Telerik Academy

Introduction to ASP.NET

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

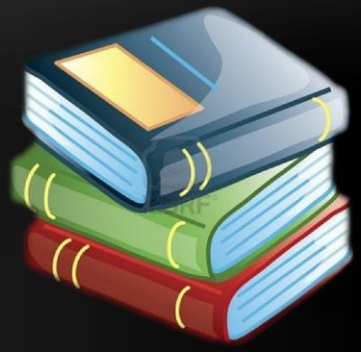
ASP.NET Web Forms

Telerik Software Academy

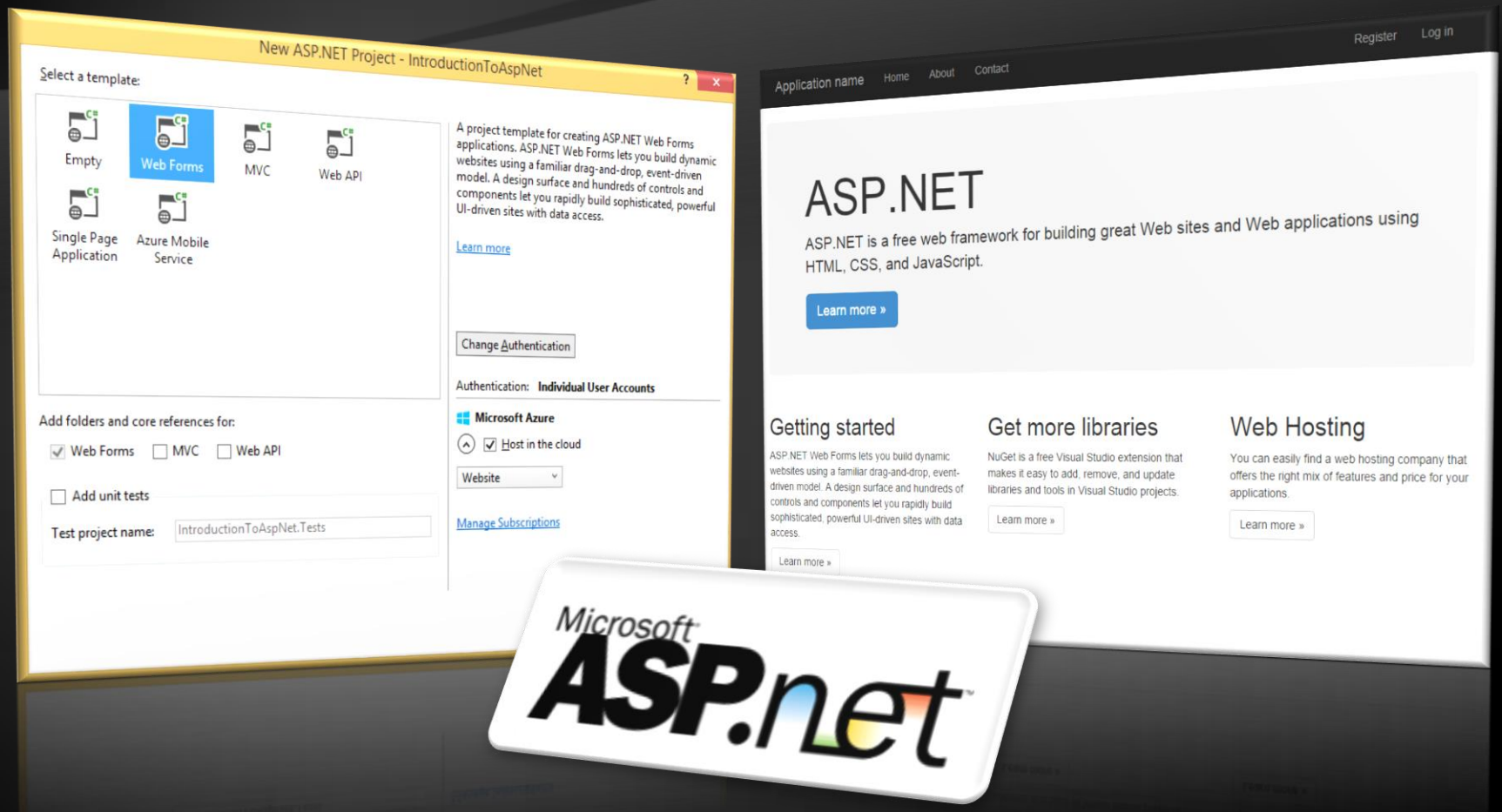
http://academy.telerik.com

Telerik Academy

Table of Contents



Telerik Academy

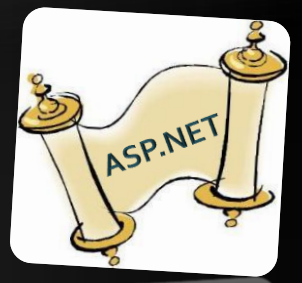


Microsoft ASP.net

Introduction to ASP.NET

History of ASP.NET

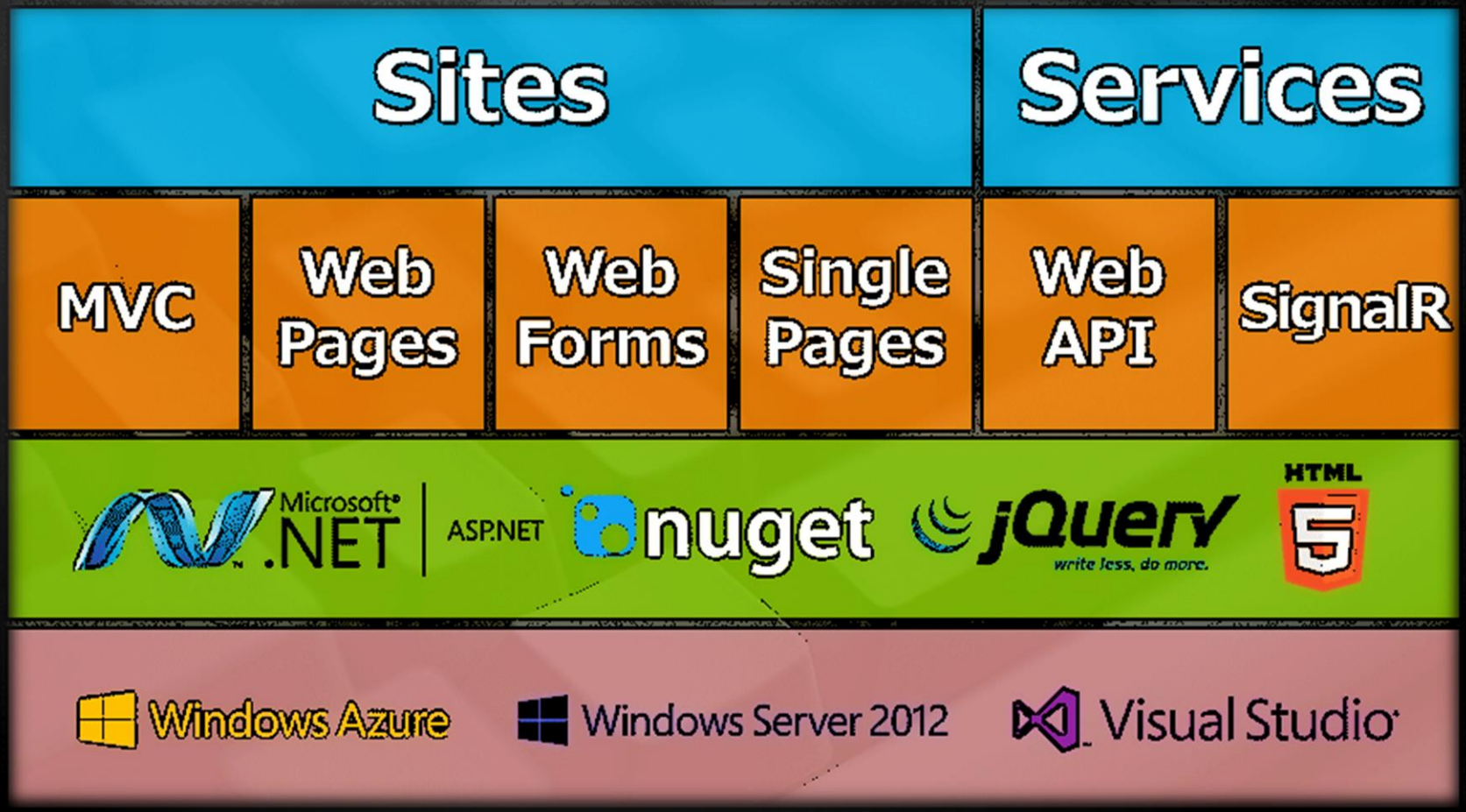
- At the beginning of Internet (up to 1997)
  - CGI, ISAPI (for C, C++), PHP
- Classic / Legacy ASP (1997-2002)
  - Based on VB Script, COM, ADO



- 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



Telerik Academy

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)

