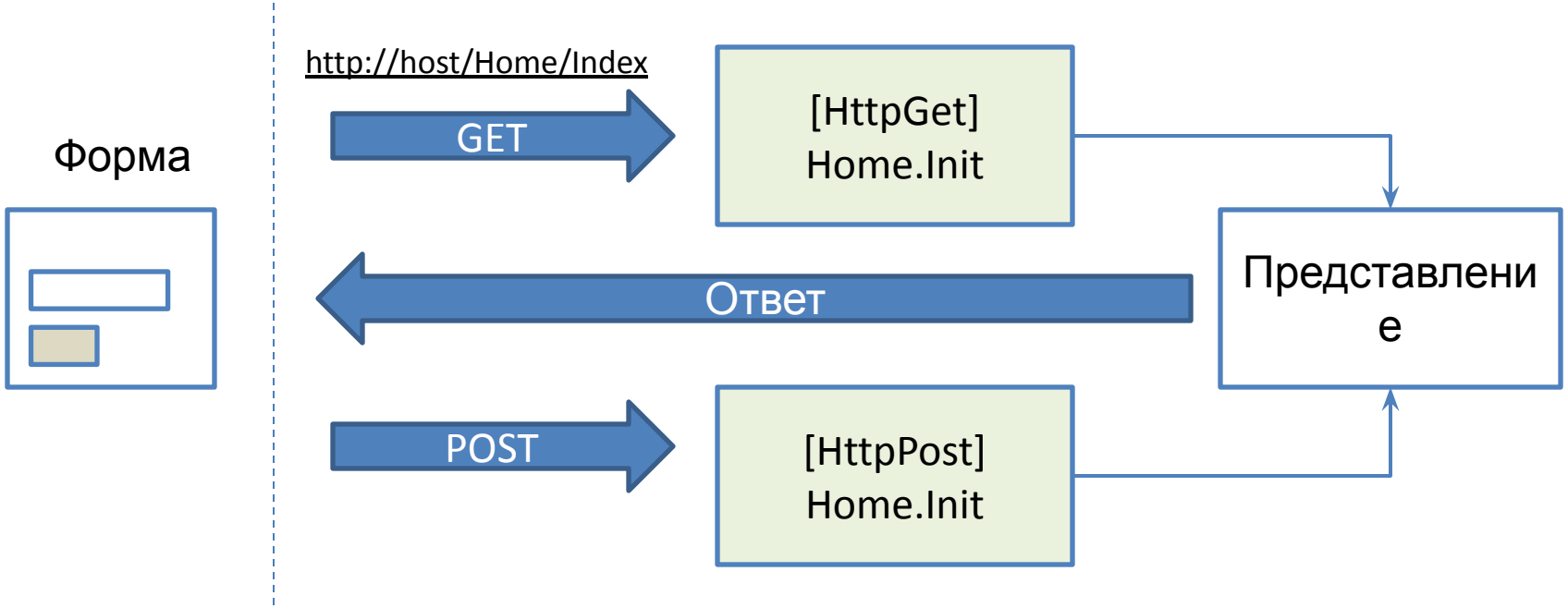


# Реализация диалога

Одна страница вызывается дважды
– первый раз из адресной строки браузера по команде GET,
– второй раз по нажатию кнопки, команда POST.

Запрос POST передает на сервер данные формы (содержимое поля ввода).

Методы контроллера нужно промаркировать атрибутом *HttpGet* или *HttpPost*.



# Извлечение параметров запроса

Команды GET и POST могут иметь именованные параметры.
Параметры команд должны быть переданы в соответствующие методы контроллера.

Есть два способа это сделать:

1) объявить одноименные параметры в соответствующем методе контроллера;

```
public ActionResult Index(string username = "")
{
      ViewBag.UserName = username;
      return View();
}
```

2) воспользоваться объектом RouteData, который содержит данные запроса в виде словаря.

```
public ActionResult Index()
{
     ViewBag.UserName = this.RouteData.Values["username"];
     return View();
}
```

# Самостоятельно

Сделать приложение, в котором пользователь может задать один из вопросов, «Который час?» или «Какой сегодня день недели?» и получить ответ от сервера.