



# Introduction to ASP.NET

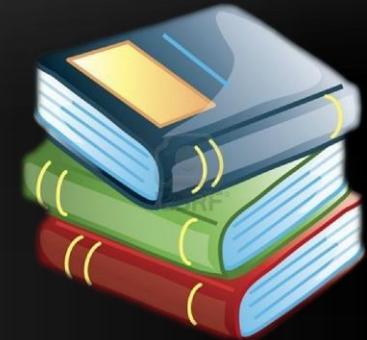
ASP.NET, Architecture, Web Forms, MVC, Web API

---

**ASP.NET Web Forms**  
Telerik Software Academy  
<http://academy.telerik.com>



- ◆ **Introduction to ASP.NET**
  - ◆ **History, Components, Frameworks**
- ◆ **ASP.NET App Structure**
  - ◆ **Typical Files and Folders in ASP.NET Projects**
- ◆ **ASP.NET App Lifecycle**
  - ◆ **Application Lifecycle, HTTP Modules, HTTP Handlers, Events, Controllers, Pages, ...**
- ◆ **ASP.NET Common Concepts**
  - ◆ **Classes & Namespaces, Web Sites & Web Apps**
- ◆ **ASP.NET vNext (5.0)**



The image shows two overlapping windows. The top window is a web browser displaying the official ASP.NET website. The bottom window is a Microsoft Visual Studio dialog for creating a new ASP.NET project.

**ASP.NET Website Content:**

- Header:** Application name, Home, About, Contact, Register, Log in
- Section 1:** ASP.NET logo, "ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS, and JavaScript.", "Learn more »"
- Section 2:** "Getting started" link, "ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.", "Learn more »"
- Section 3:** "Get more libraries" link, "NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.", "Learn more »"
- Section 4:** "Web Hosting" link, "You can easily find a web hosting company that offers the right mix of features and price for your applications.", "Learn more »"

**Visual Studio Project Dialog:**

- Title Bar:** New ASP.NET Project - IntroductionToAspNet
- Section 1:** "Select a template:"
  - Empty
  - Web Forms** (selected)
  - MVC
  - Web API
- Section 2:** "A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access." with a "Learn more" link.
- Section 3:** "Change Authentication" button.
- Section 4:** "Authentication: Individual User Accounts"
- Section 5:** "Microsoft Azure" dropdown with "Host in the cloud" checked, "Website" selected, and "Manage Subscriptions" link.
- Section 6:** "Add folders and core references for:"
  - Web Forms
  - MVC
  - Web API
- Section 7:** "Add unit tests" checkbox.
- Section 8:** "Test project name:" input field containing "IntroductionToAspNet.Tests".

# Introduction to ASP.NET

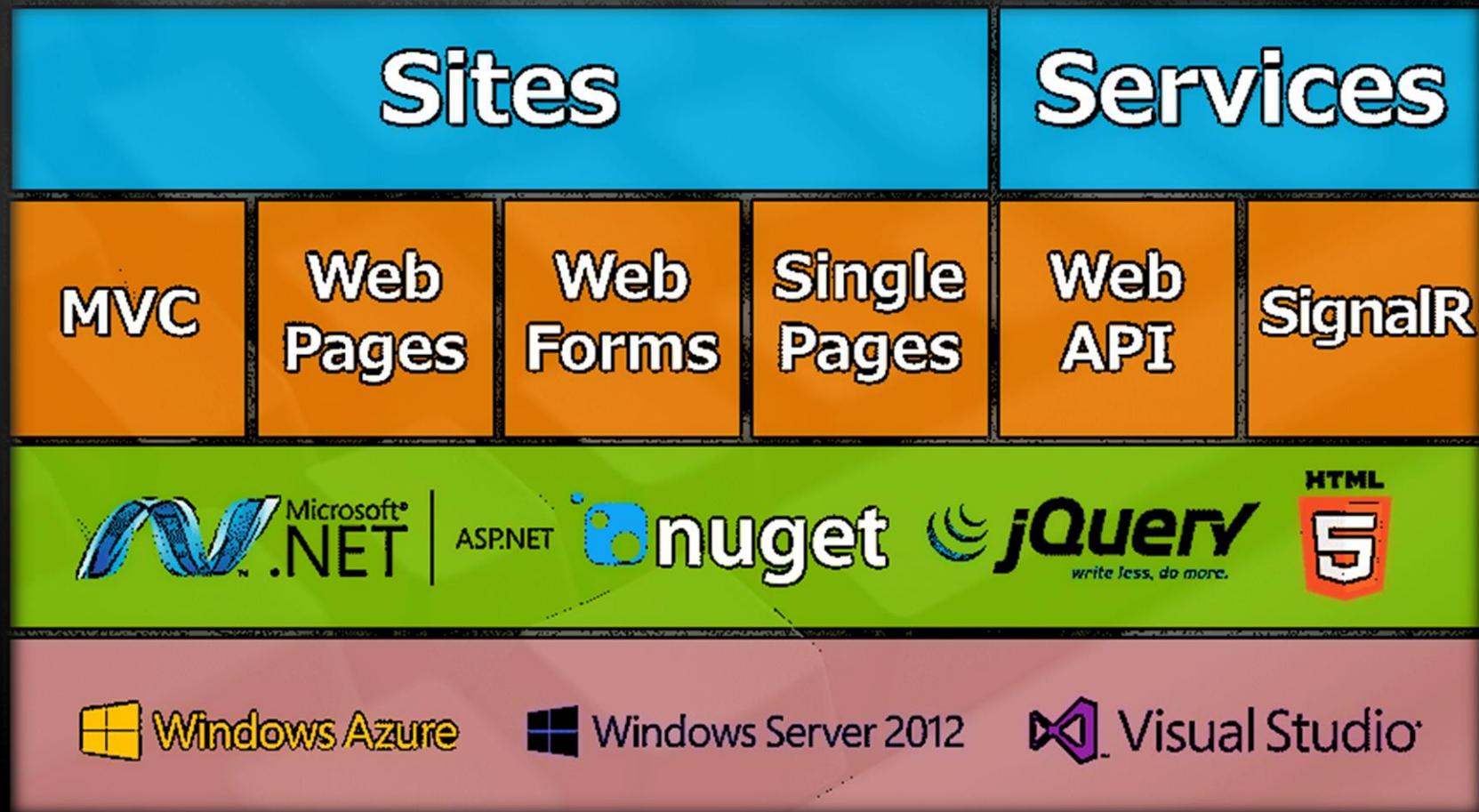
# History of ASP.NET

- ◆ At the beginning of Internet (up to 1997)
  - ◆ CGI, ISAPI (for C, C++), PHP
- ◆ Classic / Legacy ASP (1997-2002)
  - ◆ Based on VB Script, COM, ADO
- ◆ ASP.NET 1.0 (2002, January 16) – with .NET 1.0
- ◆ ASP.NET 1.1 (2003-2005) – based on .NET 1.1
- ◆ ASP.NET 2.0 (2005-2007) – based on .NET 2.0
- ◆ ASP.NET 3.5 (2007-2009) – LINQ to SQL, MVC
- ◆ ASP.NET 4.0 (2010) – Entity Framework, MVC
- ◆ ASP.NET 4.5 (2012) – One ASP.NET ([info](#))
- ◆ ASP.NET vNext (2014) – Redesigned ([info](#))



# What is ASP.NET?

- ◆ **ASP.NET** is a stack of technologies to create web sites, web services and web applications



# ASP.NET: Web Forms vs. MVC

- ◆ ASP.NET has two major frameworks for Web application development
  - ◆ ASP.NET Web Forms ([read more](#))
    - ◆ The traditional component-based approach
    - ◆ Mixes the presentation and presentation logic
  - ◆ ASP.NET MVC ([read more](#))
    - ◆ Modern approach, more clear and flexible
    - ◆ MVC architecture, like Ruby-on-Rails and Django
    - ◆ Testable (test the controllers)