Telerik Academy

ASP.NET: Web Pages, Web API

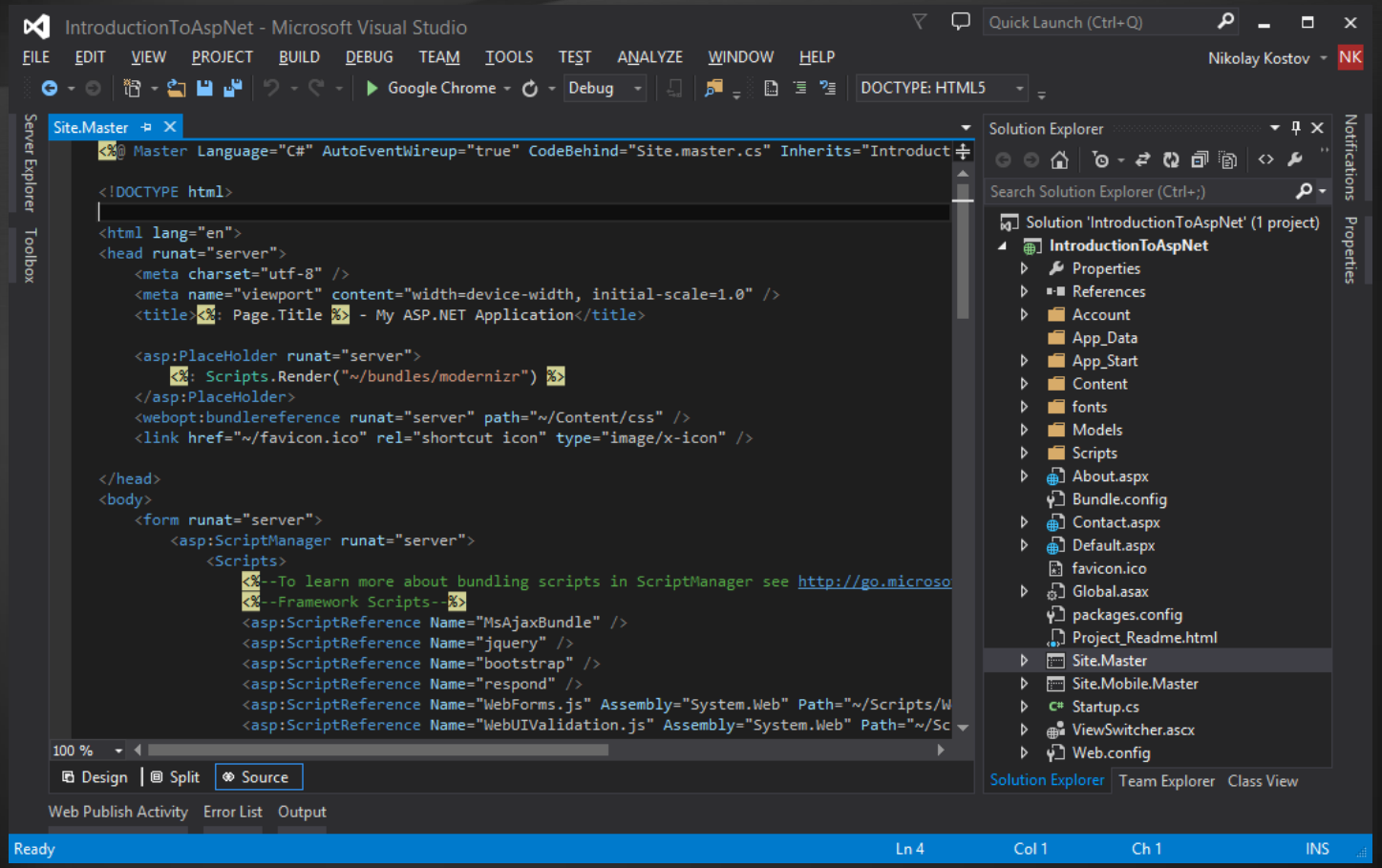
- ASP.NET Web Pages (read more)
  - Lightweight framework to combine server code with HTML to create dynamic web content
    - Like PHP: mix HTML code with C# code
    - Uses the "Razor" templating engine
- ASP.NET Web API (read more)
  - Framework for building RESTful Web services
    - Write C# code to handle HTTP requests in REST style (GET / POST / PUT / DELETE requests)
    - Return JSON / XML as result

