Introduction to ASP.NET

Ver 20.8.2013 (16:10)

***Demos created and content additions*:**
Esa Salmikangas
ICT/School of Technology
JAMK University of Applied Sciences

Agenda

- **Intro & Background**
- ASP.NET Overview
- Programming Model
- Programming Basics
- Server Controls
- ADO.NET & Data Binding
- Advanced topics
- Conclusion

INTRO

Where to start asp.net?

ASP.NET

Where to start?

- http://www.microsoft.com/net/Resources.aspx
- http://msdn.microsoft.com/en-us/netframework/default.aspx

Prerequisites

- This module assumes that you understand the fundamentals of
  - .NET Framework
  - C# programming
  - ADO.NET
- A background in web development (HTML, JavaScript, DHTML, CGI, Active Server Pages) would be helpful, but is not required

Learning Objectives

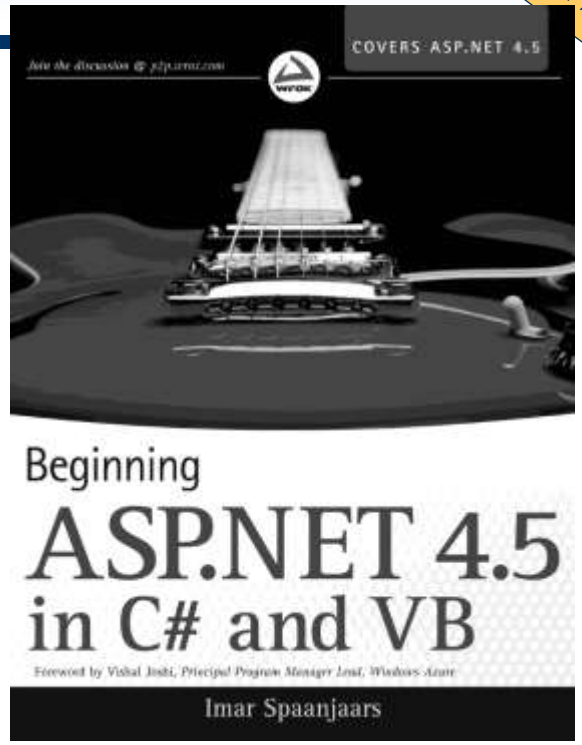
- What is ASP.NET
  - ~~History, why it was developed~~
- ASP.NET fundamental features
  - Programming model
  - HTML, CSS
  - Web Forms
  - Data handling with ADO.NET
  - Authentication and some special issues

Imaar Spaanjaars: Beginning ASP.NET 4.5

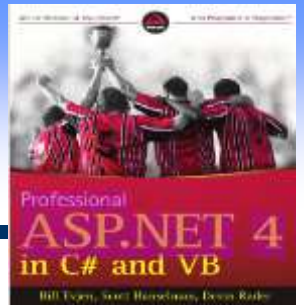
Kirjallisuutta

Beginning ASP.NET 4.5: in C# and VB

Contents



Evjen, Hanselman, Rader: Professional ASP.NET 4



http://site.ebrary.com/lib/jypoly/docDetail.action?docID=10373033

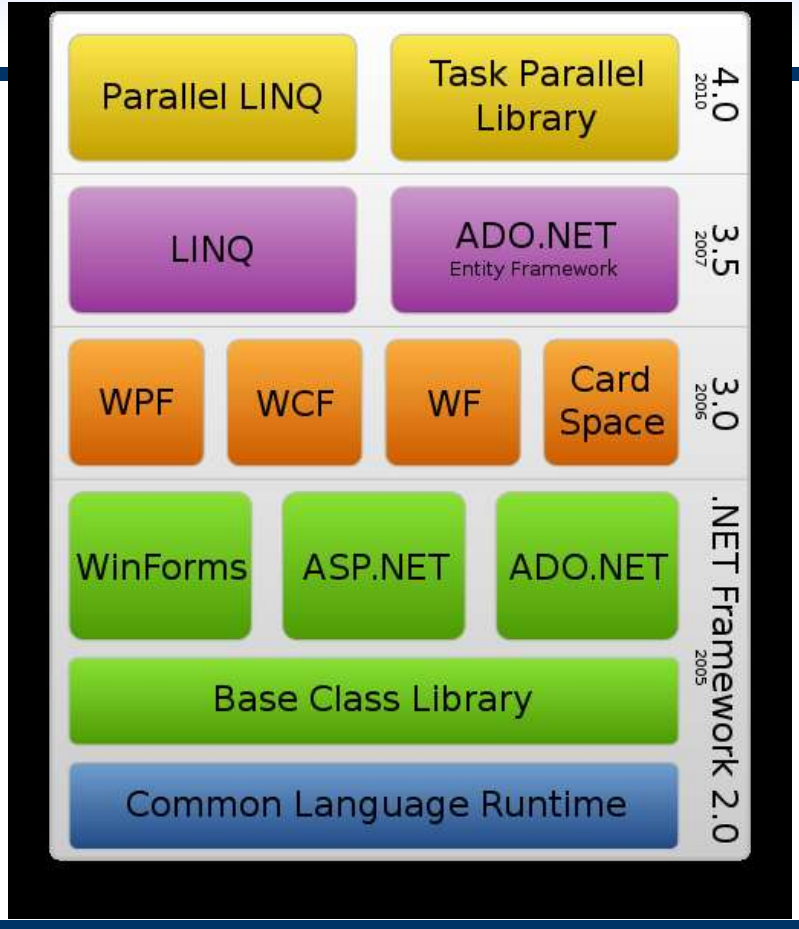
Vertailuksi

TECHNOLOGY INTRO

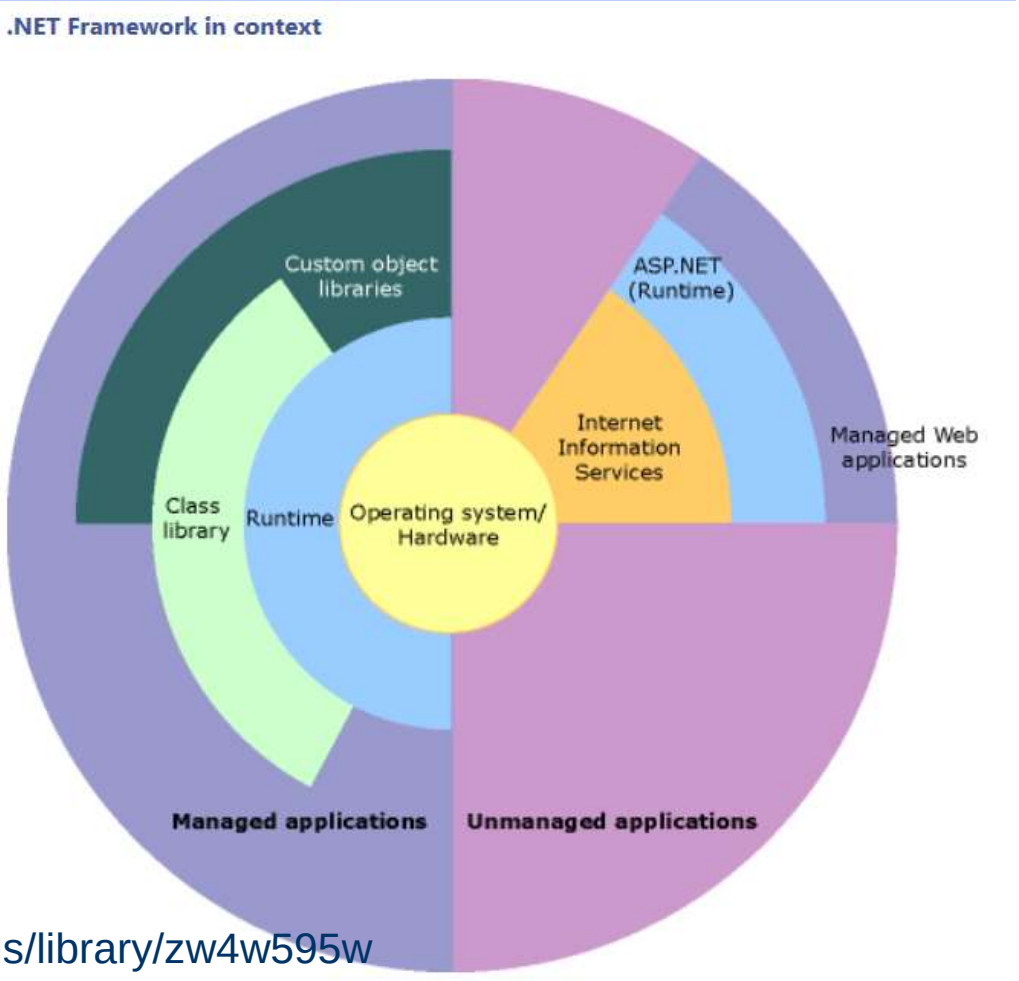
.NET Framework versions ~ ASP.NET versions

Version	Released	Visual Studio	Default in Windows
1.0	Feb 2002	Visual Studio.NET	
1.1	Apr 2003	Visual Studio 2003	Windows Server 2003
2.0	Nov 2005	Visual Studio 2005	Windows Server 2003R2
3.0	Nov 2006		Windows Vista, Server 2008
3.5	Nov 2007	Visual Studio 2008	Windows 7, Server 2008R2
4.0	Apr 2010	Visual Studio 2010	
4.5	2012-08-15	Visual Studio 2012	Windows 8, Server 2012

The .NET Framework Stack



Overview of the .NET Framework



source: http://msdn.microsoft.com/en-us/library/zw4w595w

ASP.NET development models

ASP.NET supports three different development models:

Web Pages Single Pages Model	MVC Model View Controller	Web Forms Event Driven Model
Simplest ASP.NET model. Similar to PHP and classic ASP. Built-in templates and helpers for database, video, graphics, social media and more.	MVC separates web applications into 3 different components: Models for data Views for display Controllers for input	The traditional ASP.NET event driven development model: Web pages with added server controls, server events, and server code.

Three kind of ASP.NET technologies for creating dynamic web applications

- **ASP.NET Web Pages** focuses on adding dynamic (server-side) code and database access to HTML pages, and features simple and lightweight syntax.
- **ASP.NET Web Forms** is based on a page object model and traditional window-type controls (buttons, lists, etc.). Web Forms uses an event-based model that's familiar to those who've worked with client-based (Windows forms) development.
- **ASP.NET MVC** implements the model-view-controller pattern for ASP.NET. The emphasis is on "separation of concerns" (processing, data, and UI layers).

source: http://www.asp.net/web-pages/overview/getting-started/aspnet-web-pages-(razor)-faq

What’s New?

What's New in the .NET Framework 4.5

What's New in the ASP.NET 4.5

- .NET for Windows Store App
- Portable Class Libraries
- Core New Features and Improvements
- Managed Extensibility Framework
- Asyncronous File Operations
- Tools
- Parallel Computing
- Web: Support for new HTML5 form types etc
- Networking, WPF, WCF, WF

source: http://msdn.microsoft.com/en-us/library/ms171868.aspx

ASP.NET Web Pages 2

- ASP.NET Web Pages with **Razor** syntax is a programming framework for creating web applications
- A Part of Microsoft WebMatrix2 , free development environmet

ASP.NET Razor

- ASP.NET Razor syntax uses a simple programming syntax that lets you embed server-based code into a web page.
- The page can also contain HTML markup, CSS information, and client script (JavaScript and jQuery).
- Razor syntax is based on ASP.NET, which is the part of the .NET Framework that's specifically designed for creating web applications.
- Razor syntax gives you all the power of ASP.NET, but it uses a simplified syntax that's easier to learn if you are a beginner. If you're an expert, it makes you more productive. Even though this syntax is easy to use, its relationship to ASP.NET means that as your web applications become more sophisticated, you have the power of the larger framework available to you.

