PV247 – Development I

Introduction to ASP.NET and related technologies

Kentico

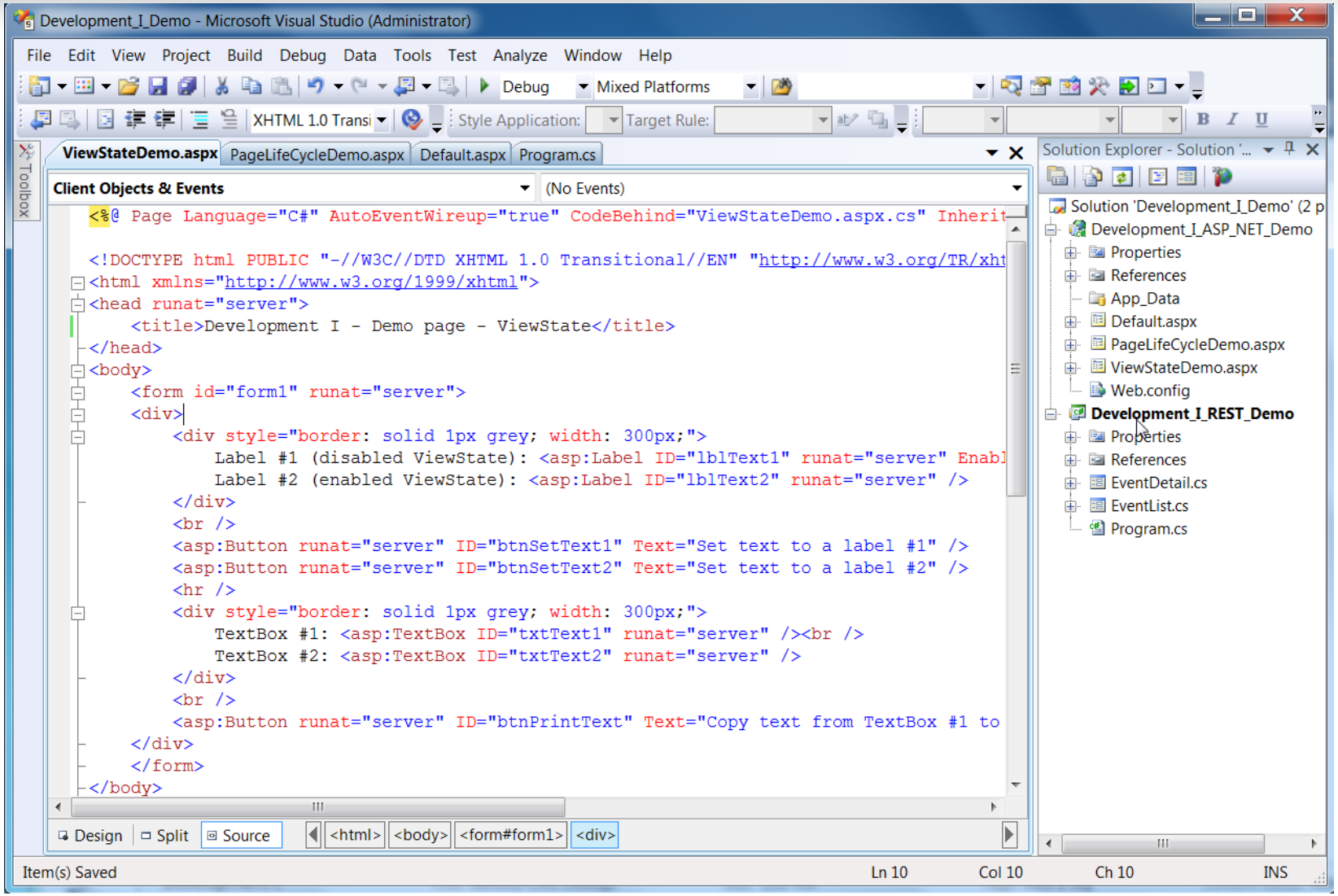
Overview

- **ASP.NET Basics**
  - What is ASP.NET?
  - What is a request?
  - How does ASP.NET deal with stateless http?
  - ASP.NET request/page/control life cycle
  - REST Services
- **Kentico CMS Platform Basics**
  - CMS.IO namespace

What is ASP.NET?