Telerik Academy

ASP.NET: SPA, SignalR

- 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

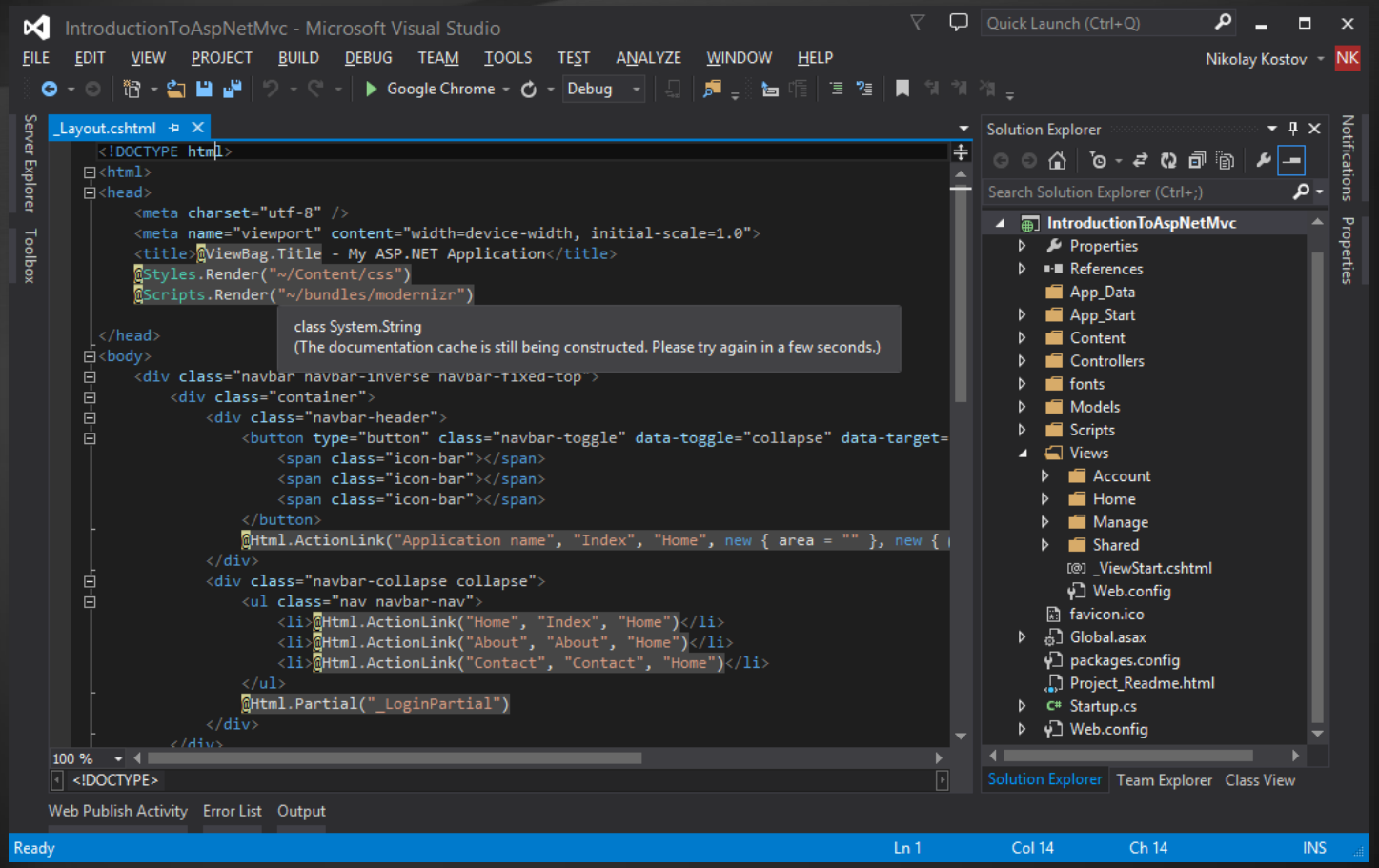
Telerik Academy



Simple Web Forms App

Live Demo

Telerik Academy



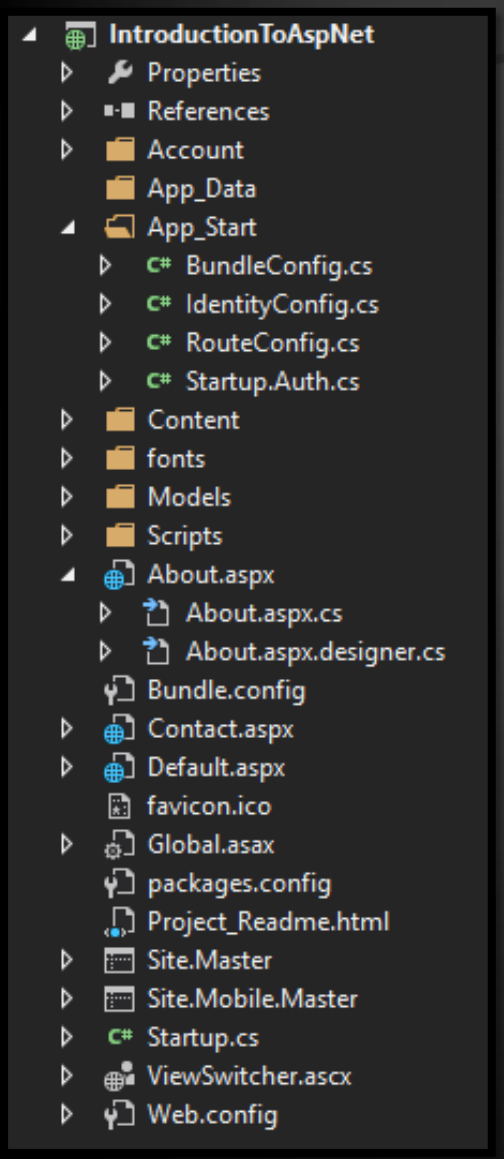
Simple MVC App

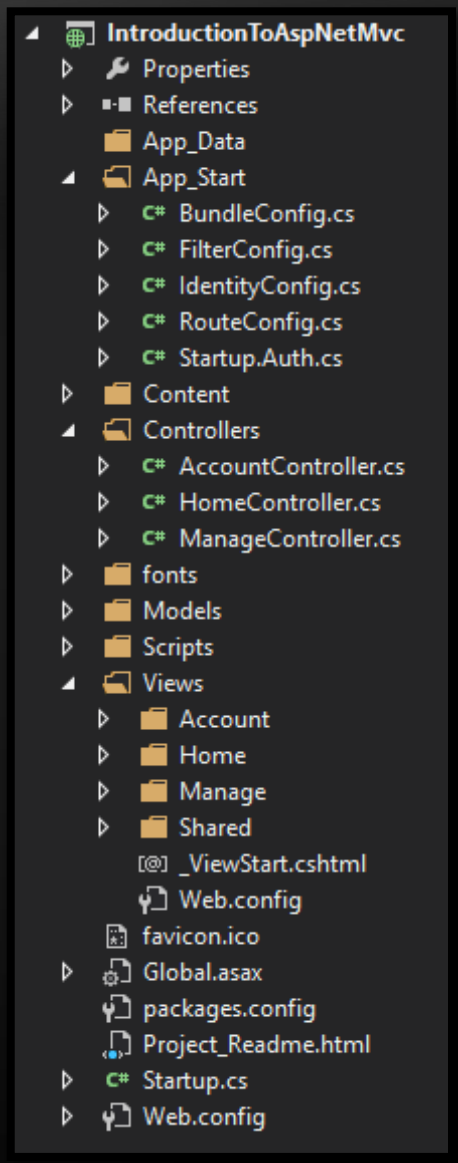
Live Demo

Telerik Academy

ASP.NET App Structure

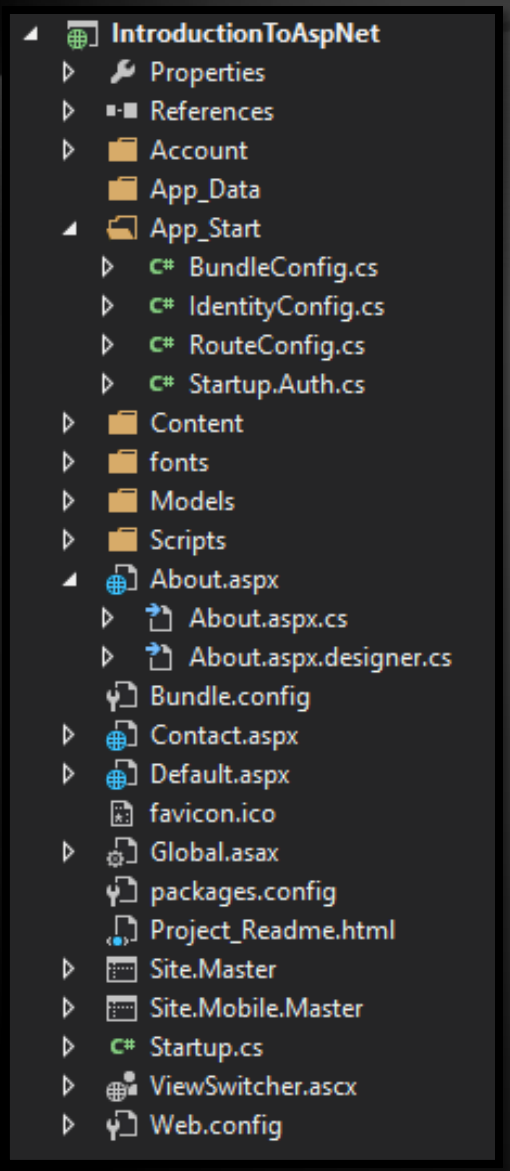
Typical Application Structure in ASP.NET





Telerik Academy

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

App_Start

- 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

App_Start (2)

- `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

App_Data

- 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

- `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 (2)