Microsoft WebMatrix

- **Microsoft WebMatrix** is a free, lightweight, cloud-connected web development application for Windows.
- enables developers to build websites using built-in templates or popular open-source applications, with full support for ASP.NET, PHP, Node.js and HTML 5.
- developed for the purpose of providing web developers with coding, customization, and publishing capabilities all in one place.

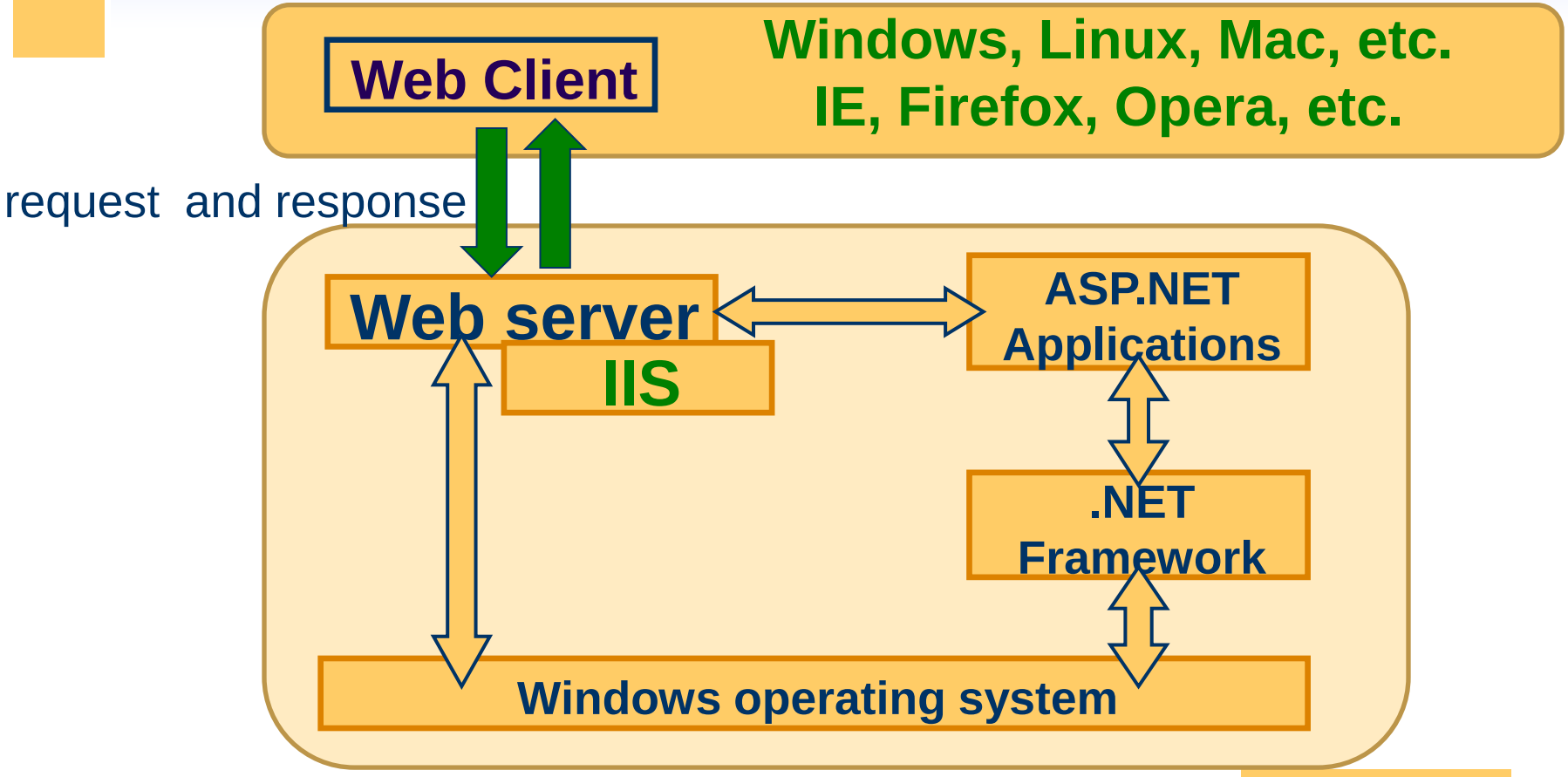
source: http://en.wikipedia.org/wiki/Microsoft_WebMatrix

ASP.NET Web Pages 2
Getting Started

- http://www.asp.net/web-pages/tutorials/introducing-aspnet-web-pages-2/getting-started

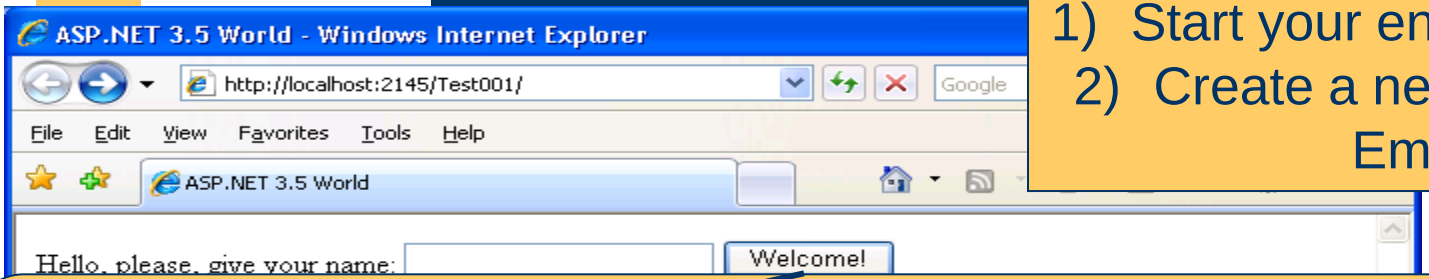
THE BIG PICTURE

The Big Picture

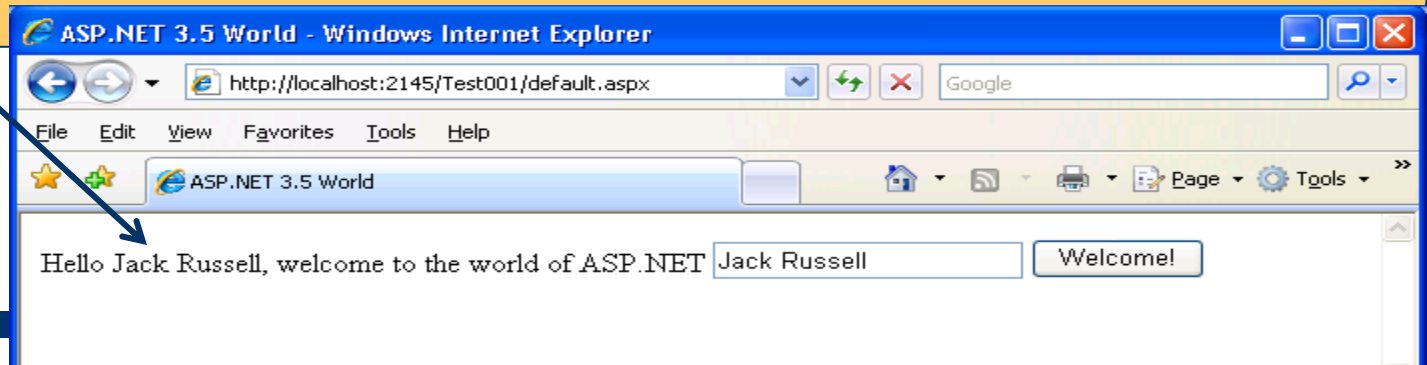


Demo: Default.aspx

1) Start your engine, please
2) Create a new Web Site Empty

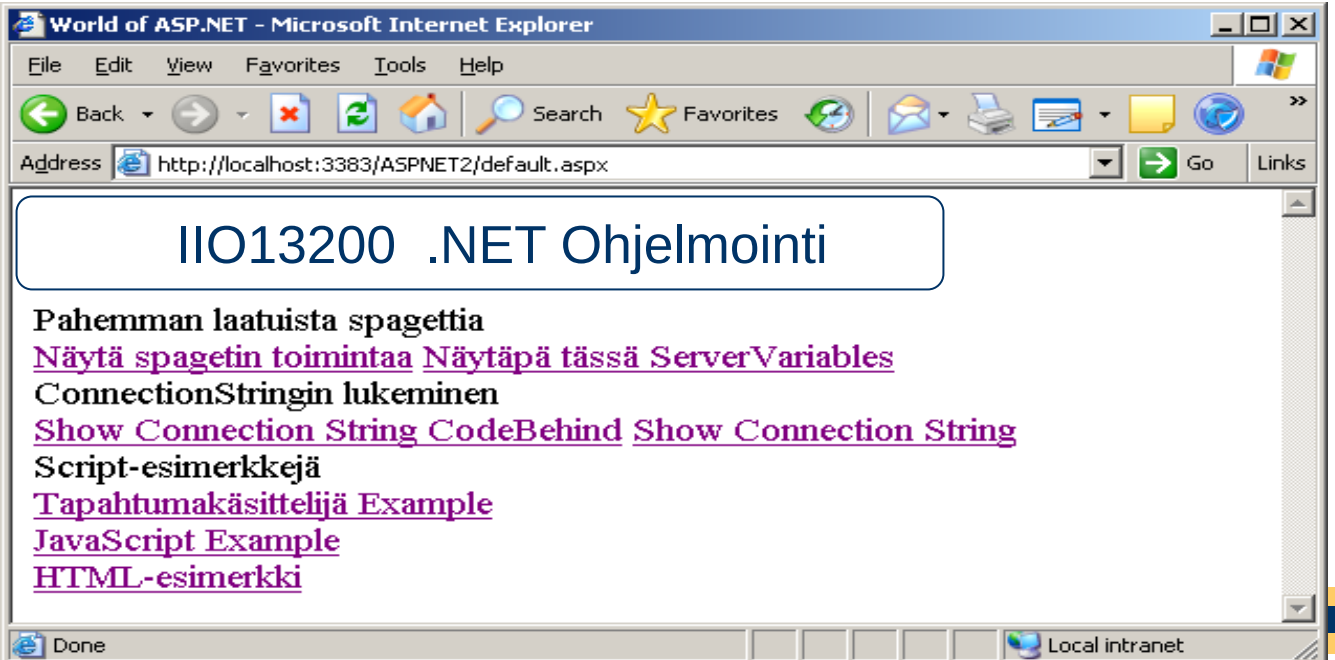



```
protected void btnWelcome_Click(object sender, EventArgs e)
  {
     if (txtName.Text.Length > 0)
        lblHello.Text = String.Format("Hello {0}, welcome to the world of ASP.NET", txtName.Text);
  }
```