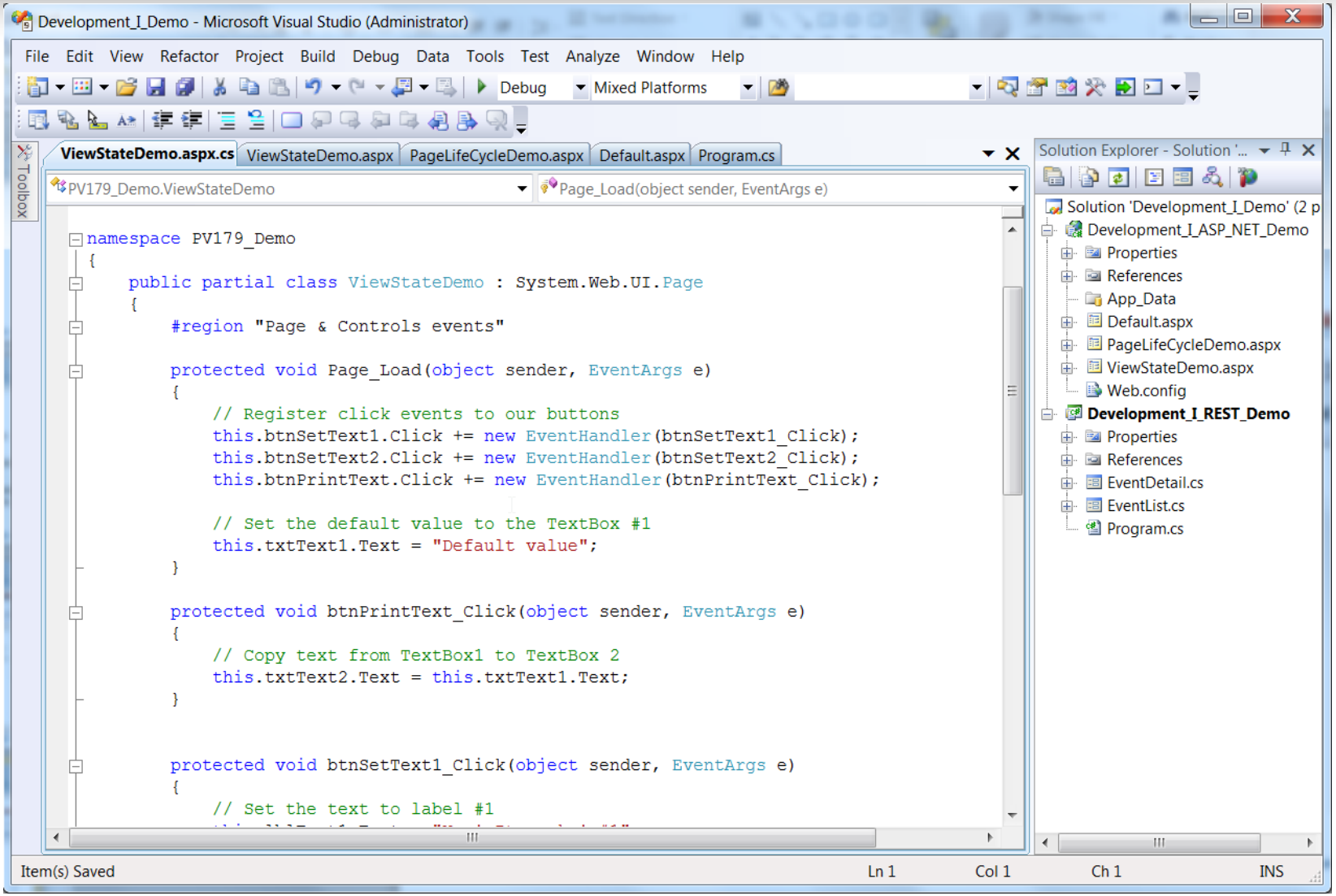
- **Active Server Pages .NET**
- Platform for creating dynamic web applications
- You can use any .NET language as a code-behind
- Development of ASP.NET WebForms applications can be similar to the development of WinForms applicatons. Similar, not same!