- 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

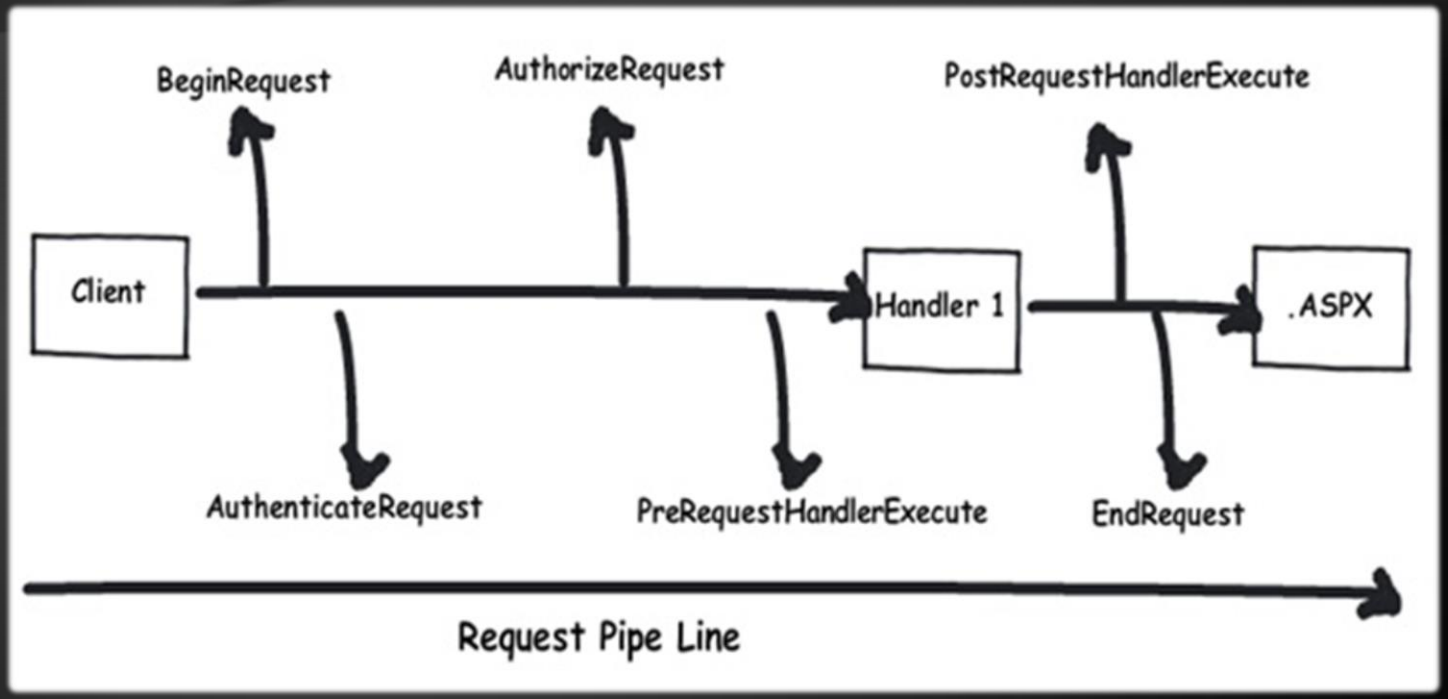
Web.config – Example

```
<?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

- 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.

Telerik Academy

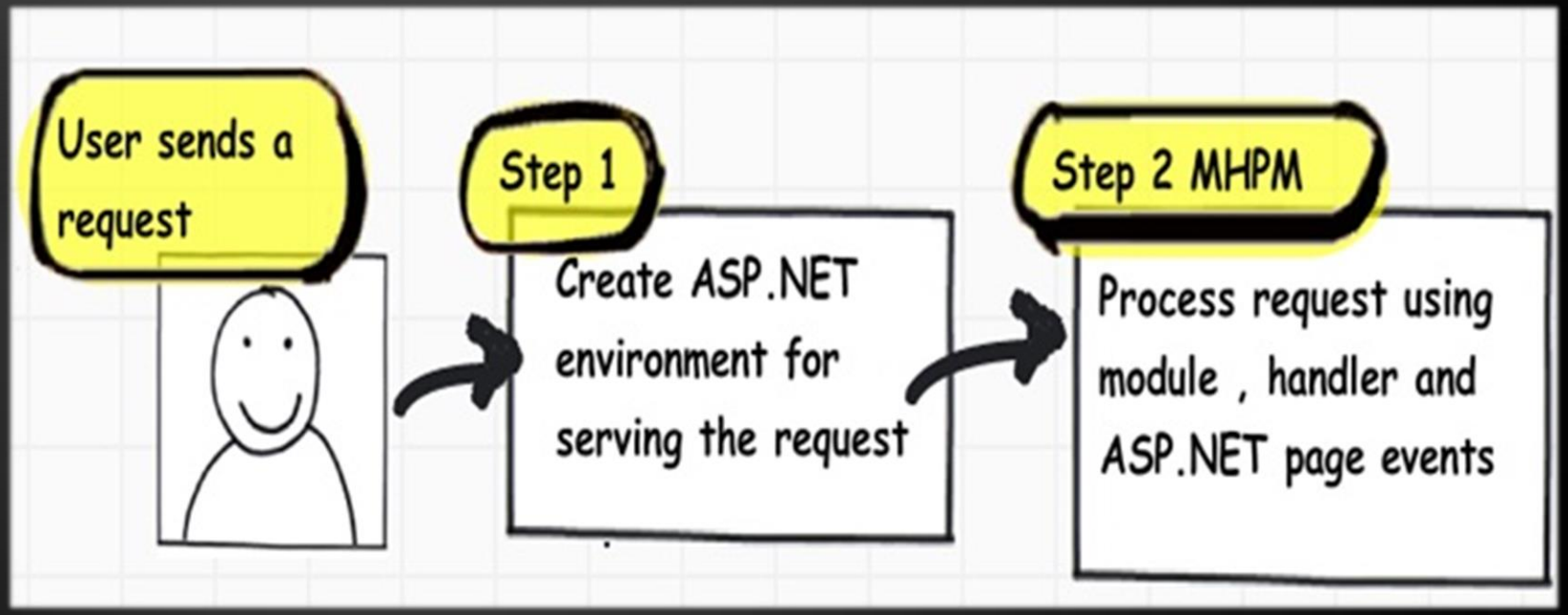


ASP.NET App Lifecycle

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

Telerik Academy

ASP.NET App Lifecycle

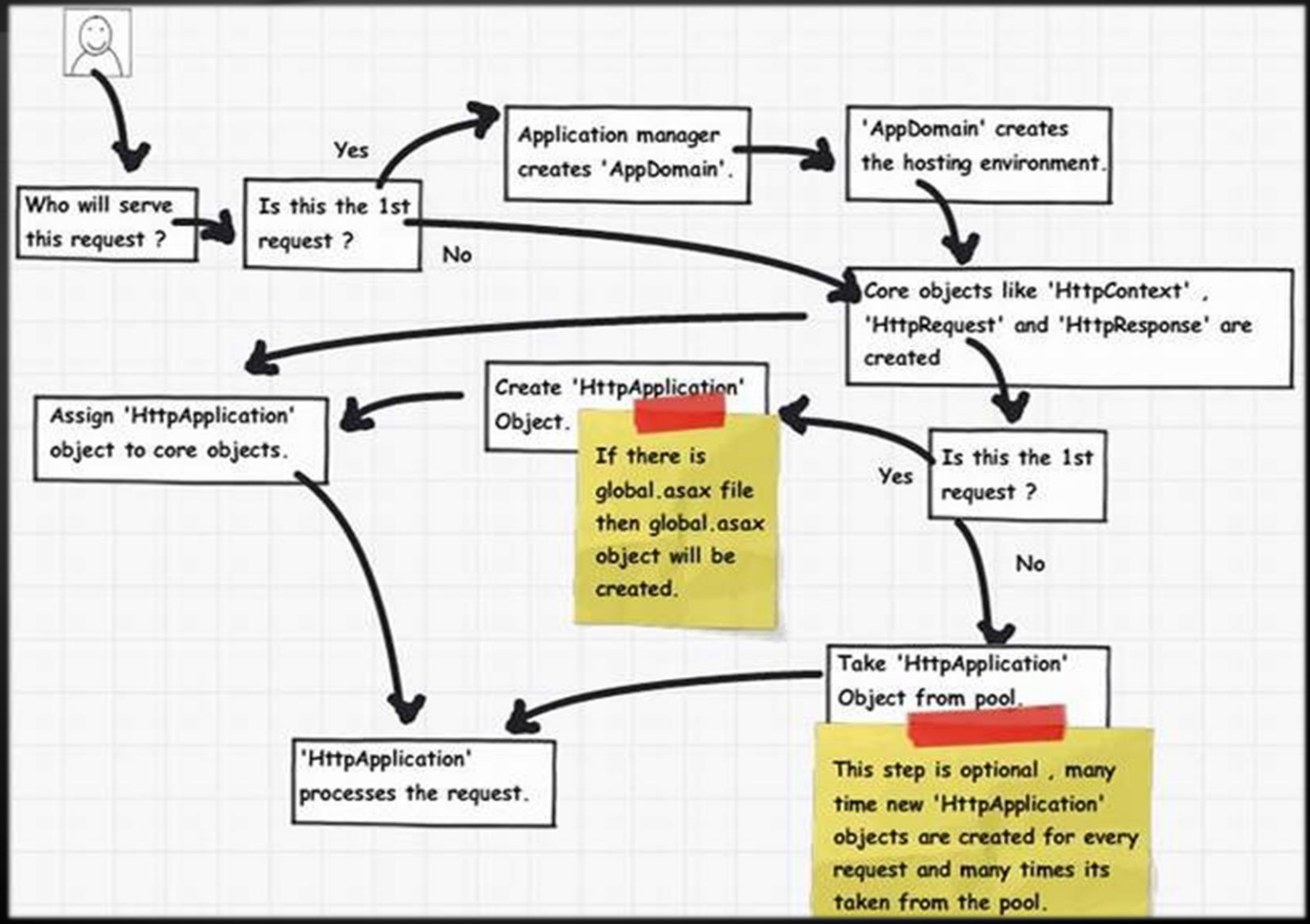


- MHPM == Module, Handler, Page Events, Module Events