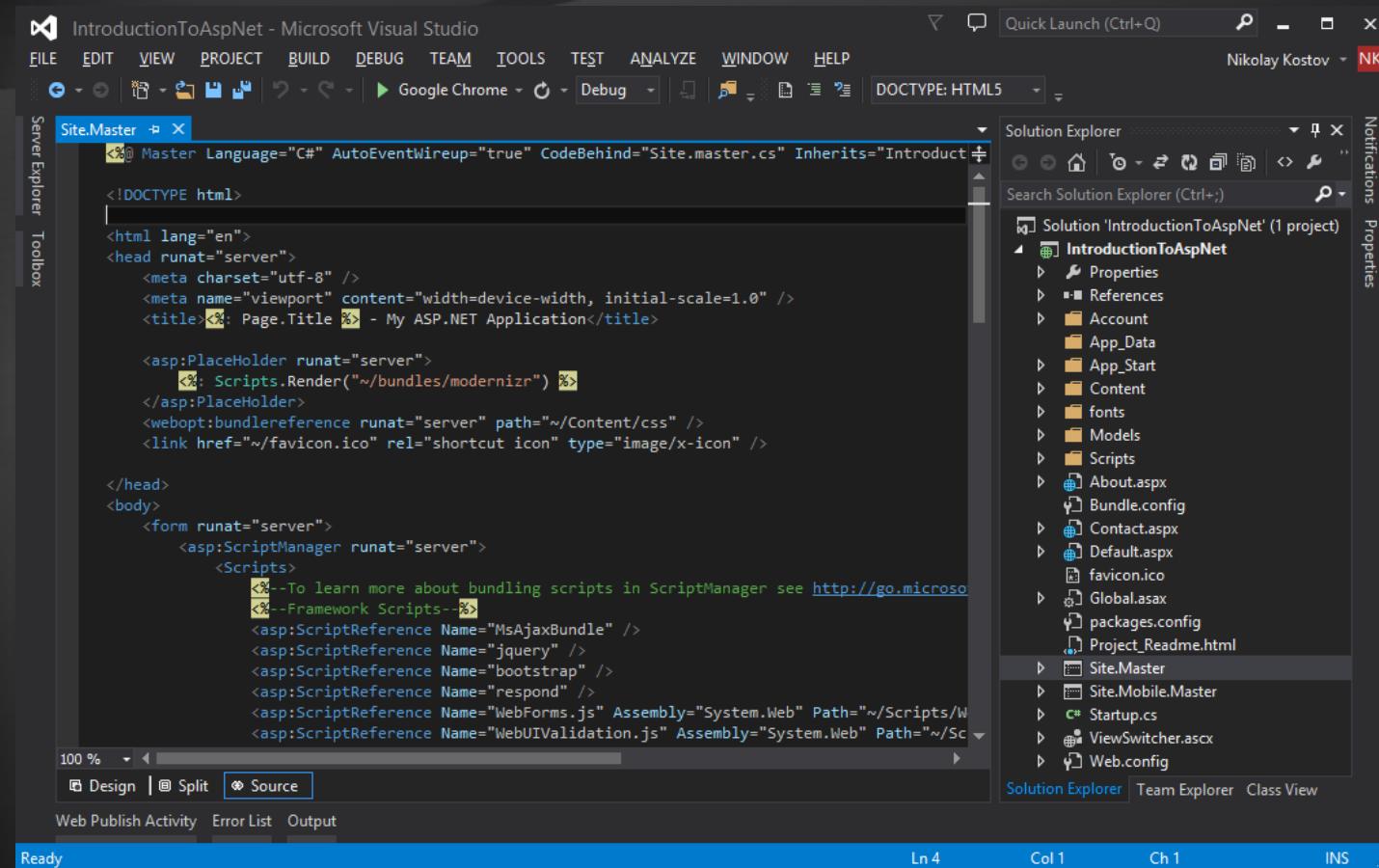
- ◆ **ASP.NET Web Pages ([read more](#))**

- ◆ **Lightweight framework to combine server code with HTML to create dynamic web content**
  - ◆ Like PHP: mix HTML code with C# code
  - ◆ Uses the "Razor" templating engine

- ◆ **ASP.NET Web API ([read more](#))**

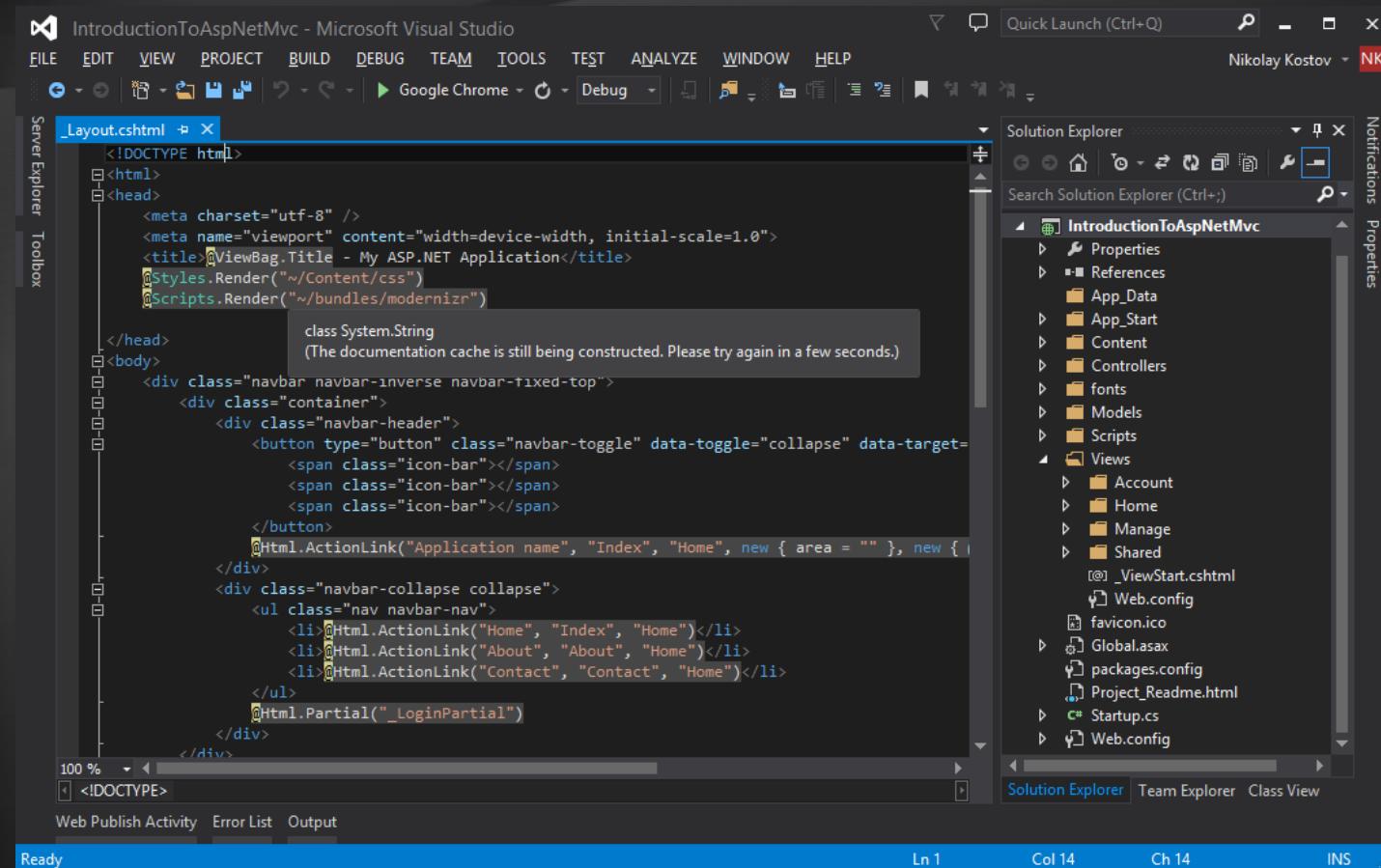
- ◆ **Framework for building RESTful Web services**
  - ◆ Write C# code to handle HTTP requests in REST style (GET / POST / PUT / DELETE requests)
  - ◆ Return JSON / XML as result

- ◆ Single Page Applications (SPA) ([read more](#))
  - Combine Web API with client-side JS
  - Write a HTML5 single page apps with jQuery / Knockout.js / other JS client-side framework
  - Client HTML5 code consumes Web API services
- ◆ SignalR ([read more](#))
  - Real-time communication between client (JS) and server (C#) over HTTP through Web Sockets
    - Server C# code can invoke JS functions at the client
    - Client JS code can invoke C# methods at the server