Example of ASP.NET page



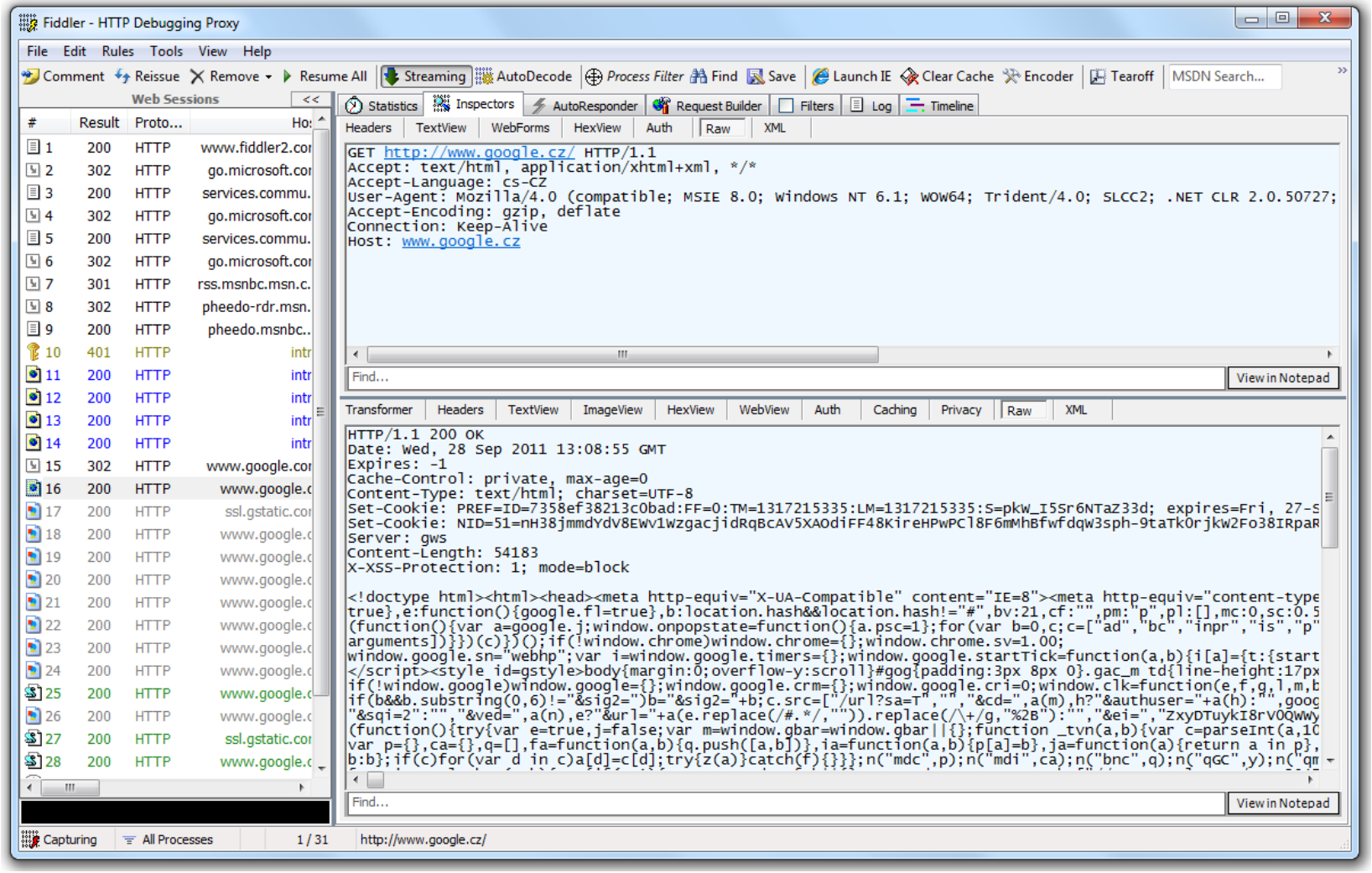
Kentico

ASP.NET page – code behind



Kentico

What is a request?



Kentico

What is a request?