* Source: http://www.codeproject.com/Articles/73728/ (by Shivprasad Koirala)

Telerik Academy

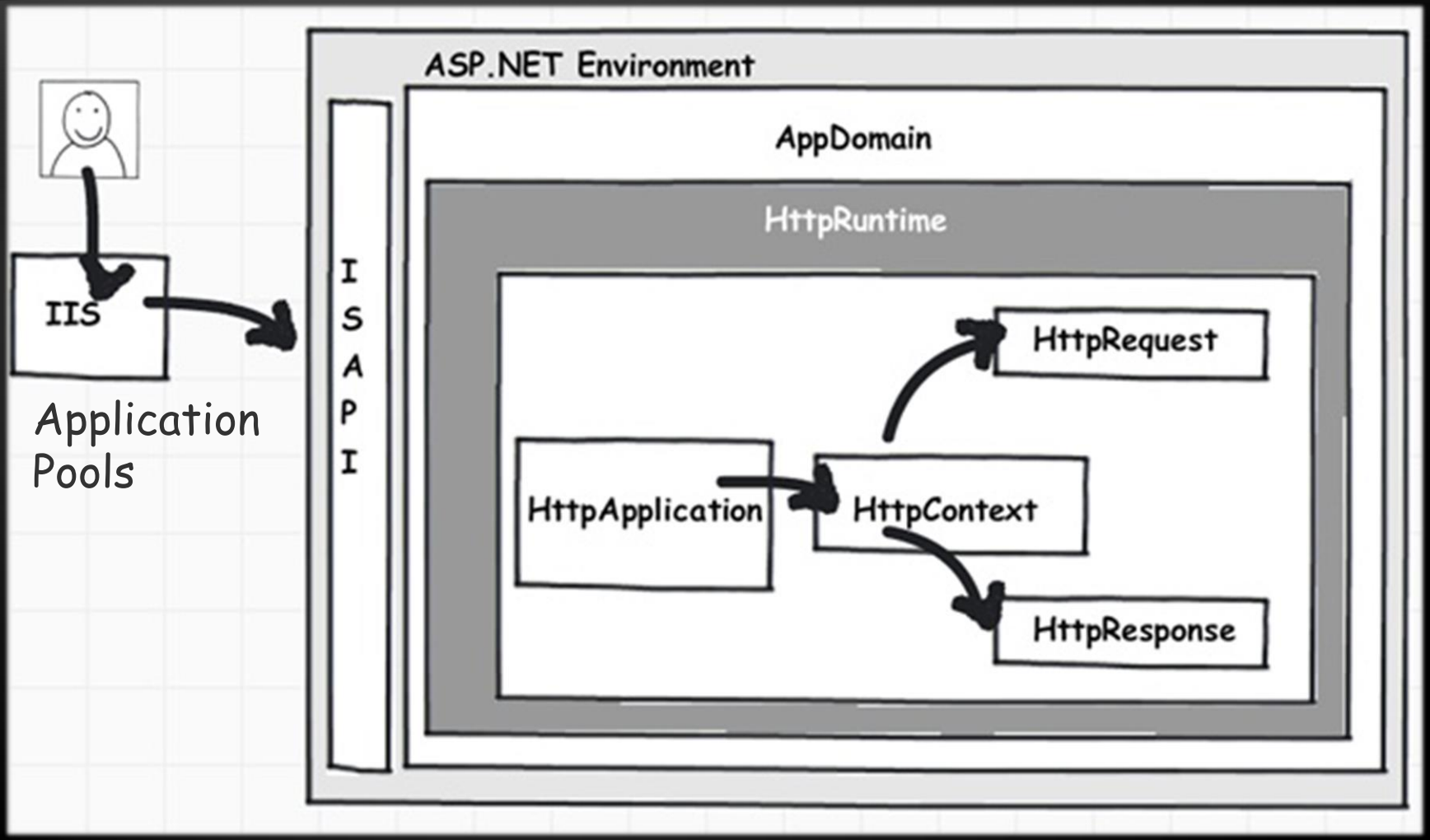
ASP.NET App Lifecycle (2)



* Source: http://www.codeproject.com/Articles/73728/

Telerik Academy

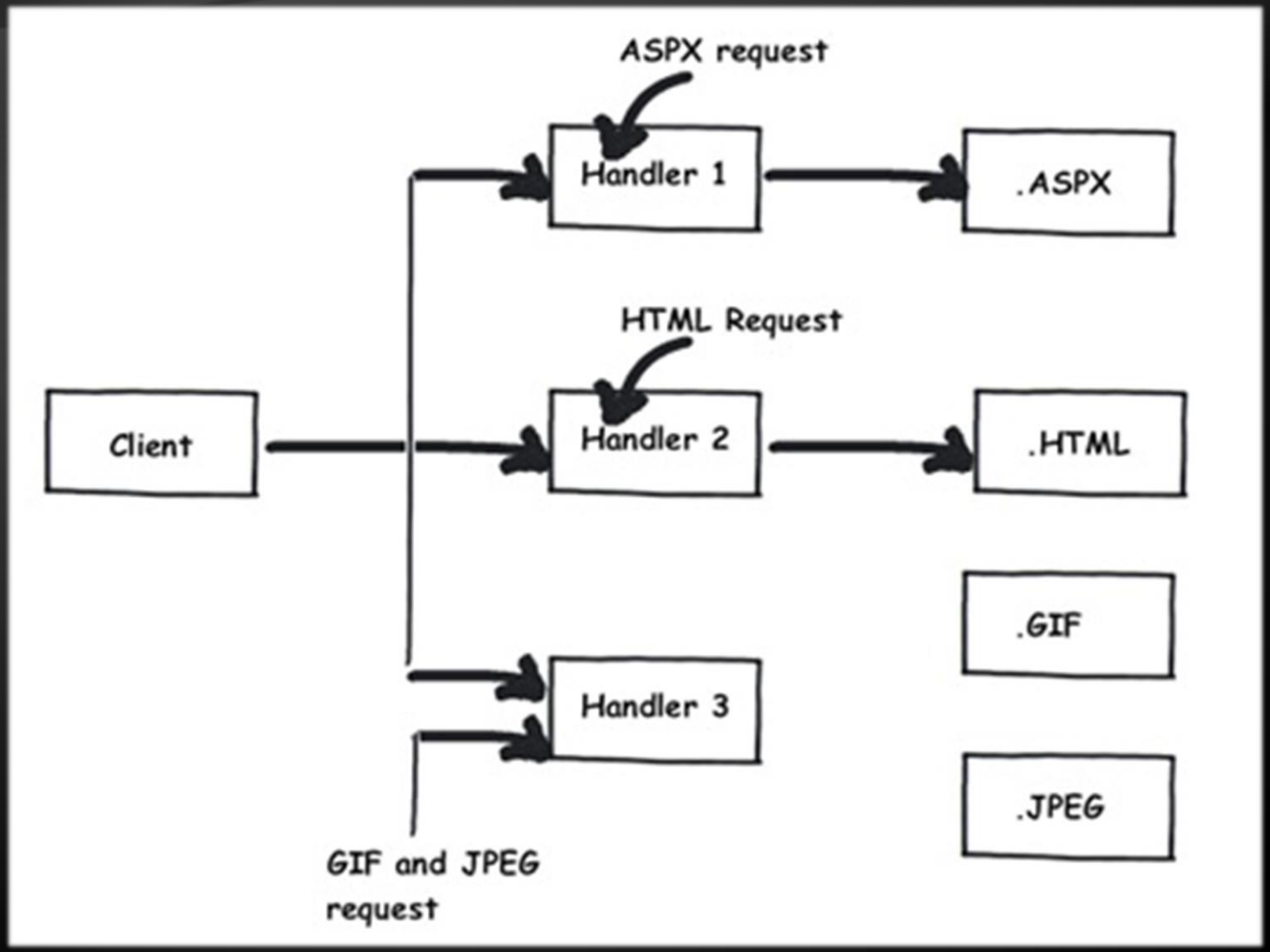
ASP.NET App Lifecycle (3)



* Source: http://www.codeproject.com/Articles/73728/

Telerik Academy

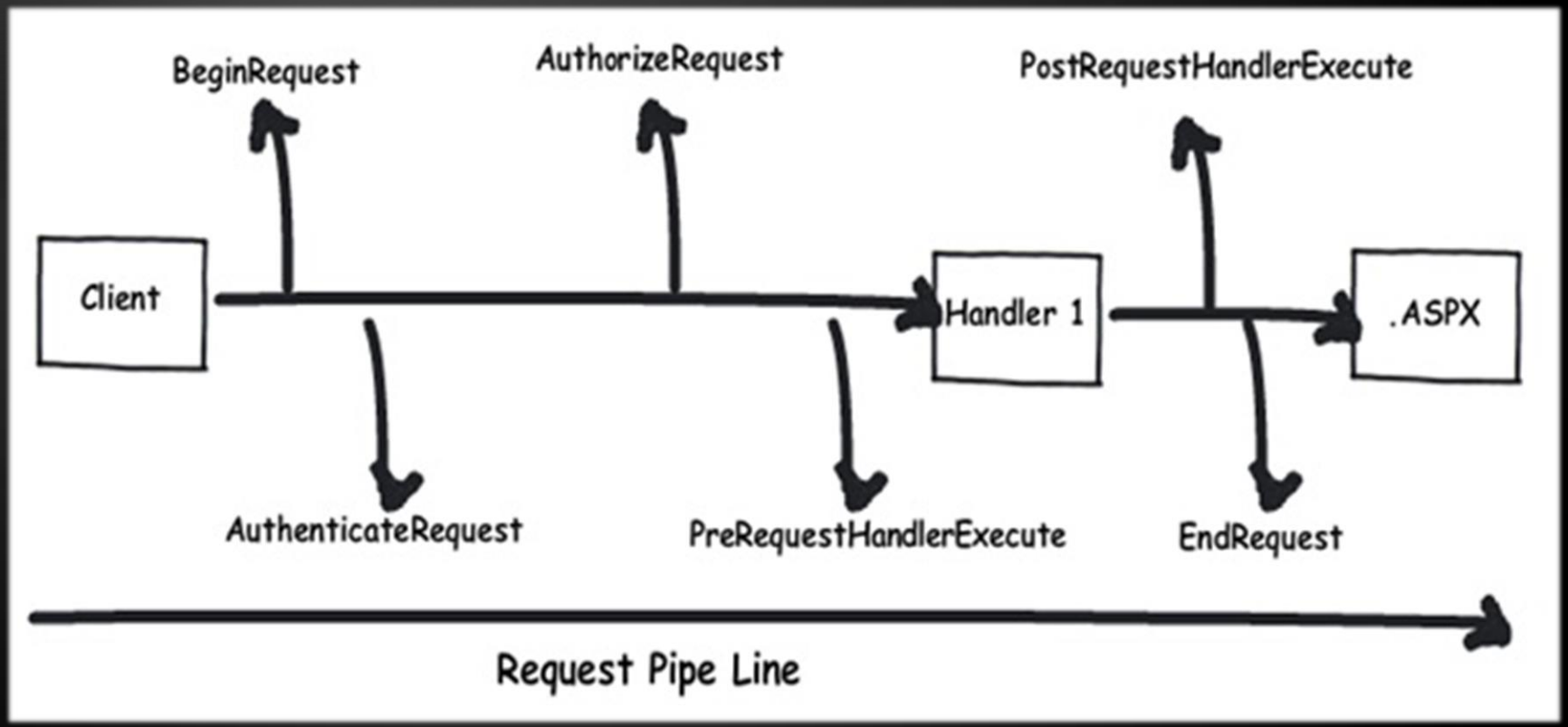
ASP.NET App Lifecycle (4)



* Source: http://www.codeproject.com/Articles/73728/