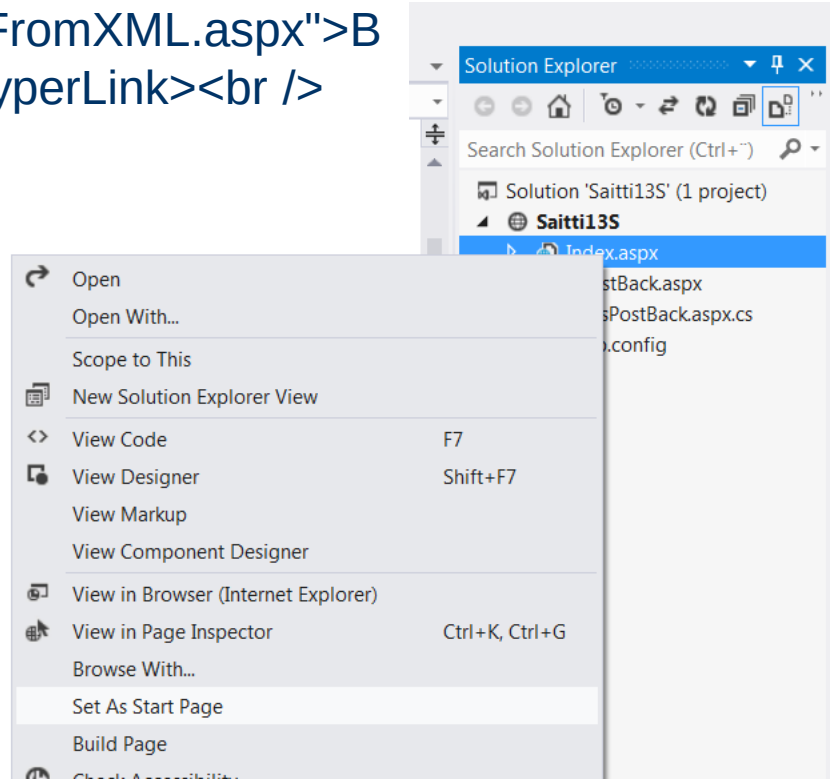
Demo: Index.html

- Tehdään pääsivu josta voi navigoida eri sivuille
  - voi olla pelkkää hötömölöä tai aspx.ää



Demo: Index.aspx

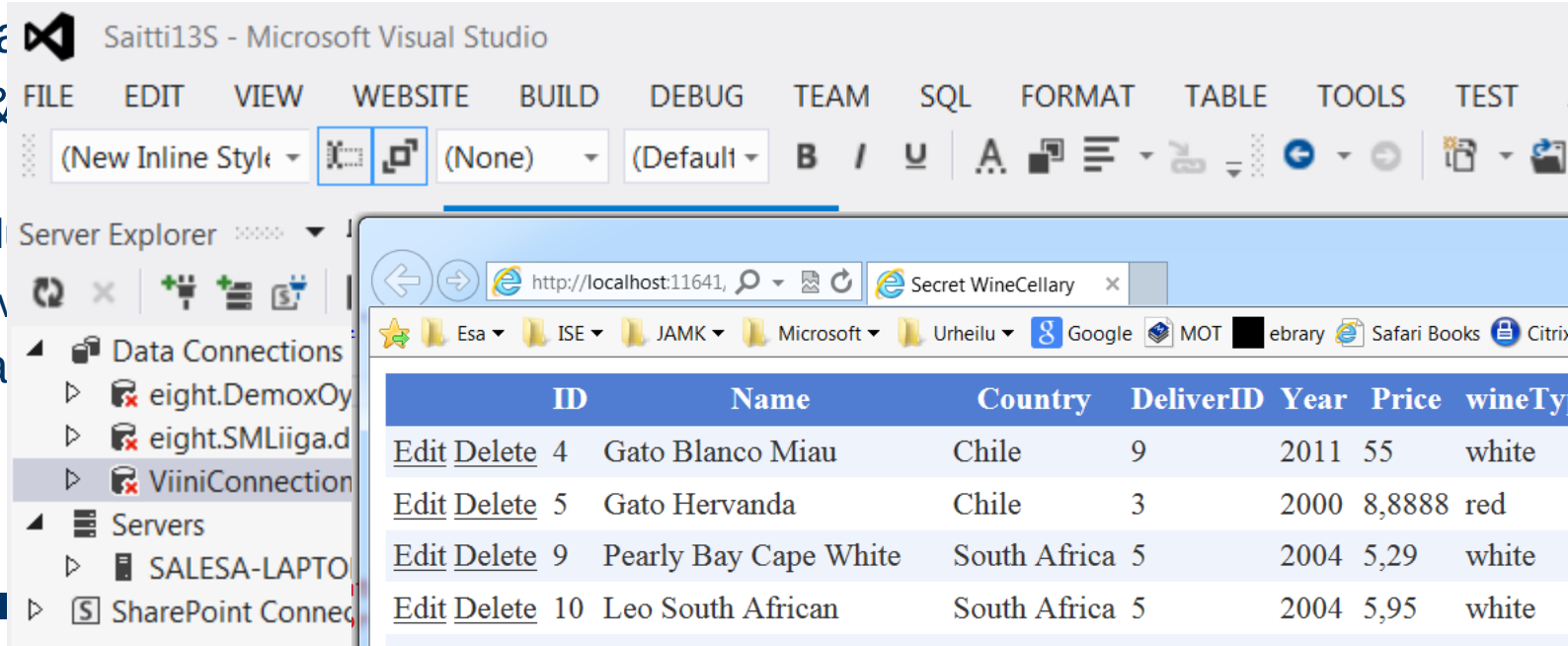

```
<asp:HyperLink ID="HyperLink2"
runat="server"
NavigateUrl="~/BooksFromXML.aspx">B
ooksFromXML</asp:HyperLink><br />
```



Demo: Viinikellari @ eight

Nopea demo tietokantapohjaisen web-sovelluksesta,

1. käynnistä VS
2. tee uusi Web site *File* | *New* | *Web site*
3. luodaa
4. Drag&
   → luo
   - hal
   - tarv
   - Ena



Background
Web Architecture

Client

PC/Mac/Unix/…
+ Browser

Request:
http://www.digimon.com/default.asp

Network

HTTP, TCP/IP

Response:
<html>….</html>

Server

Web Server

Background
Web Development Technologies

- Client-side technologies
  - HTML, DHTML, JavaScript

Tuttua?
→osana demoja

- Server-side technologies
  - PHP
  - JSP (JavaServer Pages)

Muilla kursseilla

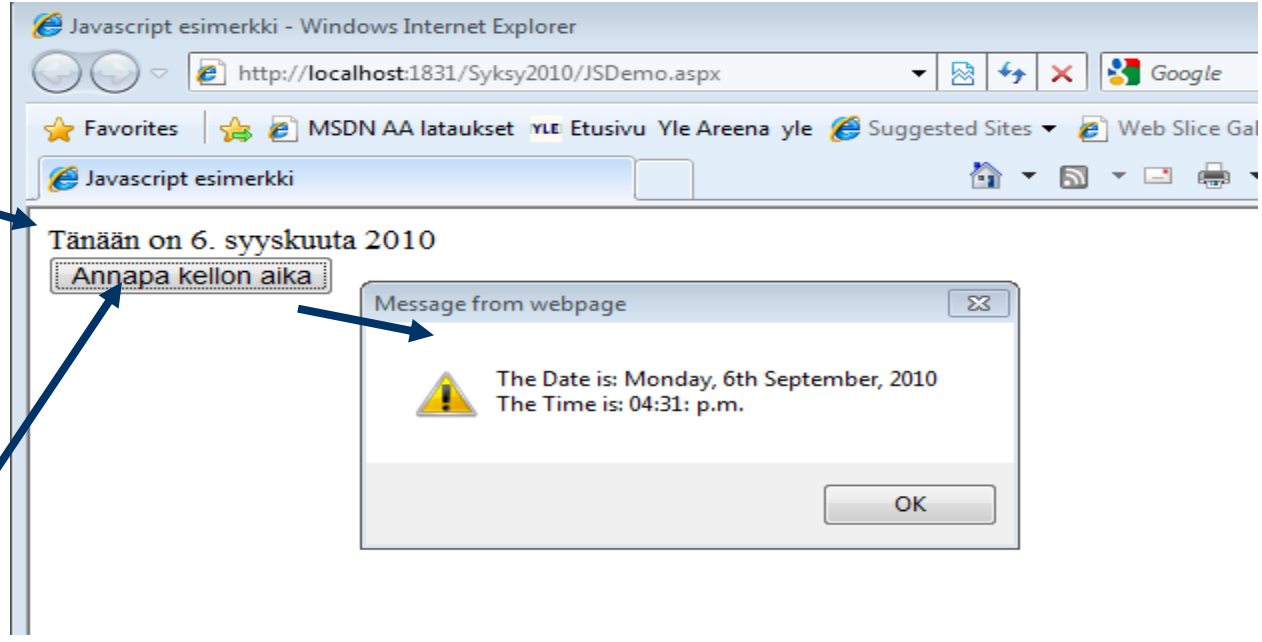
  - ASP (Active Server Pages)
  - ASP.NET is the next generation of ASP

Ei niin tuttua?
→ demoja

CLIENT-SIDE TECHNOLOGIES

Demo: JavaScript

Jsdemo.js



DateTimeDemo.js

```
<input id="Button1" type="button"
value="Annapa kellon aika" onclick="sivamtime()" />
```

Window.alert('Today is…

Demo: JavaScript

Jsdemo.js

```
<!-- JavaScript here  -->
<script language ="javascript">
var months=new Array(13);
months[1]="tammikuu";
months[2]="helmikuu";
months[3]="maaliskuu";
months[4]="huhtikuu";
months[5]="toukokuu";
months[6]="kesäkuu";
months[7]="heinäkuu";
months[8]="elokuu";
months[9]="syyskuun";
months[10]="lokakuu";
months[11]="marraskuu";
months[12]="joulukuu";
var time=new Date();
var lmonth=months[time.getMonth() + 1];
var date=time.getDate();
var year=time.getYear();
if (year < 2000)
  year = year + 1900;
document.write("<bold>Tänään on " + date + "
document.write(lmonth + "ta " + year + "</bo
</script>
```

DateTimeDemo.js

```
<HTML>
<TITLE>Hello javascript</TITLE>
<BODY>
<SCRIPT LANGUAGE="JavaScript1.2">
var months=new Array(13);
months[1]="January";
months[2]="February";
months[3]="March";
months[4]="April";
months[5]="May";
months[6]="June";
months[7]="July";
months[8]="August";
months[9]="September";
months[10]="October";
months[11]="November";
months[12]="December";
var time=new Date();
var lmonth=months[time.getMonth() + 1];
var date=time.getDate();
var year=time.getYear();
```

Javascript demo

- Tee javascript-funktio **inchestometers** joka muuttaa tuumat senteiksi

Demo: Data from SqlServer using only HTML

Buttonen

```
<asp:Button ID="Button1" runat="server"
Text="Show customers“  onclick="Button1_Click
/>
```

Paikka johon haun tulokset kirjoitetaan

```
<div runat="server" id="tulokset">...</div>
```

Kirjailijat haetaan kannasta ja kirjoitetaan suoraan HTMLään

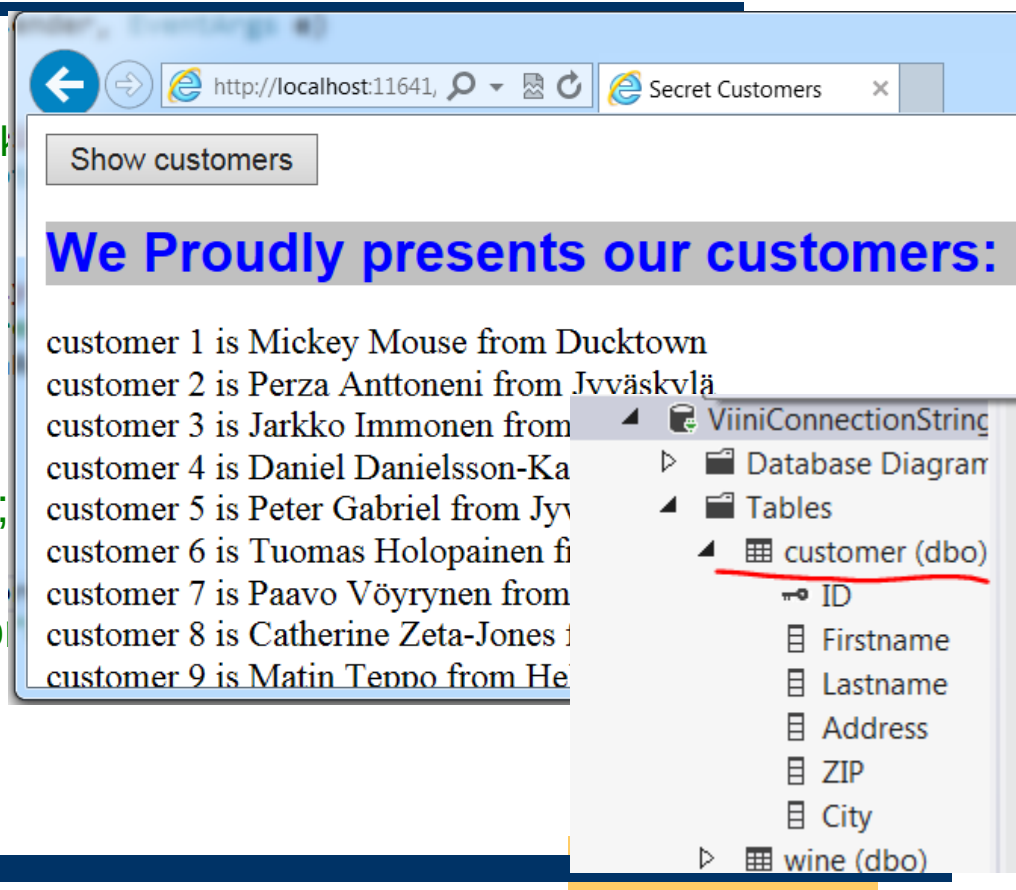
```
tulokset.InnerHtml =
"<h1>We Proudly presents authors:</h1>";
```