- Main thing you need to remember about HTTP protocol:

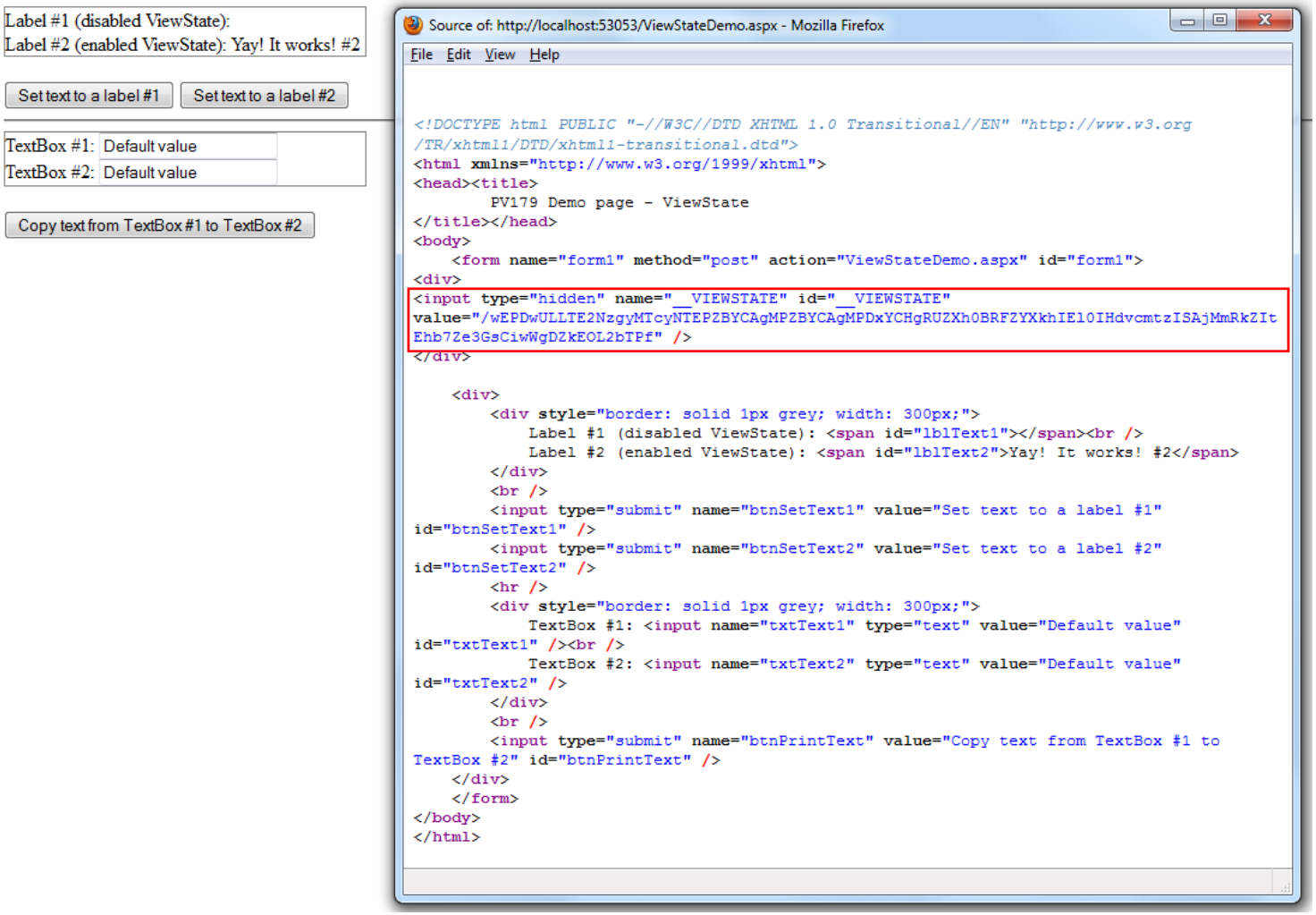
  HTTP is stateless protocol!

- But we need state in dynamic web applications!

How does ASP.NET deal with stateless http?

- The answer is ... **ViewState**!
- It is a technique used by an ASP.NET Web page to persist changes to the state of a Web Form across postbacks (HTTP POST to the same page that the form is on).
- **Use ViewState carefuly** and only when it‘s really needed! It‘s helpful technique, but it might become too greedy and can cause the application to be less effective.

ViewState – How is it send within requests?