Telerik Academy

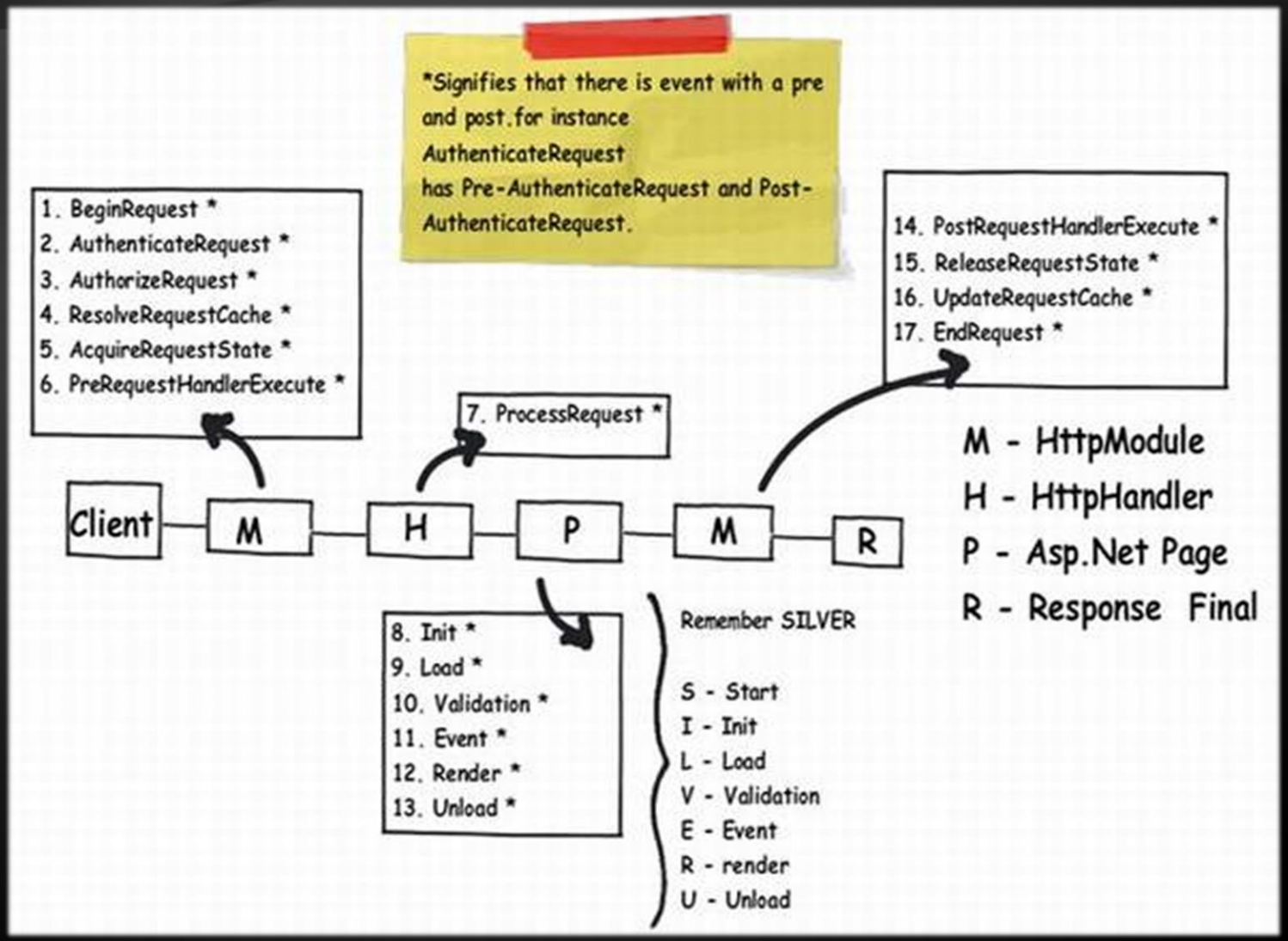
ASP.NET App Lifecycle (5)



* Source: http://www.codeproject.com/Articles/73728/

Telerik Academy

ASP.NET App Lifecycle (6)

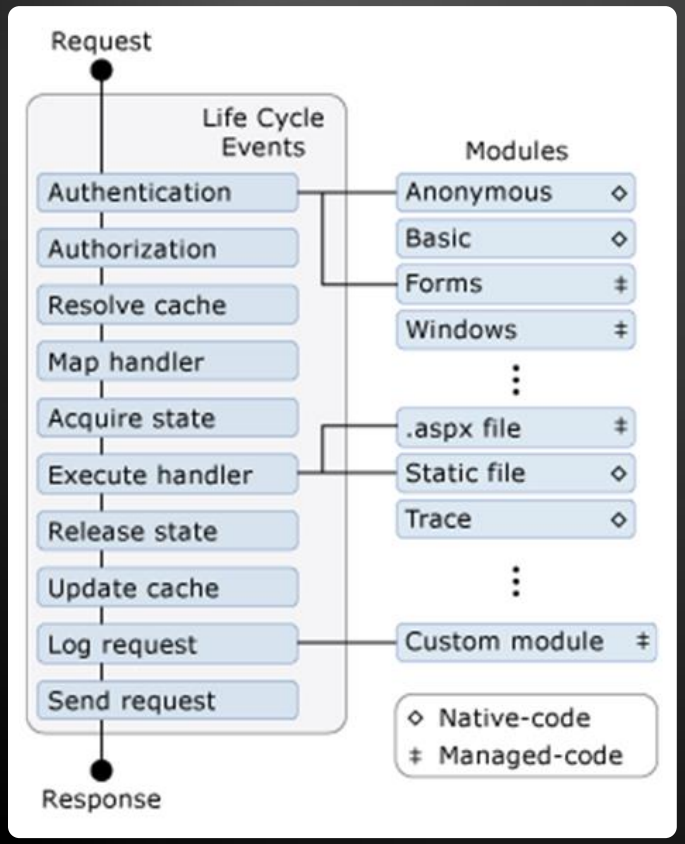


* Source: http://www.codeproject.com/Articles/73728/

Telerik Academy

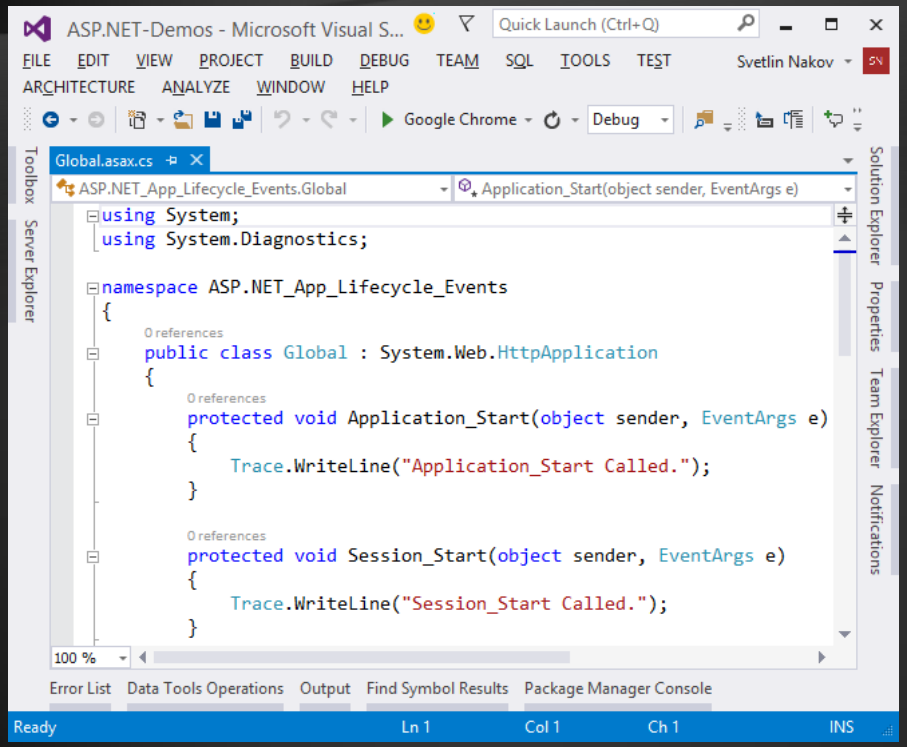
Application Lifecycle Events

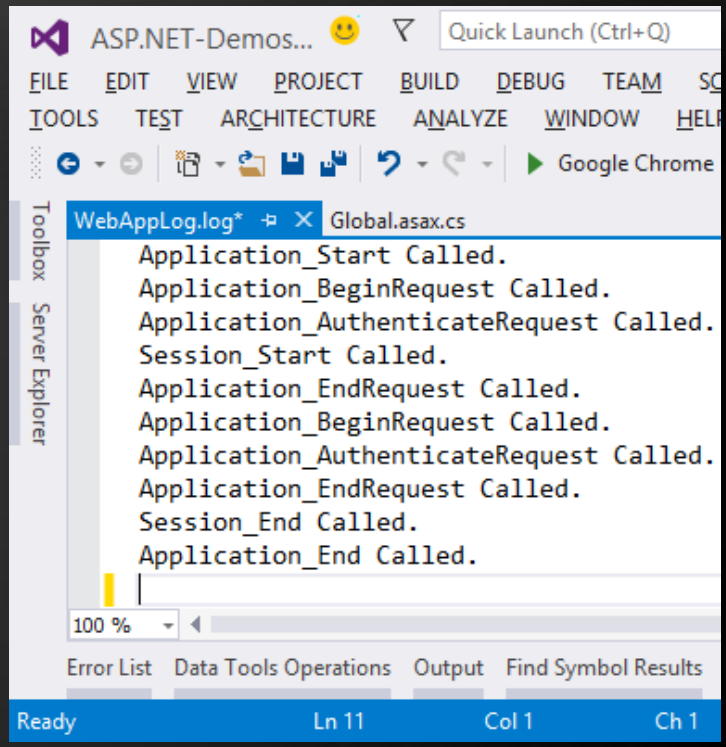
- `HttpApplication` have a complex pipeline to the process HTTP requests (read more)