# Simple Web Forms App

## Live Demo

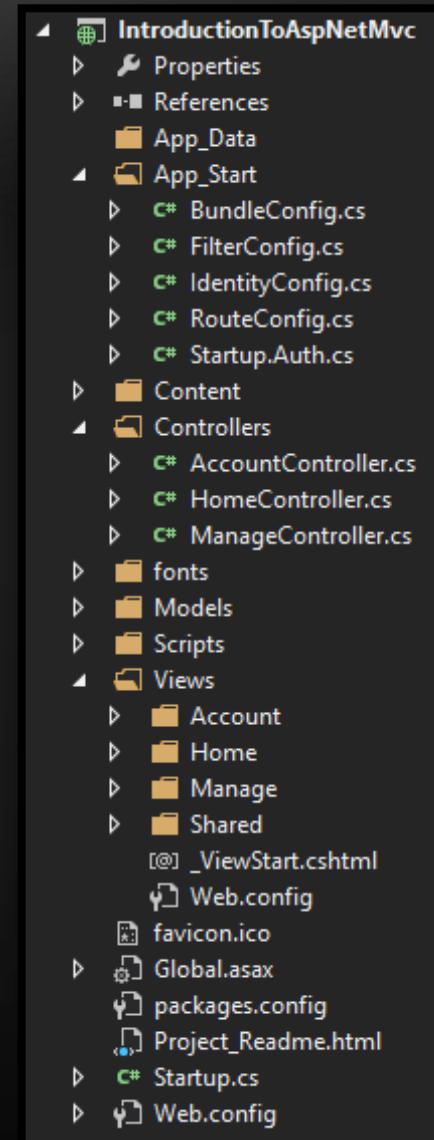
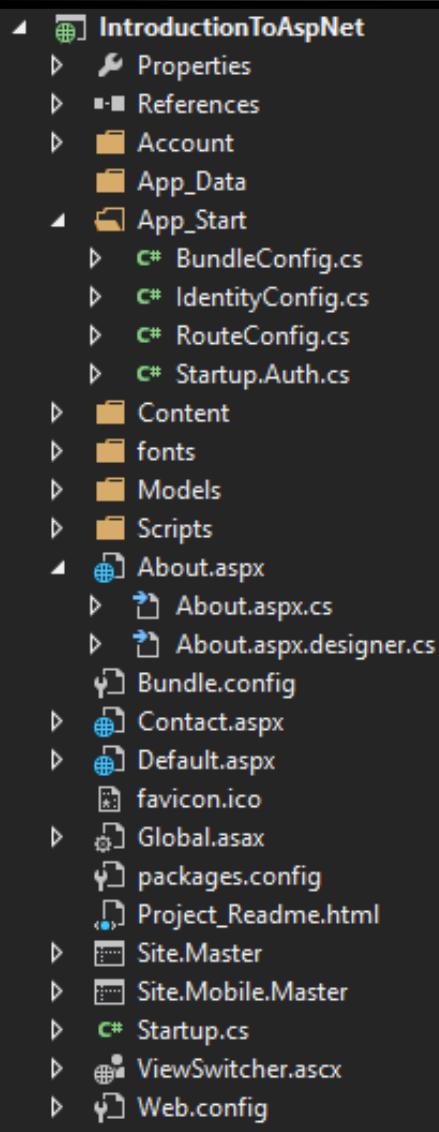


# Simple MVC App

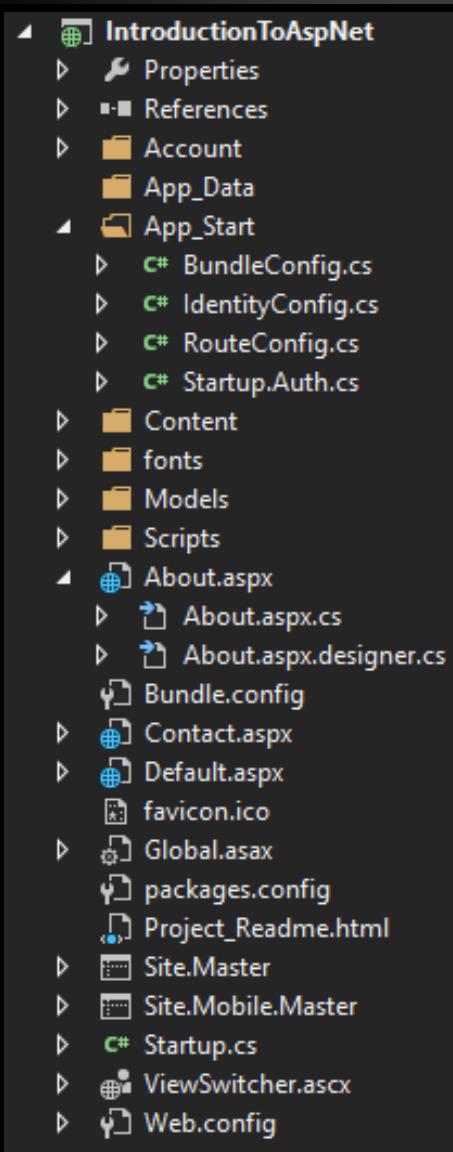
Live Demo

# ASP.NET App Structure

## Typical Application Structure in ASP.NET



# ASP.NET App Structure



- ◆ App\_Start
  - ◆ BundleConfig / RoutesConfig / IdentityConfig / Startup.cs
- ◆ App\_Data
- ◆ Web.config
- ◆ Global.asax
- ◆ Content and Content\themes
- ◆ Scripts, img, fonts
- ◆ Models / Views / Controllers
- ◆ Site.Master / Site.Mobile.Master

- ◆ The App\_Start folder
  - ◆ Holds global configuration logic
  - ◆ Classes that are loaded at application start-up
- ◆ BundleConfig.cs ([read more](#))
  - ◆ Combines and optimizes CSS and JS files
- ◆ FilterConfig.cs ([read more](#))
  - ◆ Configures filters in MVC / Web API apps
  - ◆ Configures pre-action and post-action behavior to the controller's action methods

- ◆ **RouteConfig.cs ([read more](#))**
  - ◆ Configures URL patterns and their handlers
  - ◆ Maps user-friendly URLs to certain page / controller
- ◆ **IdentityConfig.cs / Startup.Auth.cs**
  - ◆ Configures the membership authentication
    - ◆ Users, roles, login, logout, user management
  - ◆ OAuth login (cross-sites login, [read more](#))
    - ◆ Facebook / Twitter / Microsoft / Google login

- ◆ The App\_Data directory holds the local data files of the Web application
  - ◆ E.g. MyWebApp.mdf + MyWebApp.ldf
  - ◆ E.g. Countries.xml
- ◆ The SQL Server "Local DB" ([read more](#))
  - ◆ Local .mdf + .ldb files, attached at startup
  - ◆ SQL Server process started on demand
  - ◆ Database created on demand (if missing)
  - ◆ Great for development and testing

- ◆ **Web.config** is web app's configuration file
  - Holds settings like DB connection strings, HTTP handlers, modules, assembly bindings
  - Can hold custom application settings, e.g. credentials for external services
  - Changes in **Web.config** do not require rebuild
- ◆ You may have several **Web.config** files
  - One global for the application
  - Several for different folder in the application

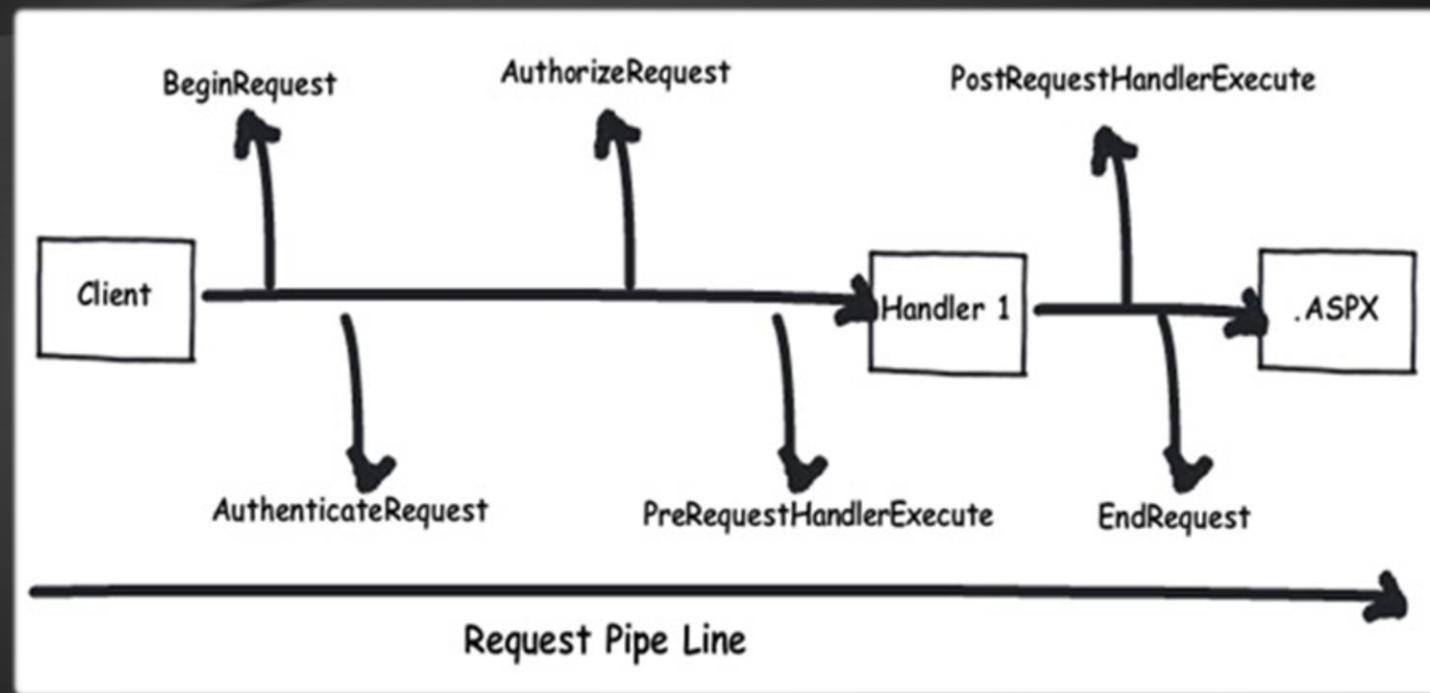
- ◆ Web.config inherits from the global Web.config and from machine.config
  - ◆ Global settings for all applications on the server