Käytä datareader-luokkaa ja Kirjoita

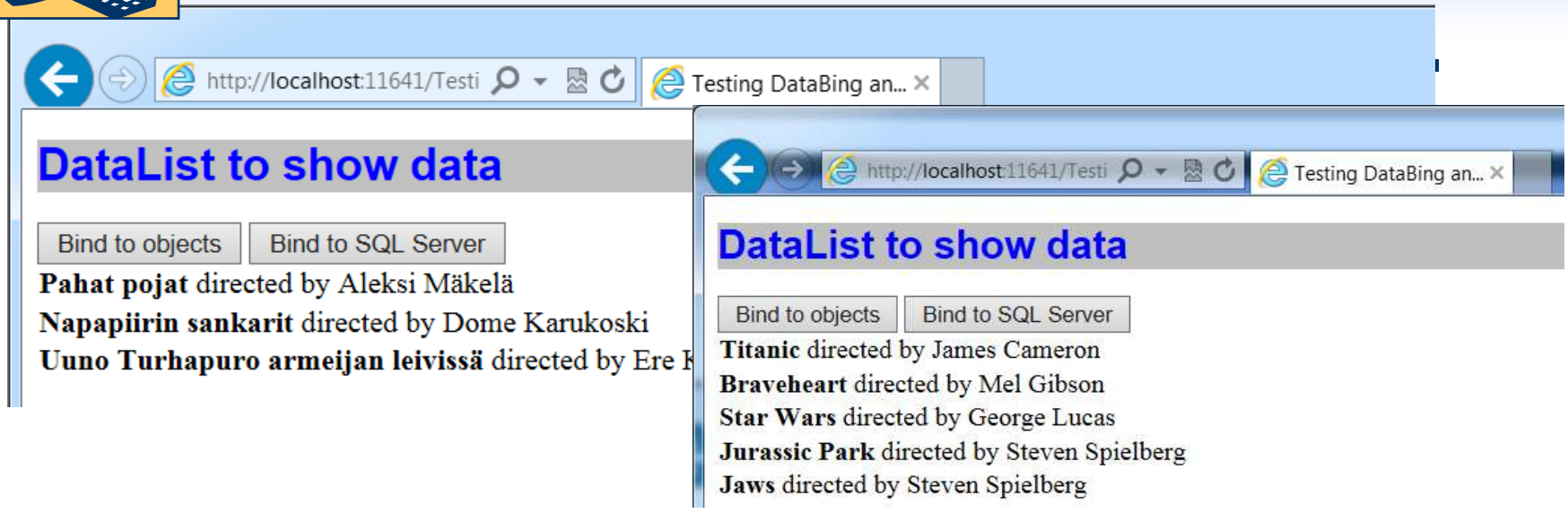
```
tulokset.InnerHtml += string.Format("autho
```

Muotoilut css:llä

```
<link href="Demo.css"
rel="Stylesheet"type="text/css" />
```



Demo: DataList, DataBind, Eval



- Tehdään yksinkertainen webbisivu, joka näyttää elokuva-data joko olio-kokoelmasta tai tietokannasta.

```
<asp:DataList ID="myDataList" runat="server">
    <ItemTemplate><b><%#Eval("Title") %> </b> directed by <%#Eval("Director") %></ItemTemplate>
</asp:DataList>
```

Agenda

- Background
- **ASP.NET Overview**
- Programming Model
- Programming Basics
- Server Controls
- Data Binding
- Advanced topics
- Conclusion

ASP.NET Overview

Goals

- Keep the good parts of ASP and improve the rest
- Simplify: less code, easier to create and maintain
- Multiple, compiled languages
- Fast
- Scalable
- Manageable
- Available
- Customizable and extensible
- Secure
- Tool support

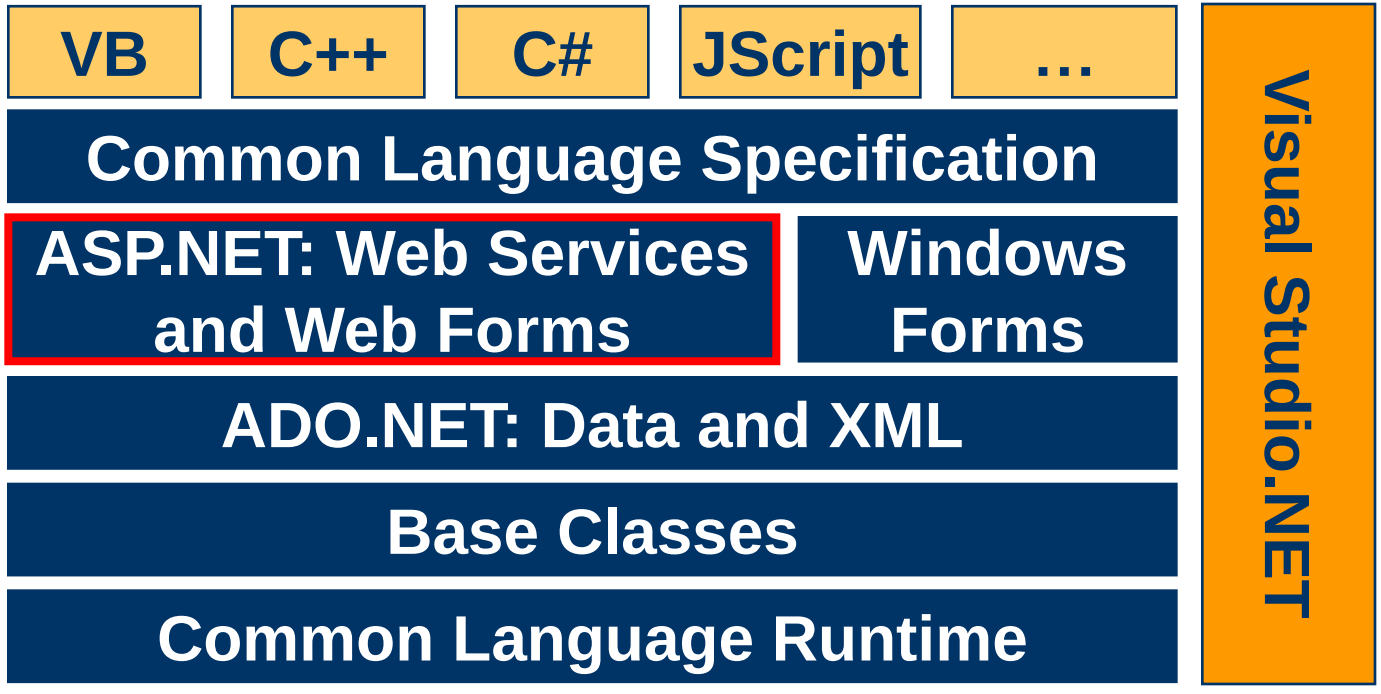
ASP.NET Overview

- ASP.NET provides services to allow the creation, deployment, and execution of Web Applications and Web Services
- ASP.NET is a server-side technology
- Web Applications are built using Web Forms
- Web Forms are designed to make building web-based applications as easy as building Visual Basic applications

ASP.NET ARCHITECTURE

ASP.NET Overview

Architecture



ASP.NET Overview
Architecture

- ASP.NET is built upon .NET Framework
  - **IIS** Internet Information Services (former Internet Information Server)
    It is the world's second most popular web server in terms of overall websites behind the industry leader Apache HTTP Server
    - Windows XP ➔ IIS 5.1
    - Windows Server 2003 ➔ IIS 6.0
    - Vista ja Windows Server 2008 ➔ IIS 7.0
    - Windows 7 ja Windows Server 2008 R2 ➔ IIS 7.5
    - Windows Server 2012 and Windows 8 ➔ IIS 8.0
  - The protocols supported in IIS:
    - FTP, FTPS, SMTP, NNTP, and HTTP/HTTPS.

ASP.NET Overview
Architecture

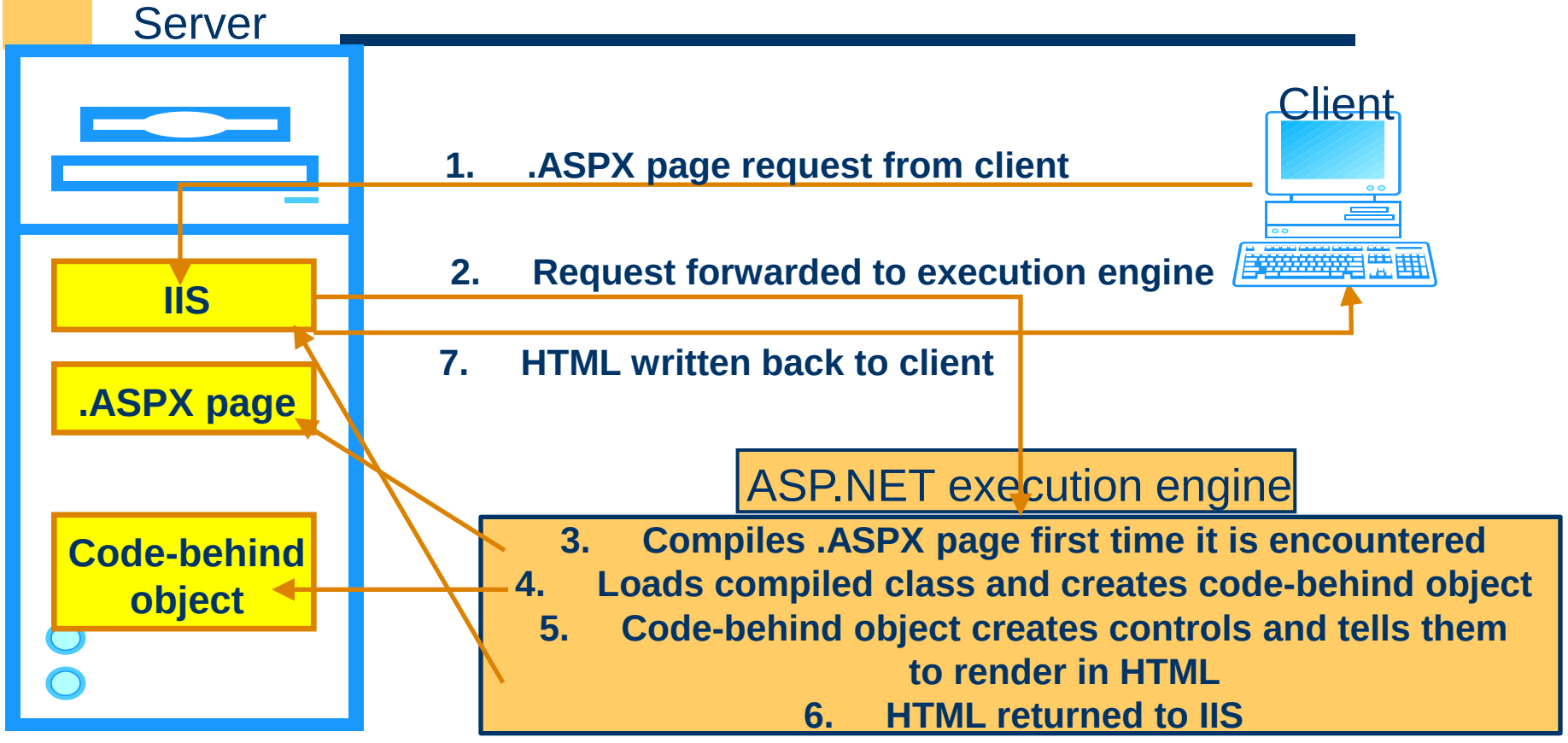
- Internet Information Service (IIS)
  - Tool to manage IIS: Internet Information Services Manager
  - Virtual Directories
    - Provides a mapping between URL and file path
    - E.g., on my machine the URL:

      `http://localhost/CS594`

      maps to the file path:

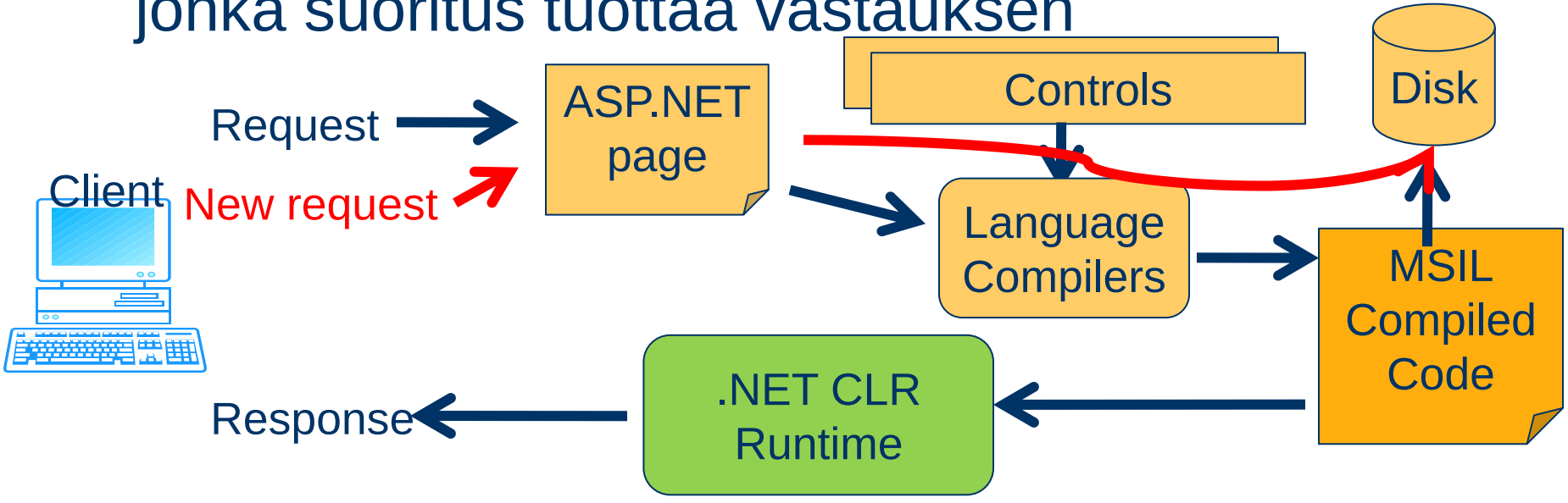
      `C:\_CS594Fall2001`

ASP.NET application model



ASP.NET toiminta suomeksi

- Pyyntöä vastaava sivu käännetään IL koodiksi, jonka suoritus tuottaa vastauksen



Käännös

- Kaikki sivut (*pages*) parseroidaan ja käännetään olioiksi
  - Web Forms sivut periytyvät Page-luokasta
  - Toiminnallisuus millä tahansa .NET-kielellä
- Just In Time käännös jos tiedoston sisältö muuttunut
  - Muuten otetaan valmiiksi käännetty sivu
- Sovelluksen sivut referoivat automaattisesti kaikki komponentit (DLL) jotka ovat sovelluksen **Bin** -alihakemistossa

Page class – Mother of all pages

- All the .aspx pages inherit from **Page** class
  - Includes methods, properties and events like any class
- Page declaration can include
  - Directives

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="_Default" %>
```

  - Web Server Control Syntax

```
<asp:Button ID="Button1" runat="server" Text=”Submit" />
```

  - HTML Server Control Syntax

```
<input id="Text1" type="text" runat="server" />
```

  - Code Declaration Blocks

```
<script runat="server">
void Button1_Click(Object sender, EventArgs e) {
    Label1.Text = "Clicked at " + DateTime.Now.ToString();
```

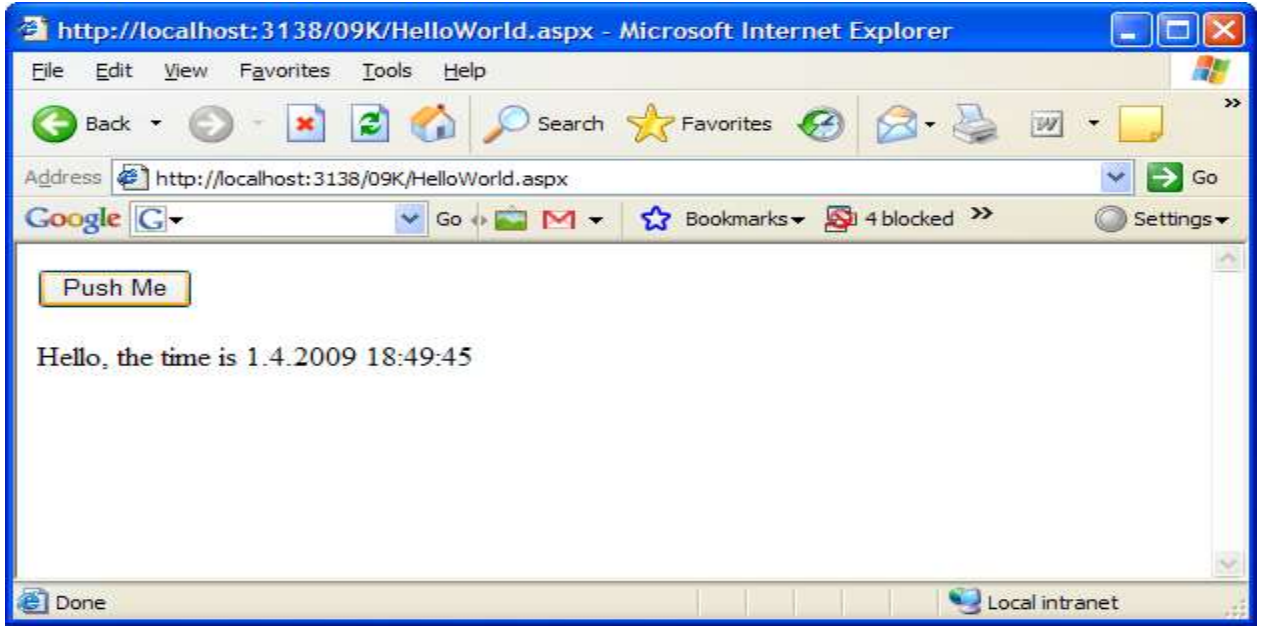
Programming Model
Code-behind pages

- Two styles of creating ASP.NET pages
  - Controls and code in .aspx file
  - Controls in .aspx file, code in **code-behind page**
    - Supported in Visual Studio.NET
- Code-behind pages allow you to separate the user interface design from the code
  - Allows programmers and designers to work independently

```
<%@ Codebehind="WebForm1.cs"
    Inherits=WebApplication1.WebForm1" %>
```

ASP.NET Overview

Demo: HelloWorld.aspx



ASP.NET Overview

Demo: HelloWorld.aspx

```
<%@ Page language="c#" %>
<html>
<head></head>
<script runat="server">
public void B_Click (object sender, System.EventArgs e) {
  Label1.Text = "Hello, the time is " + DateTime.Now;
}
</script>
<body>
  <form method="post" runat="server">
    <asp:Button onclick="B_Click" Text="Push Me"
      runat="server" /> <p>
    <asp:Label id=Label1 runat="server" />
  </form>
</body>
```

ASP.NET Overview

Demo: HelloWorld2.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="HelloWorld2.aspx.cs"
Inherits="HelloWorld2" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server"><title>Hello again</title></head>
<body>
    <form id="form1" runat="server">
    <div>
    <asp:Button ID="Button1" onclick="B_Click"
Text="Push I
    <asp:Lal
    </div>
    </form>
```

In codebehind file HelloWorld2.aspx.cs

```
...
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
public partial class HelloWorld2 : System.Web.UI.Page{
```

Page lifespan

PreInit

Page_Init

InitComplete

Control instantiation

PreLoad

Page_Load

txtTextBox_Changed

btnButton_Click

LoadComplete

PreRender

Page_PreRender

Saving the state of controls

SaveStateComplete

Rendering

Page_Unload

Video: Page Lifecycle Events

http://www.asp.net/general/videos/page-lifecycle-events

ASP.NET Overview

Key Features

- Web Forms
- Web Services
- Built on .NET Framework
- Simple programming model
- Maintains page state
- Multibrowser support
- XCOPY deployment
- XML configuration
- Complete object model
- Session management
- Caching
- Debugging
- Extensibility
- Separation of code and UI
- Security
- ASPX, ASP side by side
- Simplified form validation
- Cookieless sessions

Agenda

- Background
- ASP.NET Overview
- **Programming Model**
- Programming Basics
- Server Controls
- Data Binding
- Advanced topics
- Conclusion