- `BeginRequest`
- `AuthenticateRequest`
- `AuthorizeRequest`
- `AcquireRequestState`
- `ProcessRequest`
- `ReleaseRequestState`
- `EndRequest`

Telerik Academy





App Lifecycle Events

Live Demo

HTTP Handlers

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

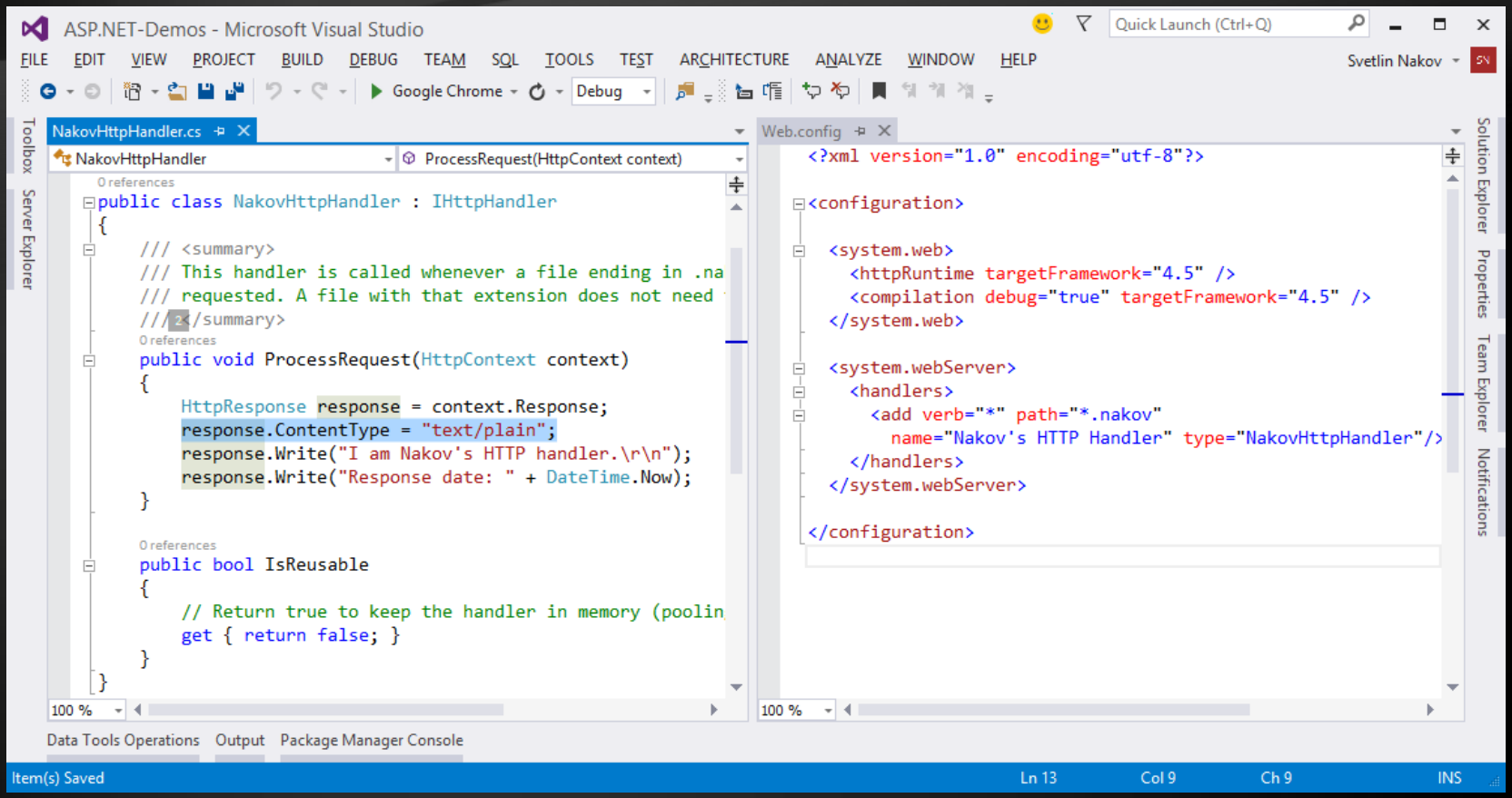
```
public class TelerikAcademyHttpHandler : IHttpHandler
{
  public void ProcessRequest(HttpContext context)
  { context.Response.Write("I am a HTTP handler."); }