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\Config\machine.config
```

- ◆ Web.Debug.config
  - ◆ Local settings for debugging
  - ◆ E.g. local database instance for testing
- ◆ Web.Release.config
  - ◆ Production settings for real world deployment

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <section name="entityFramework"
      type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework,
      Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
      requirePermission="false" />
  </configSections>
  <connectionStrings>
    <add name="DefaultConnection" connectionString="Data
      Source=(LocalDb)\v11.0;AttachDbFilename=|DataDirectory|\aspnet.mdf;Initial
      Catalog=aspnet;Integrated Security=True" providerName="System.Data.SqlClient" />
  </connectionStrings>
  <appSettings>
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
    ...
  </appSettings>
  <system.web>
    <compilation debug="true" targetFramework="4.5" />
    ...
  </system.web>
  <system.webServer> ... </system.webServer>
  <runtime> ... </runtime>
  <entityFramework> ... </entityFramework>
</configuration>
```

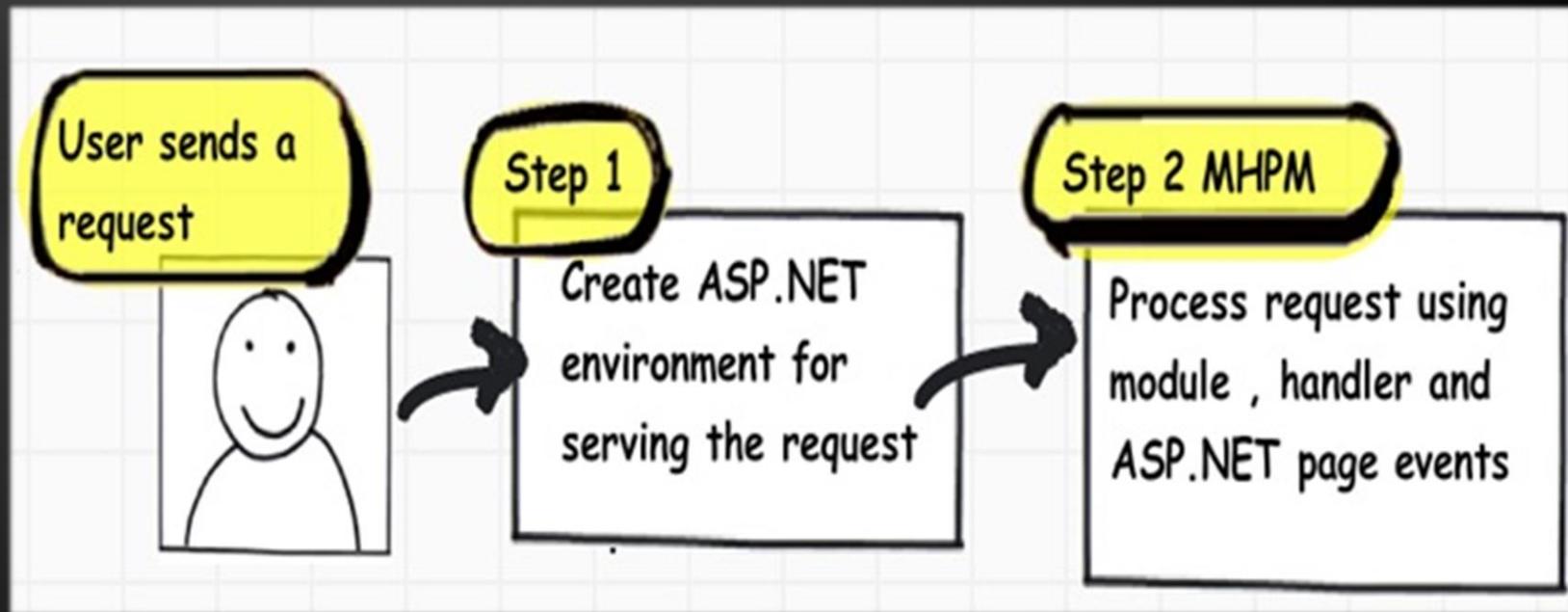
- ◆ **Global.asax** defines the HTTP application
  - ◆ Defines global application events like
    - ◆ Application\_Start
    - ◆ Application\_BeginRequest
    - ◆ Application\_EndRequest
    - ◆ Application\_Error
    - ◆ ...
  - ◆ Typically invokes BundleConfig,  
RouteConfig, FilterConfig, etc.



# ASP.NET App Lifecycle

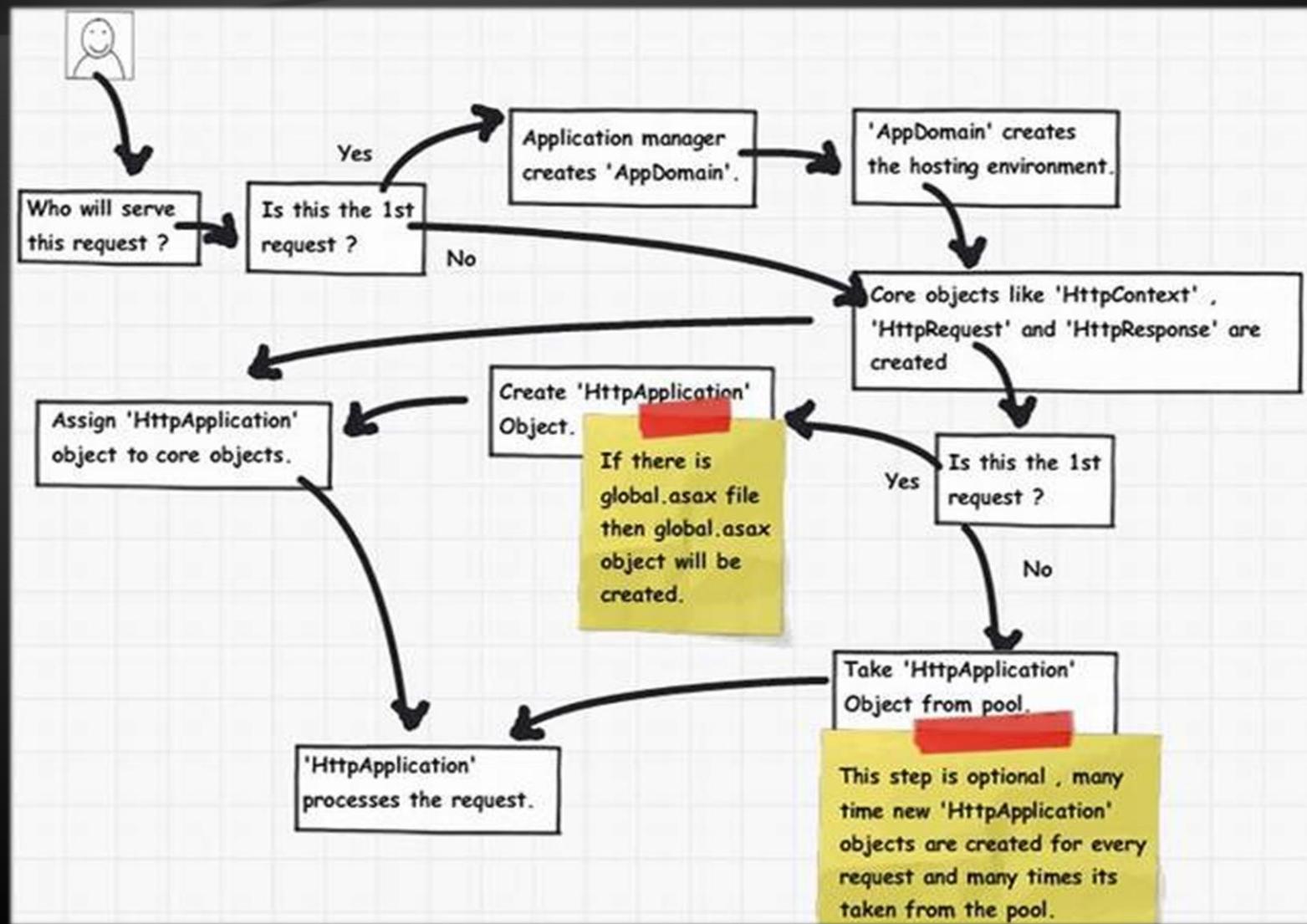
Application Lifecycle, HTTP Modules,  
Handlers, Events, Controllers, Pages, ...