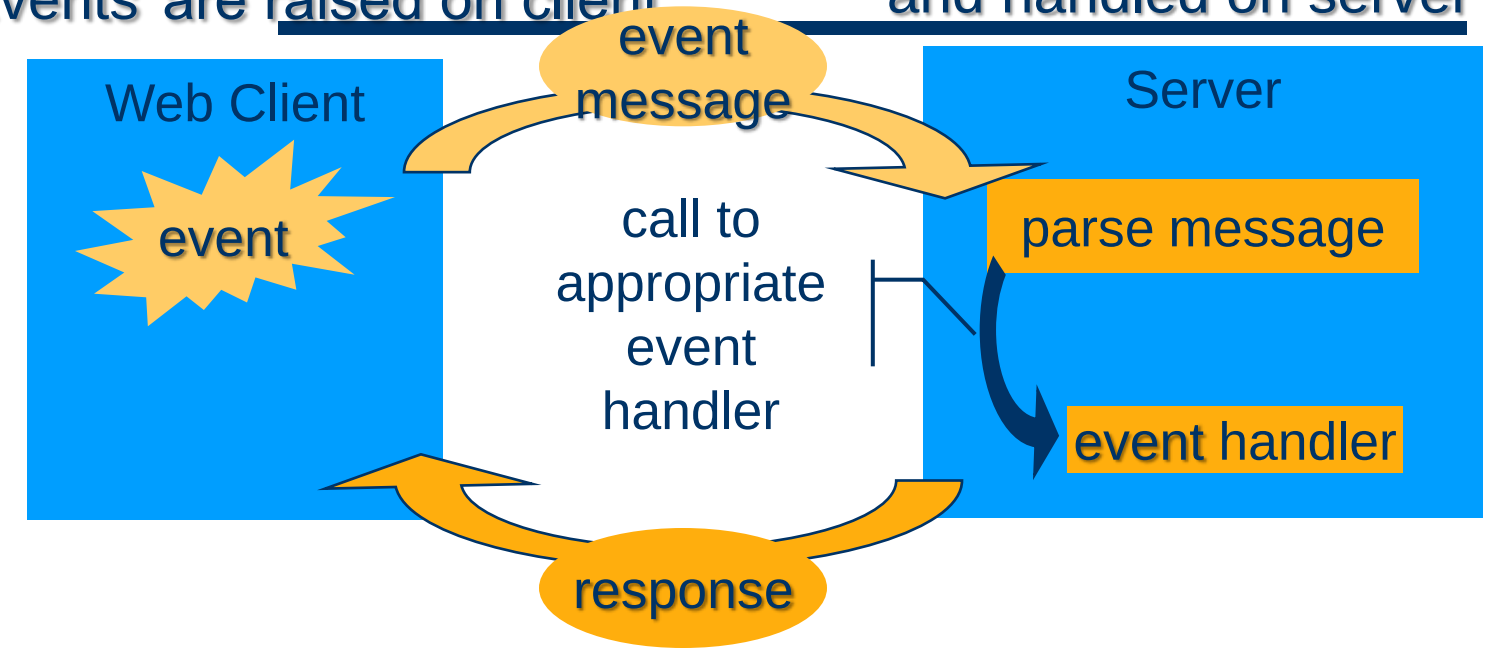
SERVER SIDE PROGRAMMING MODEL

Programming Model
Controls and Events

- Server-side programming model
- Based on controls and events
  - Just like Visual Basic
  - Not “data in, HTML out”
- Higher level of abstraction than ASP
- Requires less code
- More modular, readable, and maintainable

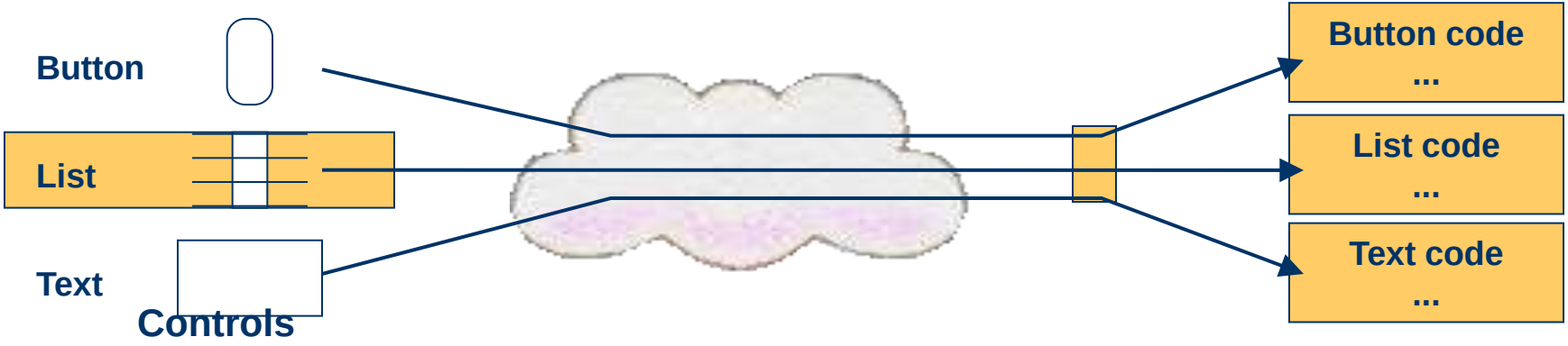
ASP.NET event handling model

Events are raised on client and handled on server



- Handling similar to that of client applications..... almost
  - Events are raised on client and event message is transmitted to the server through an HTTP post
- Round trip to server causes a performance penalty
  - Unintentional events (like MouseMove) are not supported

Programming Model
Controls and Events



Browser

ASP.NET

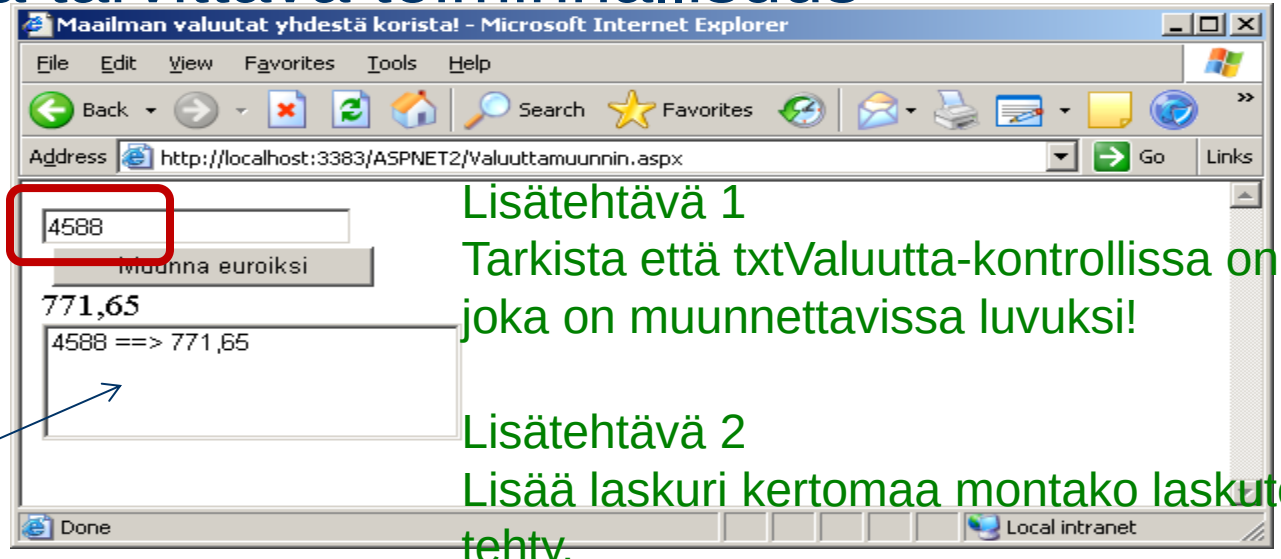
Event handlers

Kontrollien arvot säilyvät

- Server-kontrollit osaavat automaattisesti asettaa arvonsa sivun vastaanottamisen (formin data) ja lähettämisen yhteydessä
  - Ei tarvita lainkaan koodia arvojen kierrättämiseen!
- Ei tilan säilyttämistä palvelimessa
  - Perustuu Formin Hidden-kenttään (_ViewState)
  - Toimii kaikkien selaimien kanssa
  - Tilanhallintaa voidaan ohjata myös kontrollikohtaisesti EnableViewstate ominaisuuden avulla

Demo: Valuuttamuunnin1

- Luo oheinen UI ja koodaa tarvittava toiminnallisuus



txtValuutta
btnMuunna
lblTulos
lstTulos

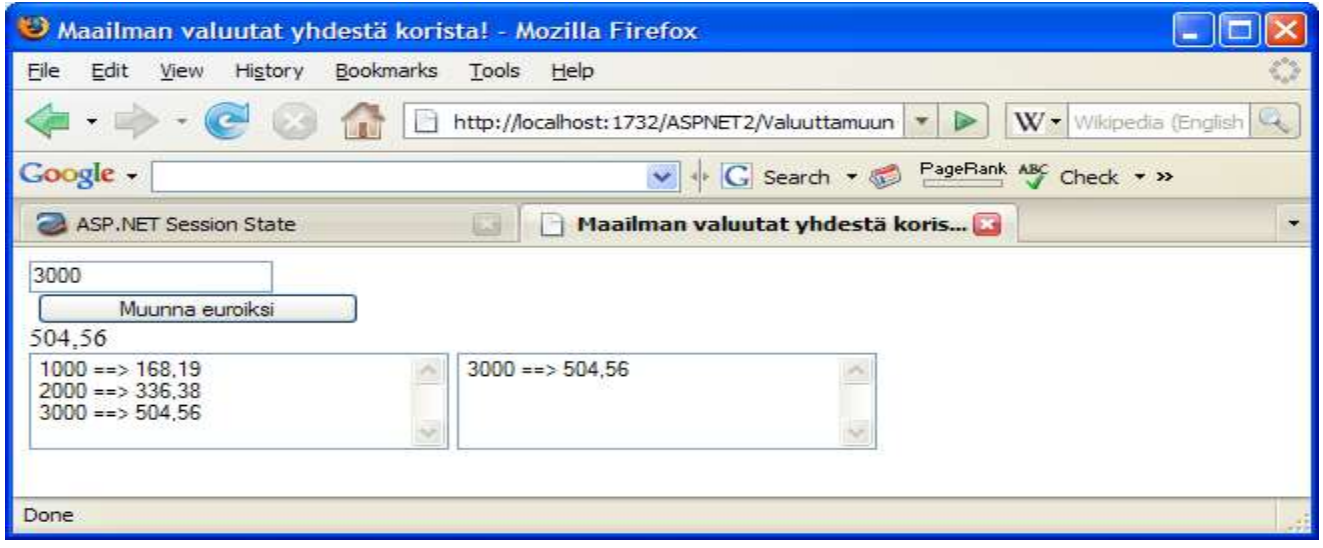
Lisätehtävä 1
Tarkista että txtValuutta-kontrollissa on teksti joka on muunnettavissa luvuksi!

Lisätehtävä 2
Lisää laskuri kertomaa montako laskutoimitusta tehty.

Muistaako kontrolli edelliset laskelmat automaattisesti

Demo: Programming Basics
Maintaining State

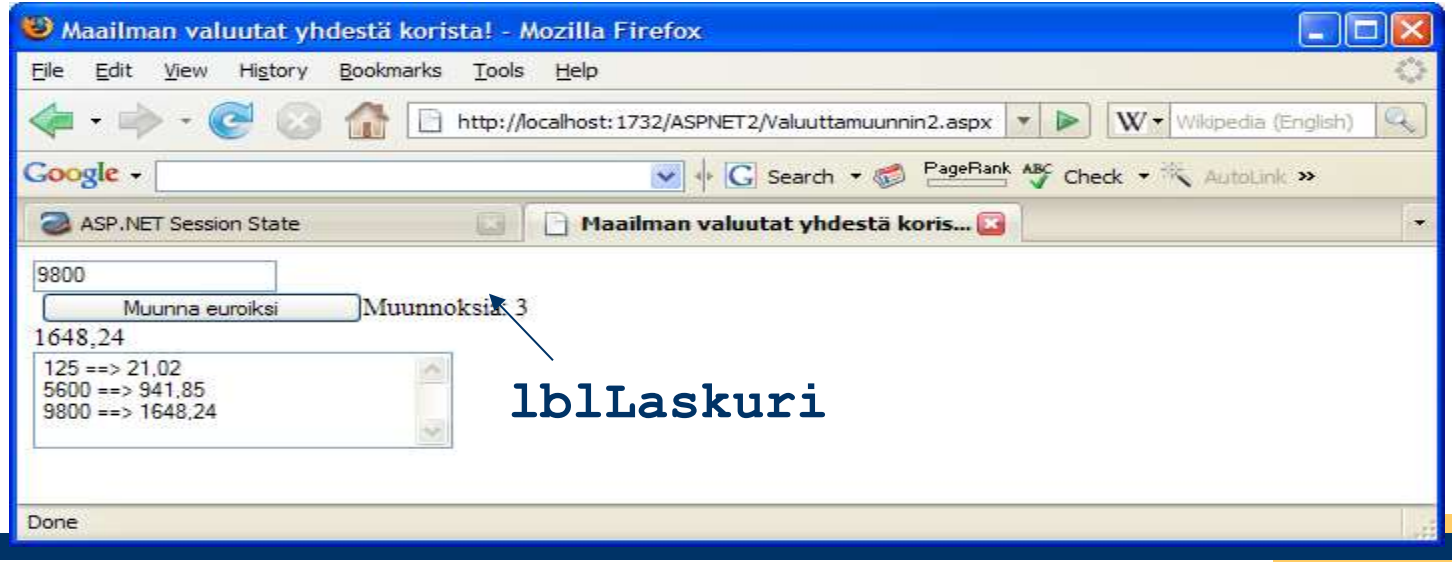
- Demo: Valuuttamuunnin1.aspx
  - *Default* EnableViewState = True



Tutki miten toteutettu?