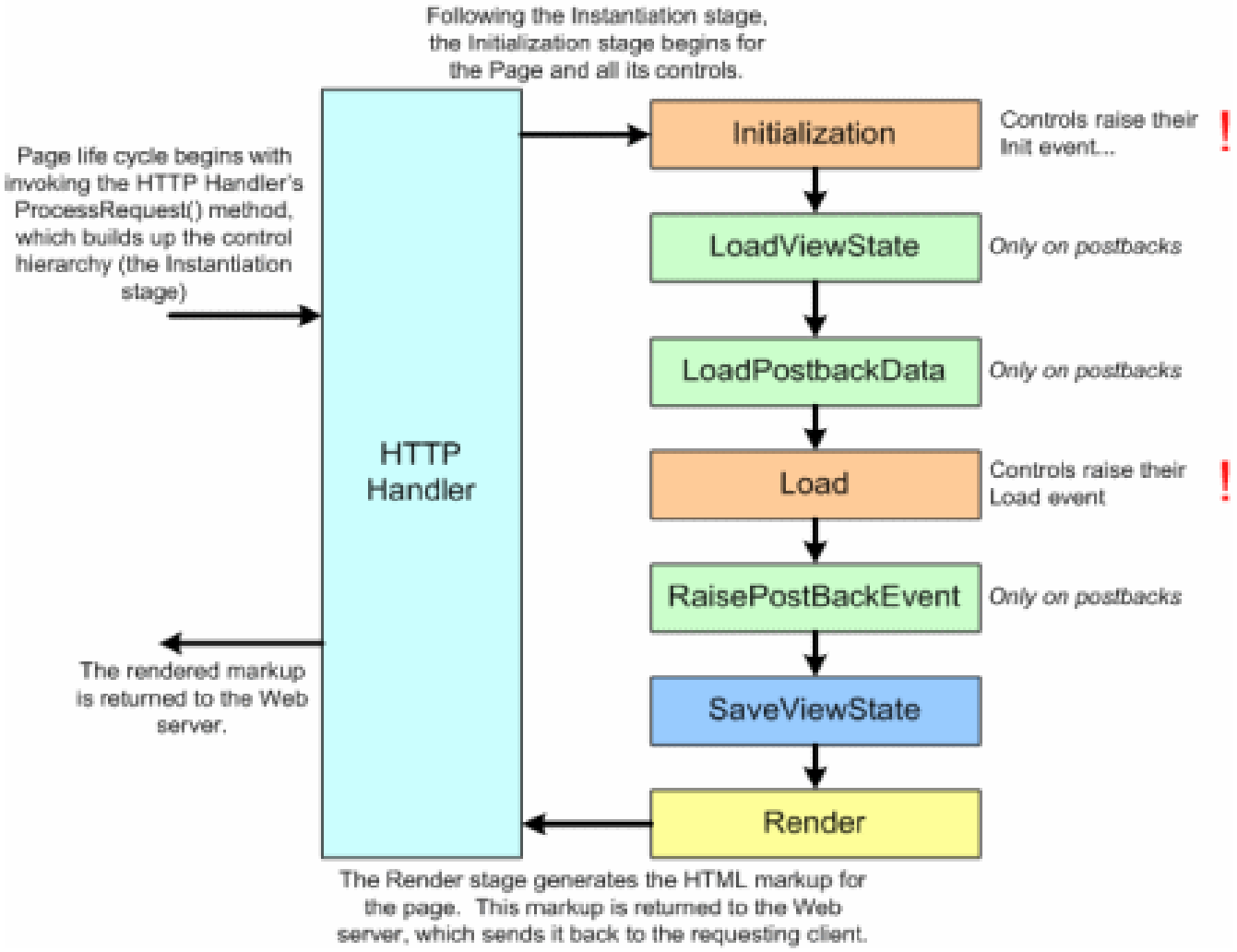
Kentico

ASP.NET page/control life cycle

- To be able to work with ASP.NET pages and controls properly you need to understand the life cycle of these elements.

- Most importat phases of page/control life cycle are:
  - PreInit
  - Init
  - Load
  - PreRender
  - Render

ASP.NET page/control life cycle



Source: http://i.msdn.microsoft.com/dynimg/IC152667.gif

Where to get more information?

- Where to start:
  http://msdn.microsoft.com/en-us/library/ywdtth2f%28v=vs.71%29.aspx
- More about page life cycle:
  http://msdn.microsoft.com/en-us/library/ms178472.aspx
- More details about how the ViewState works:
  http://msdn.microsoft.com/en-us/library/ms972976.aspx

Let‘s have a REST!