# ASP.NET App Lifecycle

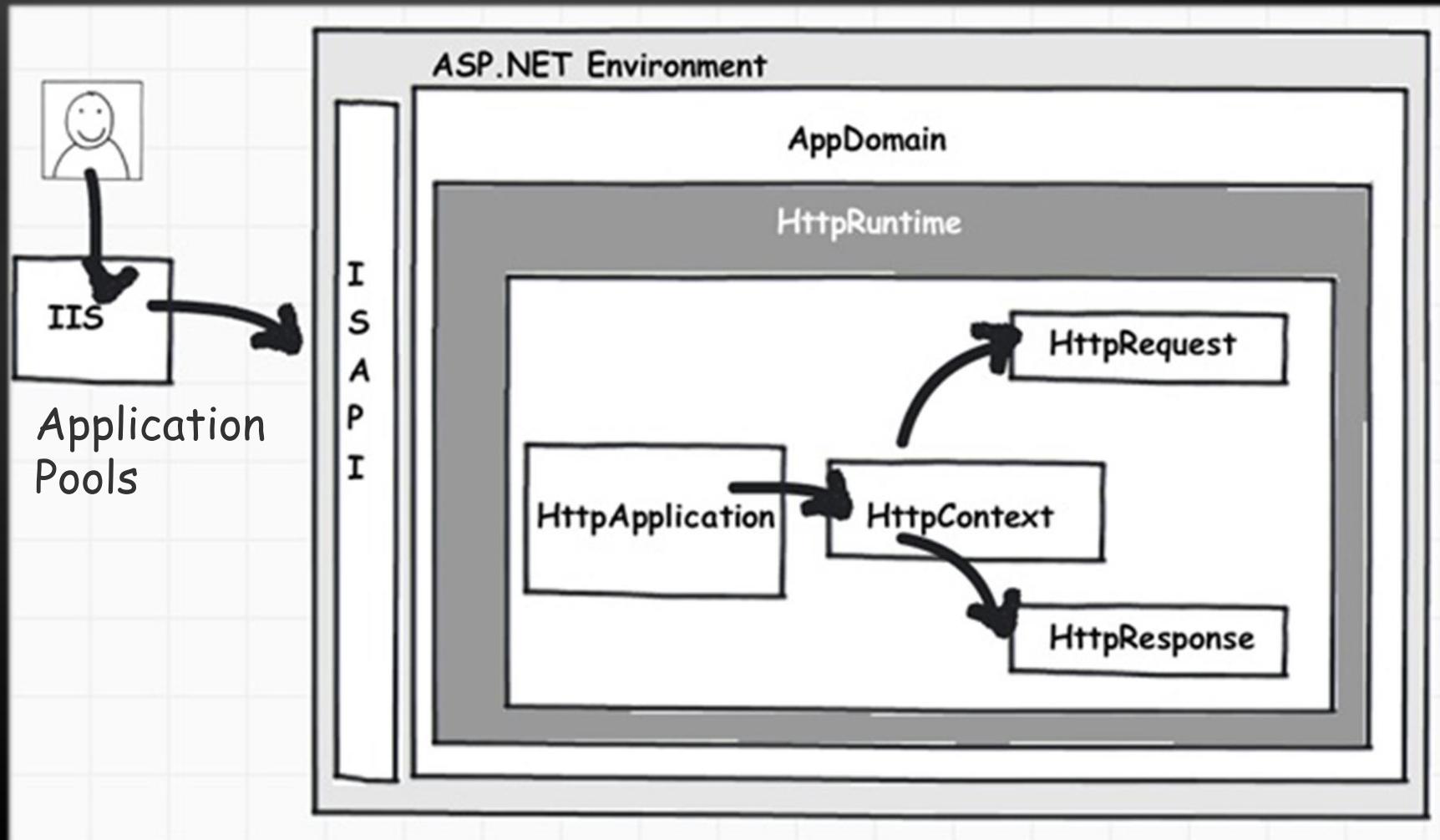


- ◆ **MHPM == Module, Handler, Page Events, Module Events**

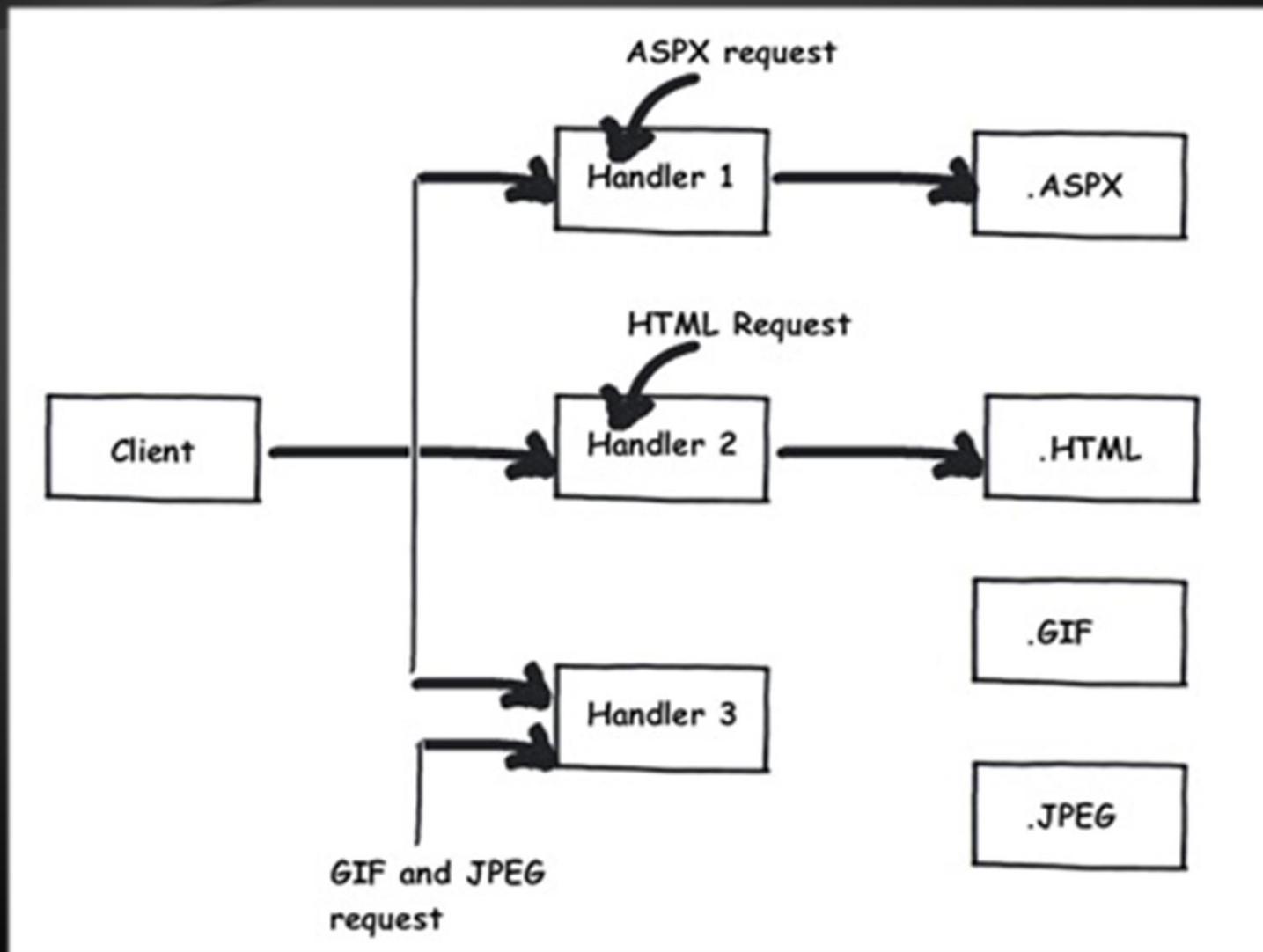
# ASP.NET App Lifecycle (2)