Demo: Valuuttamuunnin2

- Tee lisäys sovellukseen valuuttamuunnin
  - 1) niin että lasket montako kertaa käyttäjä on tehnyt (onnistuneita) valuutanmuunnoksia
    - → luo sopiva muuttuja joka laskee muunnokset ja ilmoittaa koko ajan sen sivulla.
  - 2) Eka kerran kun sivulle tullaan niin syöttökentän oletusarvona 100



A FAQ…

- Frequently Asked Question:

  How to Redirect Users to Another Page
- http://msdn.microsoft.com/en-us/library/x3x8t37x.aspx
  - `Server.Transfer("Valuuttamuunnin3.aspx");`

Redirecting Users to Another Page

- Two ways to send the user to another page:
- A1: Standard redirection

```
Response.Redirect(”UusiSivu1.aspx");
```

- A2: Transfer the processing of the current request to another page without notifying the user

```
Server.Transfer(”UusiSivu2.aspx");
```

http://msdn.microsoft.com/en-us/library/x3x8t37x.aspx

Demo: Transfer&Redirect

Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:55149/Index.aspx

Most Visited Getting Started Latest Headlines Customize Links

Response.Redirect | Server.Transfer

Done

Tämä on UusiSivu1 - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:55149/UusiSivu1.aspx

Most Visited Getting Started Latest Headlines Customize Links

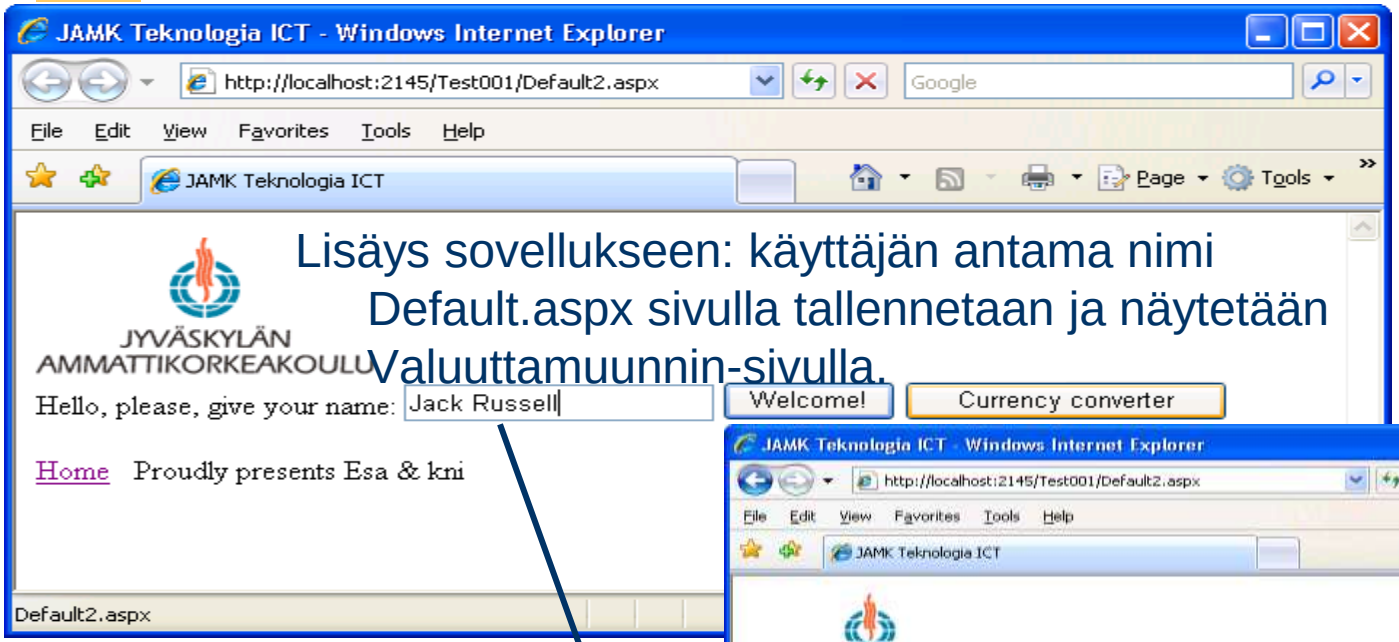
Tämä on UusiSivu2 - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:55149/Index.aspx

Most Visited Getting Started Latest Headlines Customize Links

Demo: Valuuttamuunnin3



Lisäys sovellukseen: käyttäjän antama nimi Default.aspx sivulla tallennetaan ja näytetään Valuuttamuunnin-sivulla.

Demo: Valuuttamuunnin3a

To 8.9.2011

Lisäys sovellukseen: käyttäjän antaa nimensä Index.aspx sivulla, annettu nimi tallennetaan ja näytetään Valuuttamuunnin-sivulla.

Kirjailijat Priex-palvelimelta

Seuraava sovellus vain tunnetuille käyttäjille Valuuttamuunnin

Programming Model
ASP.NET Object Model

- User code executes on the web server in page or control event handlers
- Controls are **objects**, available in server-side code
  - Derived from `System.Web.UI.Control`
- The web page is **an object** too
  - Derived from `System.Web.UI.Page` which is a descendant of `System.Web.UI.Control`
  - A page can have methods, properties, etc.

Programming Model
Postback

- A postback occurs when a page generates an HTML form whose values are posted back to the same page
- A common technique for handling form data
- In ASP and other server-side technologies the state of the page is lost upon postback...
- Unless you explicitly write code to maintain state
- This is tedious, bulky and error-prone

Programming Model
Postbacks Maintain State

- By default, ASP.NET maintains the state of all server-side controls during a postback
- Can use `method="post"` or `method="get"`
- Server-side control objects are automatically populated during postback
- No state stored on server
- Works with all browsers

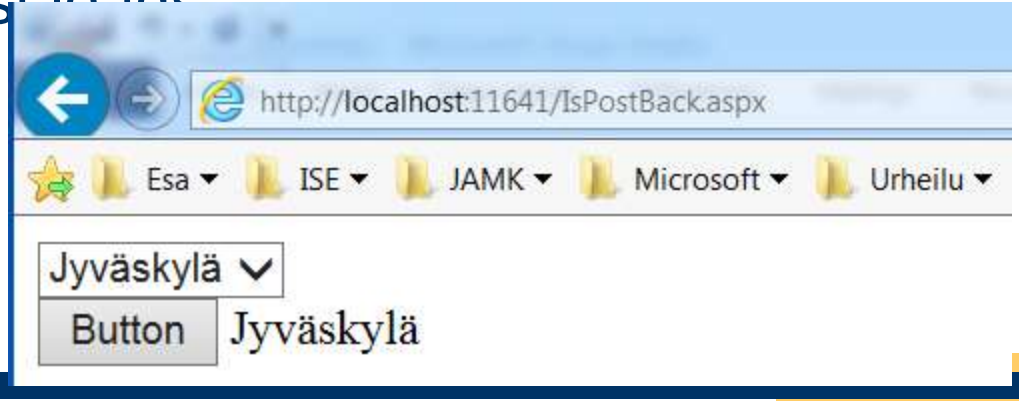
Vertaa kalvo 43

```
//koodissa
if (!IsPostBack){}
```

Demo: Testaa Valuutanmuunnin.aspx

DEMO: Using Page.IsPostBack

- The Page class has IsPostBack –property, used to detect whether the page has been already posted back to server
- we want not to initialize the property every time a a page loads. we typically initialize a control property only once when the page first loads



Programming Model
Server-side Controls

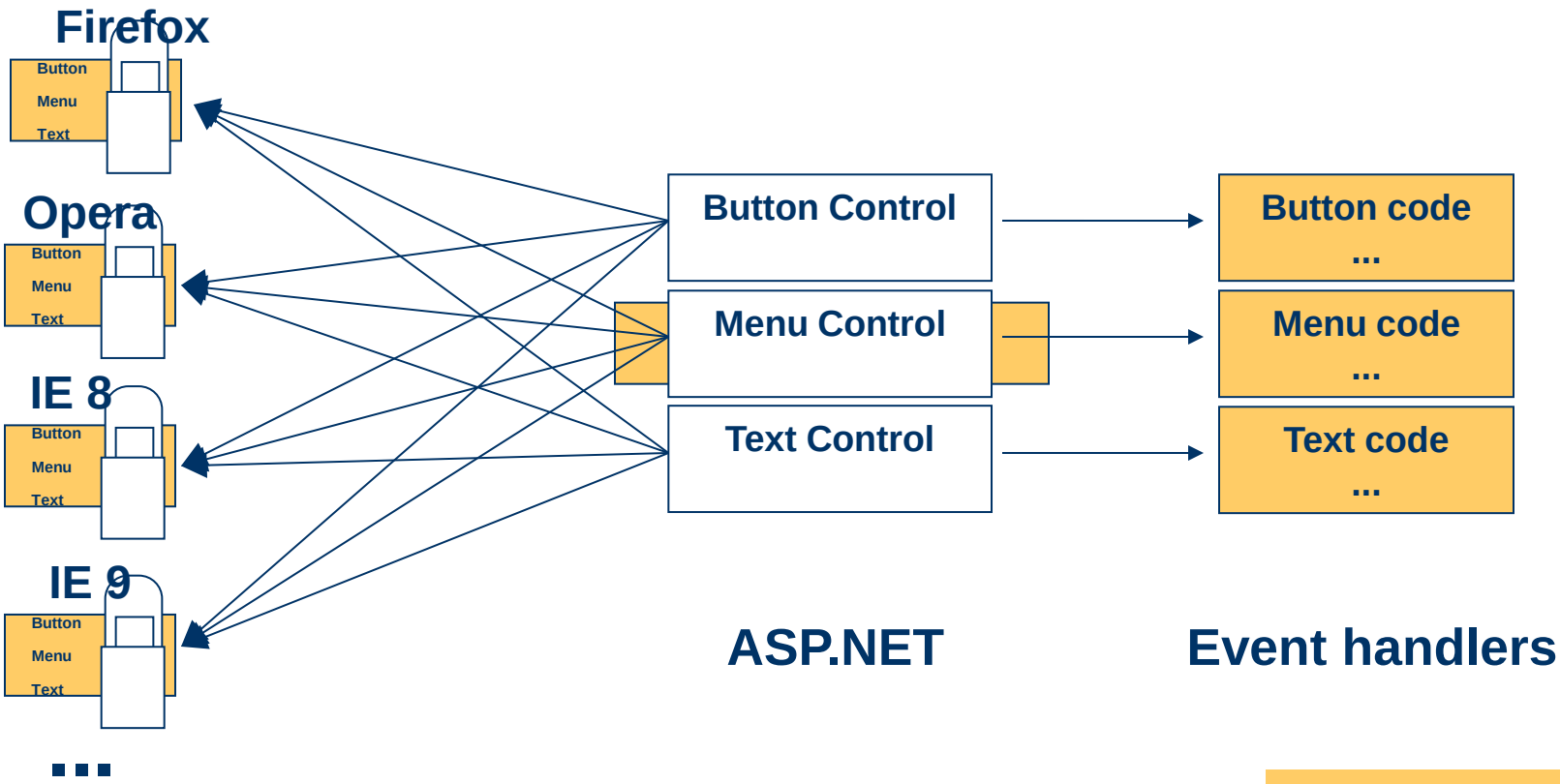
- Multiple sources of controls
  - Built-in
  - 3rd party, for example devexpress.com
  - User-defined
- Controls range in complexity and power: button, text, drop down, calendar, data grid, ad rotator, validation
- Can be populated via data binding