- **Representational state transfer** (**REST**) is a style of a service architecture
- Instead of complex technologies (RPC, SOAP, etc.) you use simple HTTP requests which are supported by all the clients!
- Commonly used formates of REST responses are XML, JSON, AtomPub

Let‘s have a REST!

- Conforming to the REST constraints is referred to as being "**RESTful**„ service
  - Access the objects within the system defining clear structure of URLs
  - The structure of URLs should be self-explaining and self-navigating
- Kentico CMS supports RESTful service

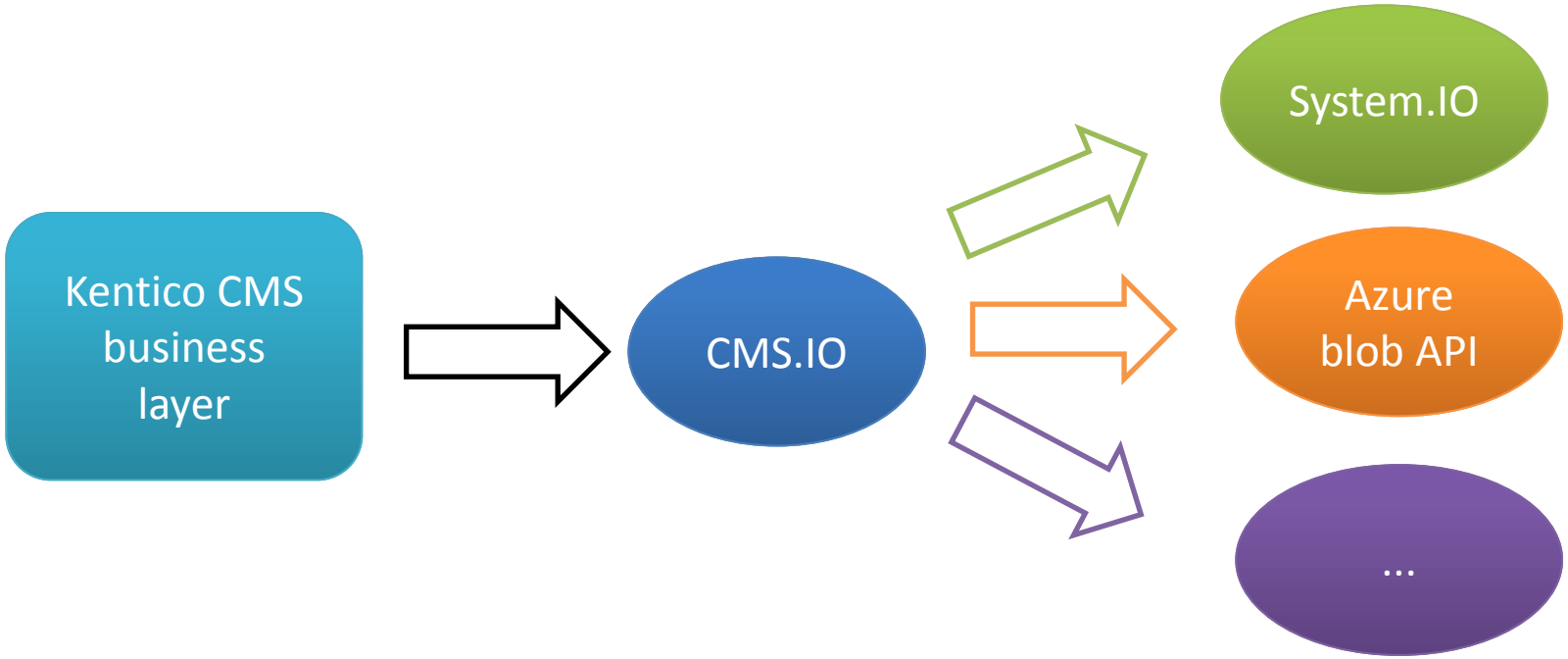
How to request a REST Service

- Create request
- Wait for response
- That‘s all!

- DEMO

CMS.IO

Layer between Kentico CMS and System.IO



Kentico

CMS.IO

- Allows the system to use various IO providers (local file system, Azure Blob, Amazon S3, …)
- To implement a provider you inherit from the base class and override the IO methods
  - Directory
  - File
  - FileStream
  - etc.