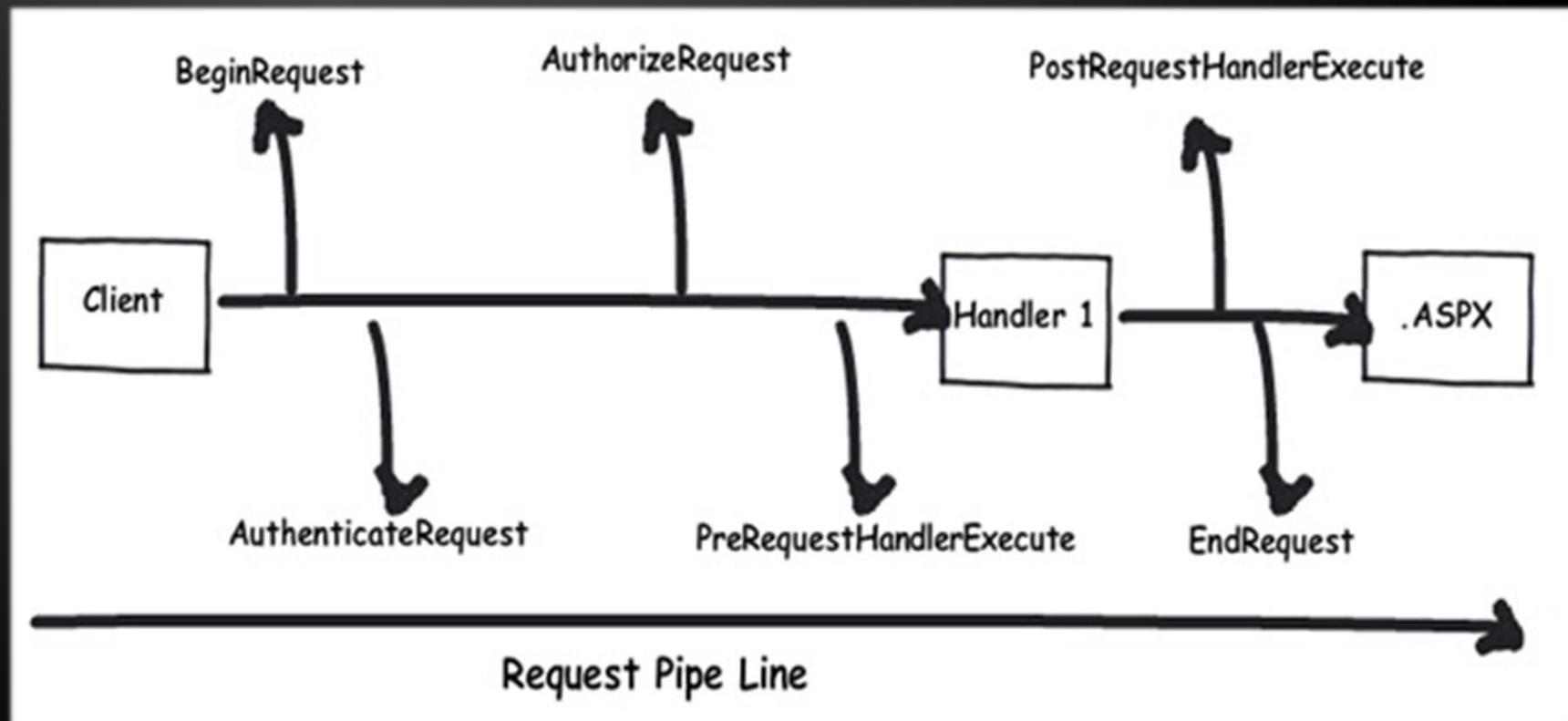
# ASP.NET App Lifecycle (3)



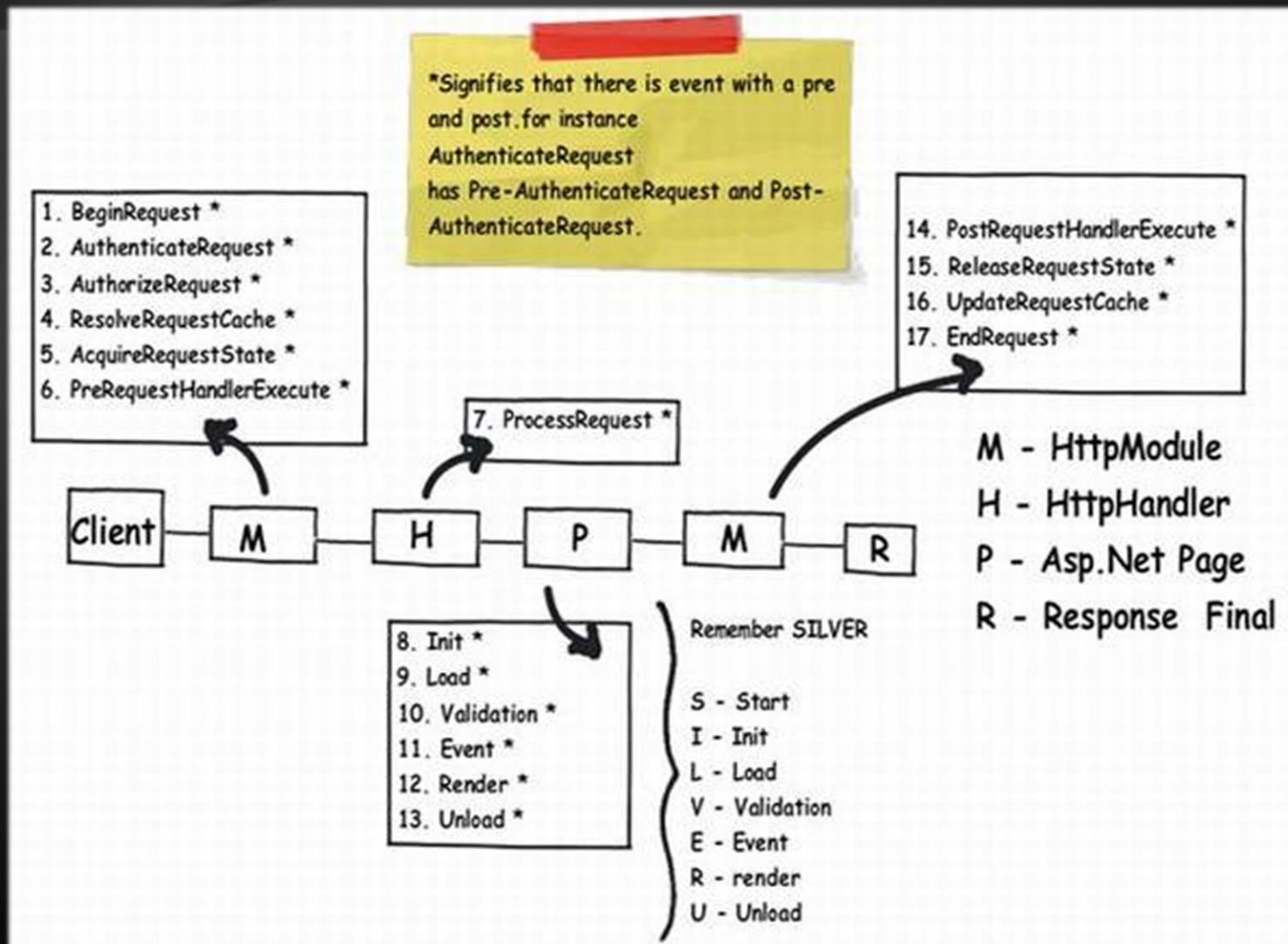
# ASP.NET App Lifecycle (4)