  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>
```

Telerik Academy



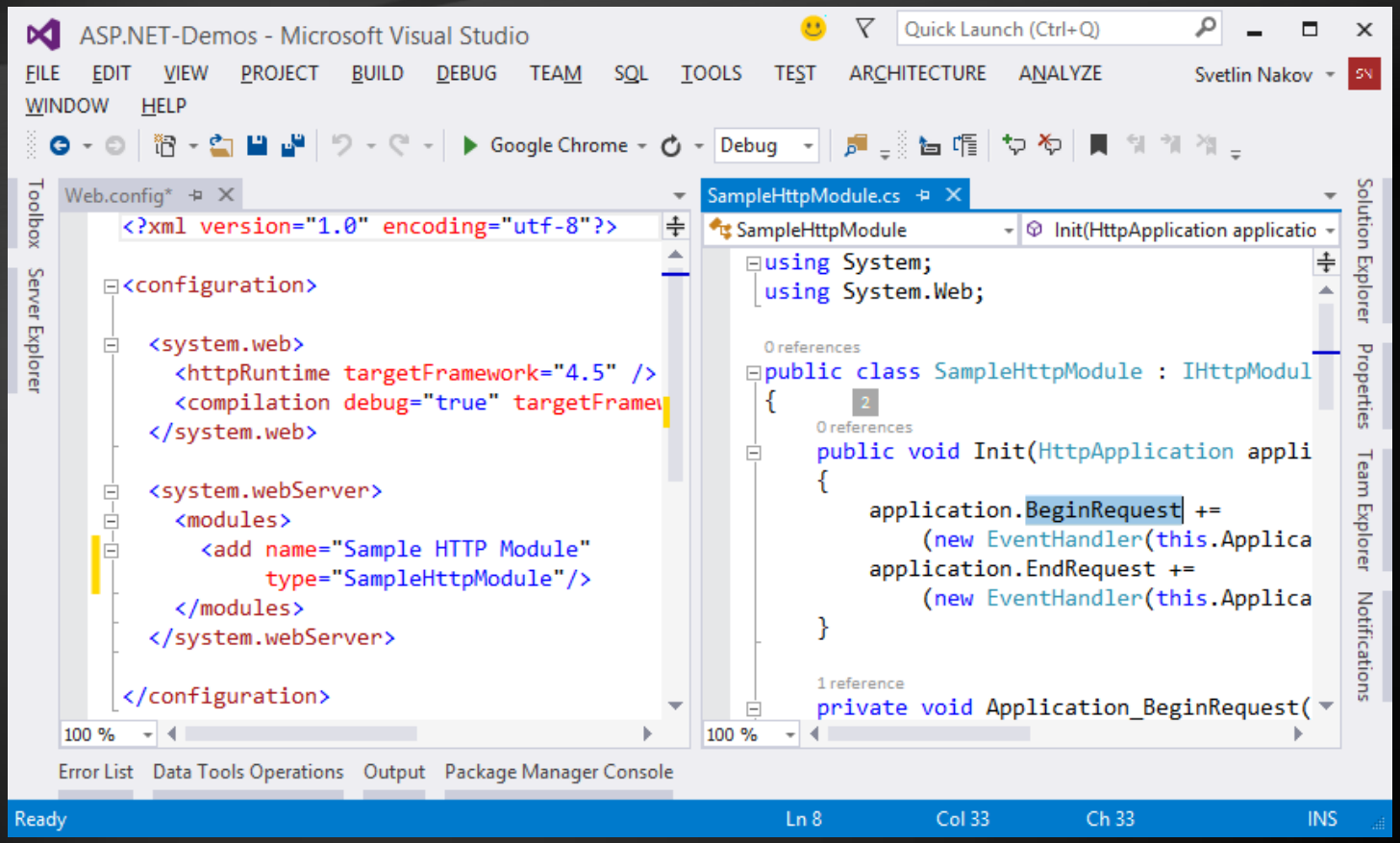
Writing a HTTP Handler

Live Demo

HTTP Modules

- 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

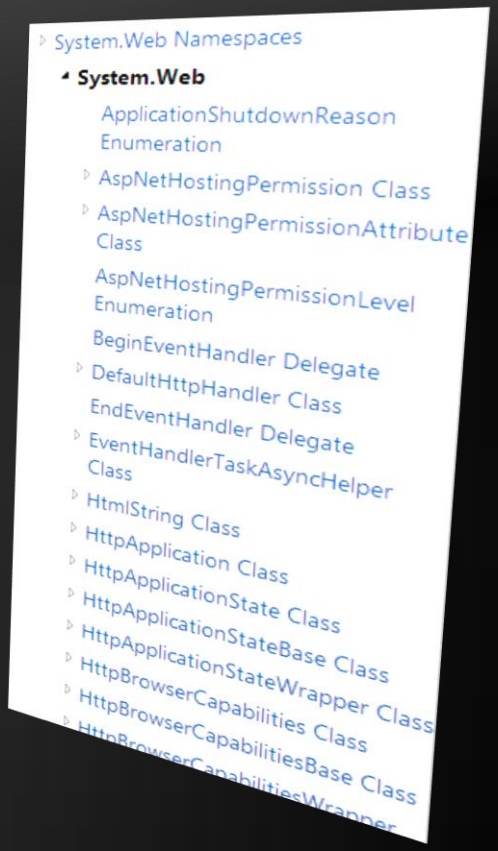
Telerik Academy



Writing a HTTP Module

Live Demo

Telerik Academy



ASP.NET Common Concepts

Major Classes, Namespaces, Web Sites, Web Apps

Telerik Academy

ASP.NET Namespaces

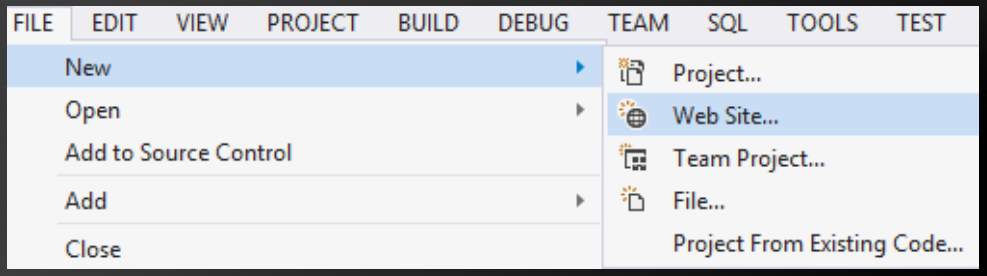
- Major ASP.NET (4.5) namespaces
  - `System.Web`
    - Web application main classes like `HttpApplication`, `HttpContext`, `HttpRequest`, `HttpResponse`, `HttpSessionState`, …
  - `System.Web.Mvc`
    - MVC classes and framework components
  - `System.Web.UI`
    - Web Forms UI controls (like `Button` and `Label`)

ASP.NET Classes

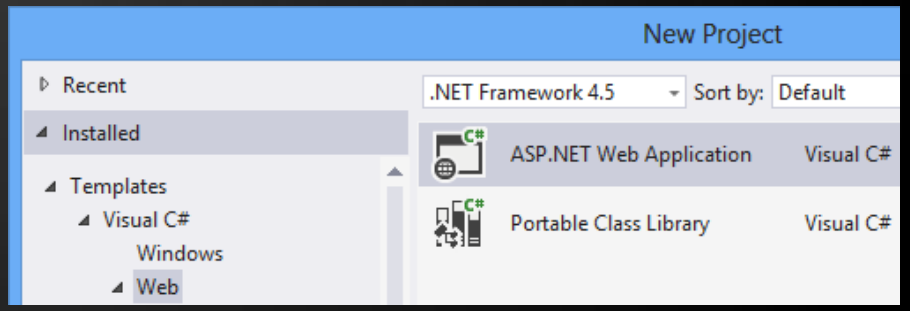
- `HttpApplication`
  - Base class for the ASP.NET Web apps (inherited in `Global.asax`)
- `HttpContext`
  - Encapsulates all HTTP-specific information about an individual HTTP request
- `HttpRequest`
  - Encapsulates an HTTP request
- `HttpResponse`
  - Encapsulates an HTTP response

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)



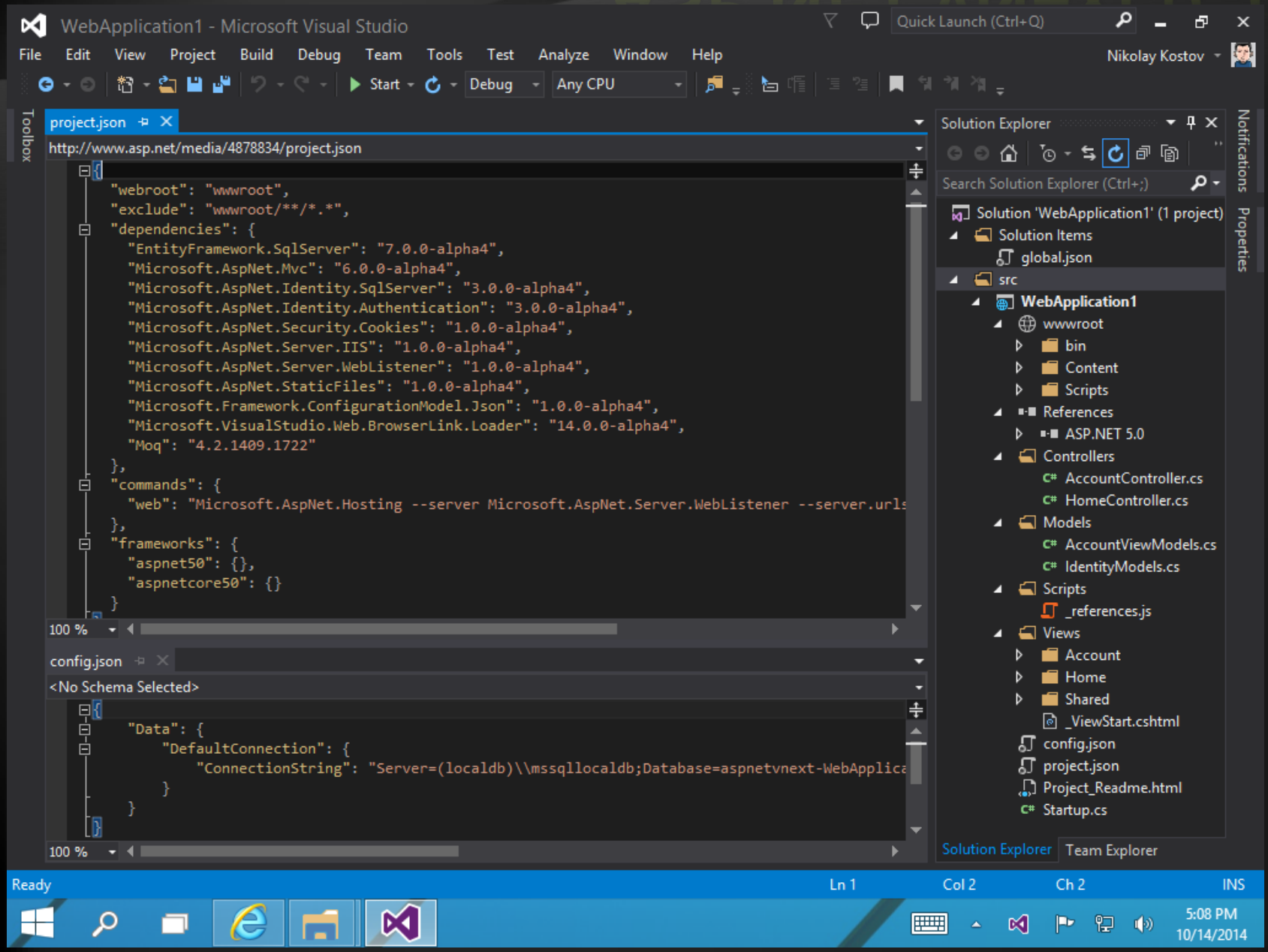
- Web Apps in VS
  - Have project file (like any C# project)
  - Compilation produces an assembly: `bin\*.dll`



- Web apps are recommended (read more)

Telerik Academy

ASP.NET vNext (5.0)



Telerik Academy

Introduction to ASP.NET

http://academy.telerik.com

Free Trainings @ Telerik Academy

- C# Programming @ Telerik Academy
  - csharpfundamentals.telerik.com

- Telerik Software Academy
  - academy.telerik.com

- Telerik Academy @ Facebook
  - facebook.com/TelerikAcademy

- Telerik Software Academy Forums
  - forums.academy.telerik.com

Exercises

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.