# ASP.NET App Lifecycle (5)

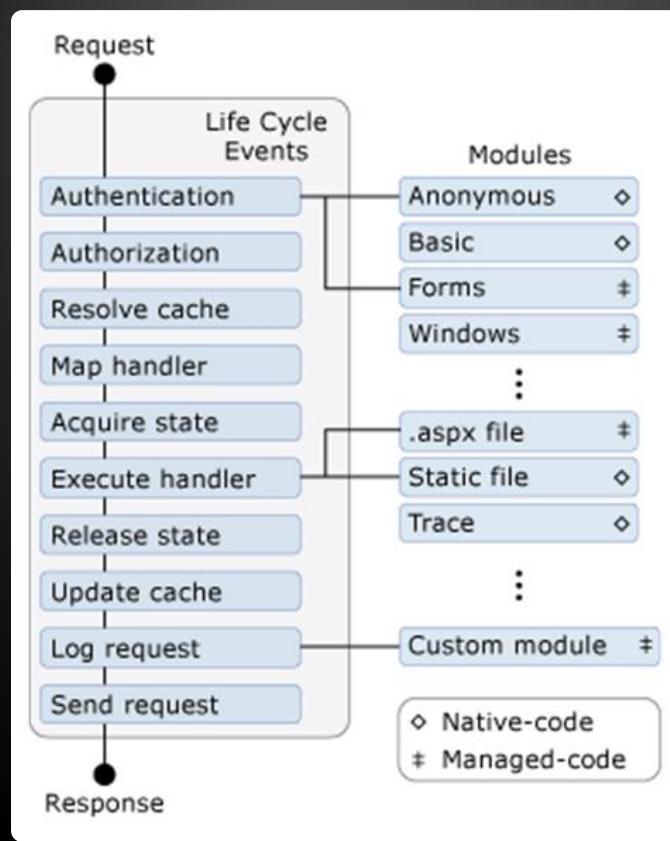


# ASP.NET App Lifecycle (6)



# Application Lifecycle Events

- ◆ **HttpApplication** have a complex pipeline to process HTTP requests ([read more](#))



- ◆ **BeginRequest**
- ◆ **AuthenticateRequest**
- ◆ **AuthorizeRequest**
- ◆ **AcquireRequestState**
- ◆ **ProcessRequest**
- ◆ **ReleaseRequestState**
- ◆ **EndRequest**

ASP.NET-Demos - Microsoft Visual Studio

```
Global.asax.cs # X
ASP.NET_App_Lifecycle_Events.Global Application_Start(object sender, EventArgs e)
using System;
using System.Diagnostics;

namespace ASP.NET_App_Lifecycle_Events
{
    public class Global : System.Web.HttpApplication
    {
        protected void Application_Start(object sender, EventArgs e)
        {
            Trace.WriteLine("Application_Start Called.");
        }

        protected void Session_Start(object sender, EventArgs e)
        {
            Trace.WriteLine("Session_Start Called.");
        }
    }
}
```

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST Svetlin Nakov

ARCHITECTURE ANALYZE WINDOW HELP

Toolbox Server Explorer

100 % Error List Data Tools Operations Output Find Symbol Results Package Manager Console

Ready Ln 1 Col 1 Ch 1 INS

ASP.NET-Demos... Quick Launch (Ctrl+Q)

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SC

TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP

Toolbox Server Explorer

WebAppLog.log\* # X Global.asax.cs

```
Application_Start Called.
Application_BeginRequest Called.
Application_AuthenticateRequest Called.
Session_Start Called.
Application_EndRequest Called.
Application_BeginRequest Called.
Application_AuthenticateRequest Called.
Application_EndRequest Called.
Session_End Called.
Application_End Called.
```

100 % Error List Data Tools Operations Output Find Symbol Results

Ready Ln 11 Col 1 Ch 1

# App Lifecycle Events

Live Demo

- ◆ A "HTTP handler" is a process / C# code that responses to HTTP requests
- ◆ Sample HTTP handler in C#:

```
public class TelerikAcademyHttpHandler : IHttpHandler
{
    public void ProcessRequest(HttpContext context)
    { context.Response.Write("I am a HTTP handler."); }

    public bool IsReusable
    { get { return false; } }
}
```

- ◆ Handler registration in Web.config:

```
<configuration><system.webServer><handlers>
    <add verb="*" path="*.academy" name="Academy's HTTP handler"
        type="TelerikAcademyHttpHandler"/>
</handlers></system.webServer></configuration>
```

The screenshot shows the Microsoft Visual Studio interface with the following details:

- Title Bar:** ASP.NET-Demos - Microsoft Visual Studio
- Menu Bar:** FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
- User Name:** Svetlin Nakov
- Toolbox:** Available on the left side.
- Solution Explorer:** Available on the right side.
- Code Editor (NakovHttpHandler.cs):**

```
public class NakovHttpHandler : IHttpHandler
{
    /// <summary>
    /// This handler is called whenever a file ending in .nakov is requested. A file with that extension does not need to exist.
    /// </summary>
    public void ProcessRequest(HttpContext context)
    {
        HttpResponse response = context.Response;
        response.ContentType = "text/plain";
        response.Write("I am Nakov's HTTP handler.\r\n");
        response.Write("Response date: " + DateTime.Now);
    }

    public bool IsReusable
    {
        // Return true to keep the handler in memory (pooling)
        get { return false; }
    }
}
```
- Web.config:**

```
<?xml version="1.0" encoding="utf-8"?>

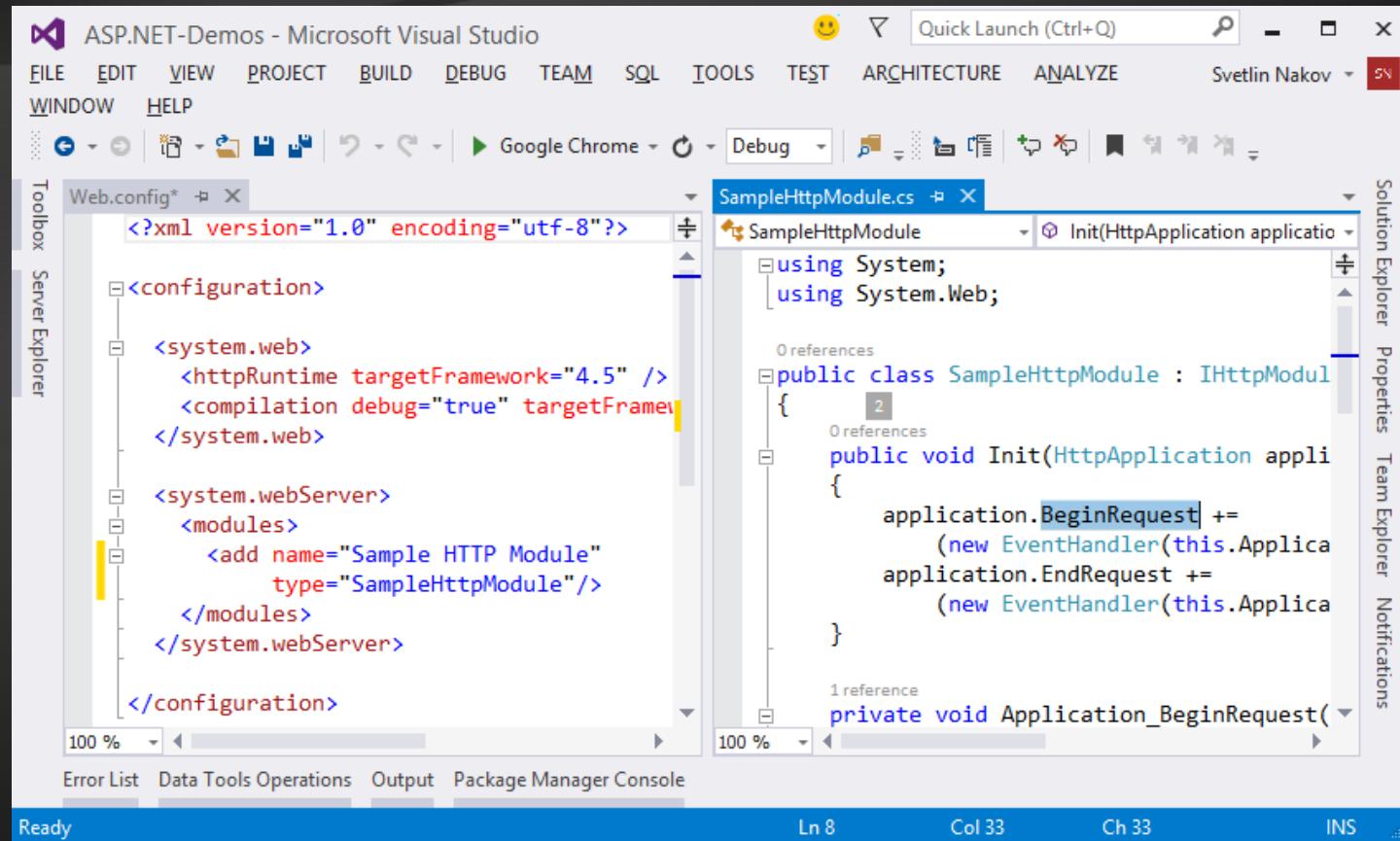
<configuration>
    <system.web>
        <httpRuntime targetFramework="4.5" />
        <compilation debug="true" targetFramework="4.5" />
    </system.web>

    <system.webServer>
        <handlers>
            <add verb="*" path="*.nakov"
                 name="Nakov's HTTP Handler" type="NakovHttpHandler" />
        </handlers>
    </system.webServer>
</configuration>
```
- Status Bar:** Item(s) Saved, Ln 13, Col 9, Ch 9, INS

# Writing a HTTP Handler

## Live Demo

- ◆ HTTP modules can customize requests for resources that are serviced by ASP.NET
  - ◆ It can intercept all HTTP requests and apply a custom logic
- ◆ Steps to create an HTTP Module
  - ◆ Implement the `IHttpModule` interface
    - ◆ Subscribe to events you want to intercept, e.g. `HttpApplication.BeginRequest`
  - ◆ Register the HTTP module in `Web.config` in the `<modules>` section



# Writing a HTTP Module

## Live Demo



- ▷ System.Web Namespaces
- ↳ **System.Web**
  - ApplicationShutdownReason Enumeration
  - ▷ AspNetHostingPermission Class
  - ▷ AspNetHostingPermissionAttribute Class
  - AspNetHostingPermissionLevel Enumeration
  - BeginEventHandler Delegate
  - DefaultHttpHandler Class
  - EndEventHandler Delegate
  - EventHandlerTaskAsyncHelper Class
  - HtmlString Class
  - HttpApplication Class
  - HttpApplication State Class
  - HttpApplicationStateBase Class
  - HttpApplicationStateWrapper Class
  - HttpBrowserCapabilities Class
  - HttpBrowserCapabilitiesBase Class
  - HttpBrowserCapabilitiesWrapper Class

# ASP.NET Common Concepts

## Major Classes, Namespaces, Web Sites, Web Apps

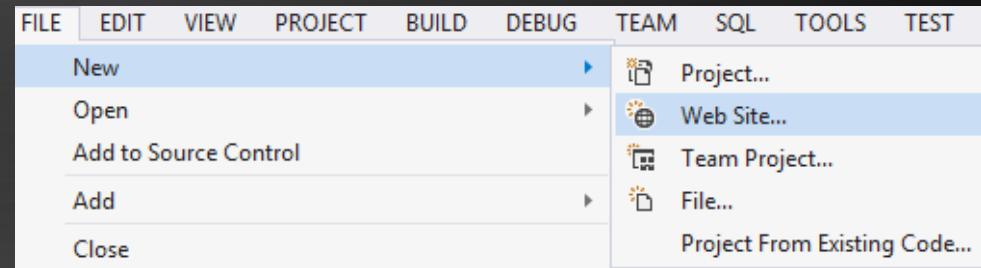
- ◆ Major ASP.NET (4-5) namespaces
  - ◆ System.Web
    - ◆ Web application main classes like **HttpApplication**, **HttpContext**, **HttpRequest**, **HttpResponse**, **HttpSessionState**, ...
  - ◆ System.Web.Mvc
    - ◆ MVC classes and framework components
  - ◆ System.Web.UI
    - ◆ Web Forms UI controls (like **Button** and **Label**)

- ◆ **HttpApplication**
  - ◆ Base class for the ASP.NET Web apps  
(inherited in Global.asax)
- ◆ **HttpContext**
  - ◆ Encapsulates all HTTP-specific information about an individual HTTP request
- ◆ **HttpRequest**
  - ◆ Encapsulates an HTTP request
- ◆ **HttpResponse**
  - ◆ Encapsulates an HTTP response

# Web Site vs. Web Application

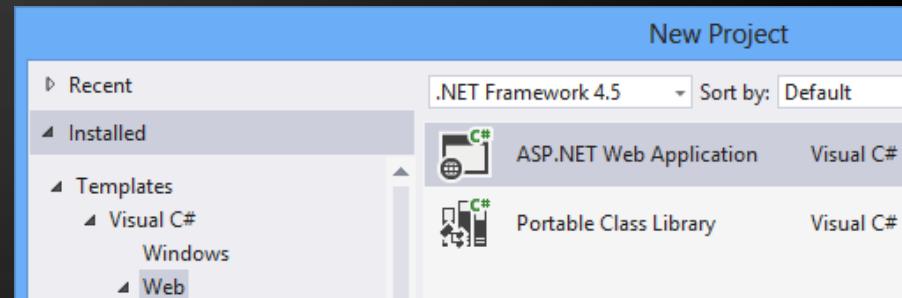
## ◆ Web Sites in VS

- ◆ No project file (.csproj / .sln)
- ◆ Code compiled dynamically at the Web server
- ◆ Can be precompiled (into multiple assemblies)



## ◆ Web Apps in VS

- ◆ Have project file (like any C# project)
- ◆ Compilation produces an assembly: bin\\*.dll
- ◆ Web apps are recommended ([read more](#))



The screenshot shows the Microsoft Visual Studio interface with the following details:

- Title Bar:** WebApplication1 - Microsoft Visual Studio
- Menu Bar:** File, Edit, View, Project, Build, Debug, Team, Tools, Test, Analyze, Window, Help
- Toolbar:** Standard icons for file operations.
- Solution Explorer:** Shows the project structure:
  - Solution 'WebApplication1' (1 project)
    - Solution Items
      - global.json
    - src
      - WebApplication1
        - wwwroot
          - bin
          - Content
          - Scripts
        - References
          - ASP.NET 5.0
        - Controllers
          - AccountController.cs
          - HomeController.cs
        - Models
          - AccountViewModels.cs
          - IdentityModels.cs
        - Scripts
          - \_references.js
        - Views
          - Account
          - Home
          - Shared
            - \_ViewStart.cshtml
        - config.json
        - project.json
        - Project\_Readme.html
        - Startup.cs
  - Properties:** Shows the properties for the selected item.
  - Toolbox:** Standard .NET development tools.
  - Code Editors:** Two code editors are open:
    - project.json**:

```
{  
  "webroot": "wwwroot",  
  "exclude": "wwwroot/**/*.*",  
  "dependencies": {  
    "EntityFramework.SqlServer": "7.0.0-alpha4",  
    "Microsoft.AspNet.Mvc": "6.0.0-alpha4",  
    "Microsoft.AspNet.Identity.SqlServer": "3.0.0-alpha4",  
    "Microsoft.AspNet.Identity.Authentication": "3.0.0-alpha4",  
    "Microsoft.AspNet.Security.Cookies": "1.0.0-alpha4",  
    "Microsoft.AspNet.Server.IIS": "1.0.0-alpha4",  
    "Microsoft.AspNet.Server.WebListener": "1.0.0-alpha4",  
    "Microsoft.AspNet.StaticFiles": "1.0.0-alpha4",  
    "Microsoft.Framework.ConfigurationModel.Json": "1.0.0-alpha4",  
    "Microsoft.VisualStudio.Web.BrowserLink.Loader": "14.0.0-alpha4",  
    "Moq": "4.2.1409.1722"  
  },  
  "commands": {  
    "web": "Microsoft.AspNet.Hosting --server Microsoft.AspNet.Server.WebListener --server.urls http://localhost:5001"  
  },  
  "frameworks": {  
    "aspnet50": {},  
    "aspnetcore50": {}  
  }  
}
```
    - config.json**:

```
{  
  "Data": {  
    "DefaultConnection": {  
      "ConnectionString": "Server=(localdb)\\mssqllocaldb;Database=aspnetvnext-WebApplication1;Trusted_Connection=True;MultipleActiveResultSets=true"  
    }  
  }  
}
```
  - Status Bar:** Ready, Ln 1, Col 2, Ch 2, INS, 5:08 PM, 10/14/2014.

# Introduction to ASP.NET

Questions?

# Free Trainings @ Telerik Academy

- ◆ C# Programming @ Telerik Academy

- ◆ [csharpfundamentals.telerik.com](http://csharpfundamentals.telerik.com)



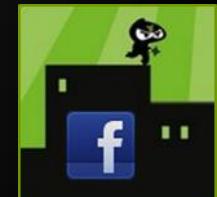
- ◆ Telerik Software Academy

- ◆ [academy.telerik.com](http://academy.telerik.com)



- ◆ Telerik Academy @ Facebook

- ◆ [facebook.com/TelerikAcademy](https://facebook.com/TelerikAcademy)



- ◆ Telerik Software Academy Forums

- ◆ [forums.academy.telerik.com](http://forums.academy.telerik.com)



1. Create and run few Web applications in Visual Studio to play with ASP.NET, compile and run them:
  - ASP.NET Web Forms application
  - ASP.NET MVC application
  - ASP.NET Web API application
  - ASP.NET Single Page application (SPA)
2. Write a simple application to sum numbers in ASP.NET Web Forms and ASP.NET MVC. Submit the code only (without the NuGet packages).
3. \* Write an HTTP handler that accepts a text as HTTP GET or POST request and returns as a result the text as PNG image. Map it to process \*.img requests.