Programming Model
Automatic Browser Compatibility

- Controls can provide automatic browser compatibility
- Can target `UpLevel` or `DownLevel` browsers
  - `UpLevel` browsers support additional functionality, such as JavaScript and DHTML
  - `DownLevel` browsers support HTML 3.2

Programming Model

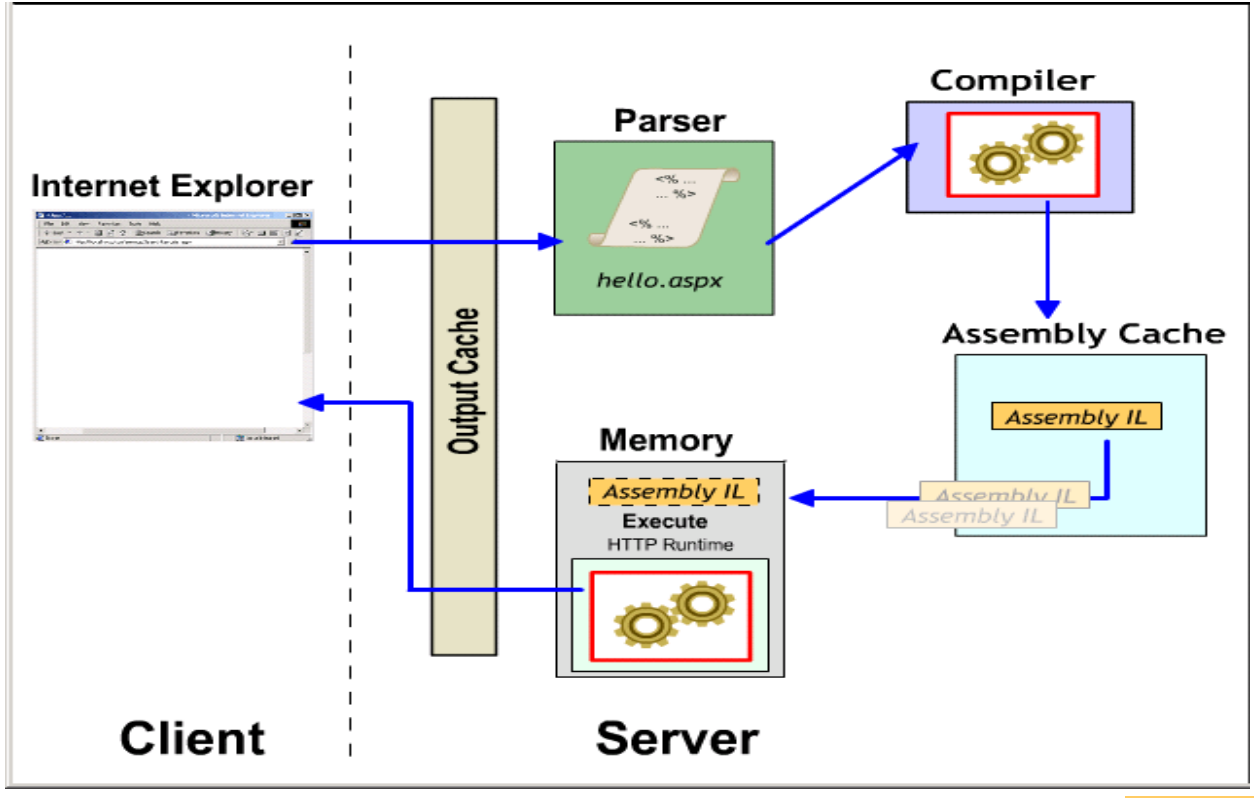
Automatic Browser Compatibility



Programming Model
Automatic Compilation

- Just edit the code and hit the page
- ASP.NET will automatically compile the code into an assembly
- Compiled code is cached in the CLR Assembly Cache
- Subsequent page hits use compiled assembly
- If the text of the page changes then the code is recompiled
  - Works just like ASP: edit, save and run

Programming Model
Automatic Compilation



OUTLOOK OF PAGES

Sivujen ulkoasu
Setting outlook of pages

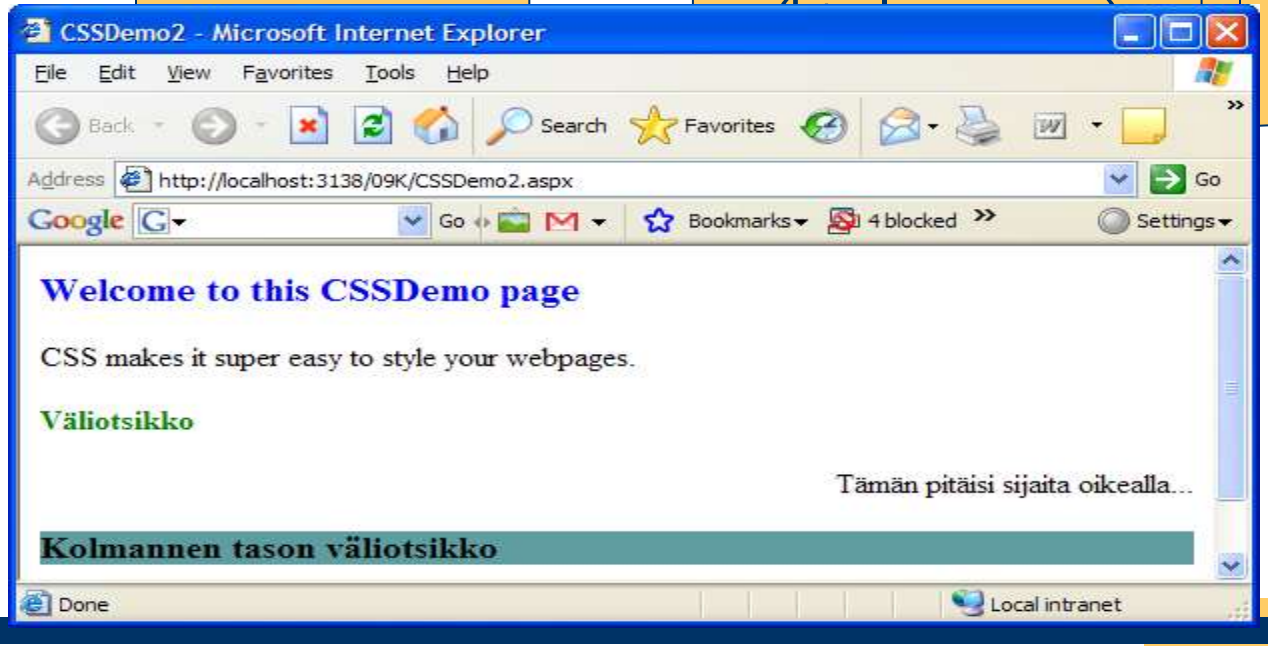
- Erilaisia tapoja yhtenäistää sivujen esitystapaa, ulkoasua ja toiminnallisuutta:
  - CSS
  - Master Pages
  - Themes
  - Olio-ohjelmoinnin keinot: Periyttäminen

Demo: CSSDemo.aspx

- CSS is well supported now!

StyleSheet.css

Bunch of webpages



Demo: CSSDemo.aspx

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
 <title>CSSDemo2</title>
 <link href="Demo.css" rel="Stylesheet" type="text/css" />
</head>
<body>
...
```

Demo.css

```
h1
{
  color:Blue;
  font-size:20px;
}
H2
...
```

mo: CSSDemo.aspx jatkuu

CSS-tiedoston muuttaminen MasterPagella muuttaa ulkoasun

```
<link href="Demo.css" rel="Stylesheet" type="text/css" />
```

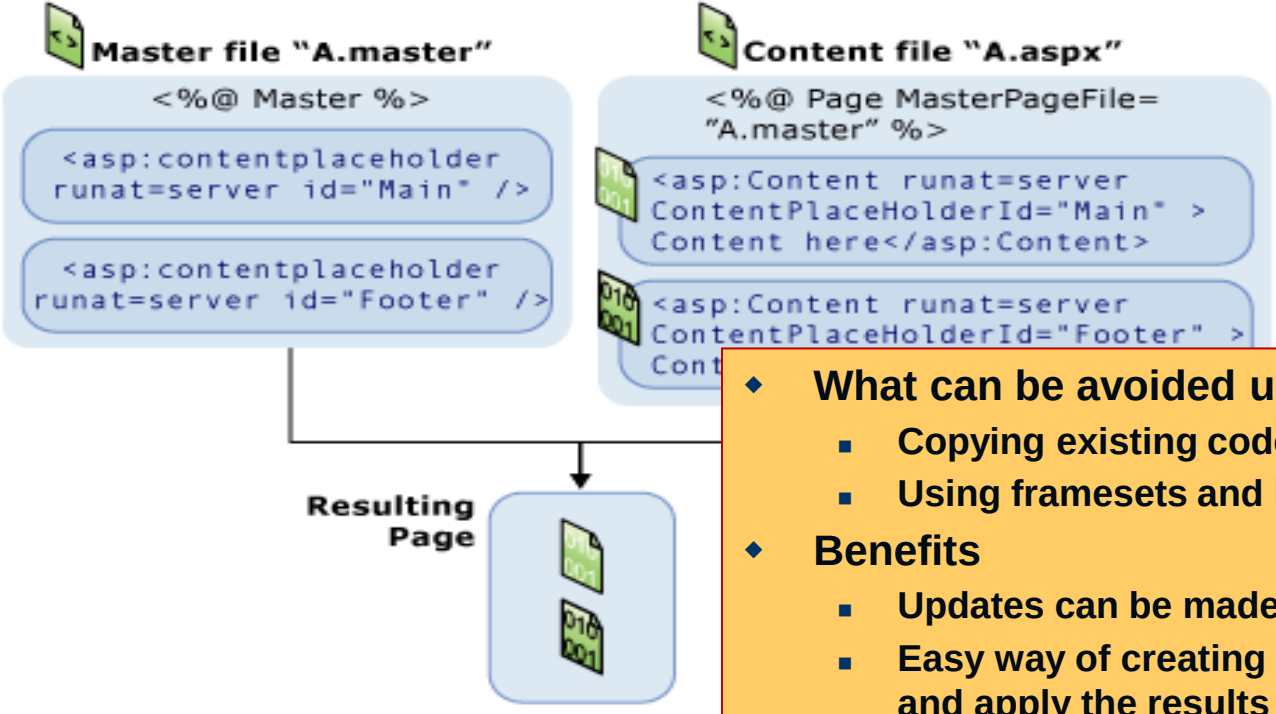

```
link href="siteDemo.css"
```

Master Pages

- ASP.NET master pages allow you to create a consistent layout for the pages in your application.
- A single master page defines
  1) **the look and feel** and
  2) **standard behavior**
  that you want for all of the pages (or a group of pages).

http://msdn.microsoft.com/en-us/library/wtxbf3hh%28v=vs.100%29.aspx

Master pages



- **What can be avoided using master pages**
  - **Copying existing code**
  - **Using framesets and include files**
- **Benefits**
  - **Updates can be made in just one place**
  - **Easy way of creating one set of controls and code and apply the results to a set of pages**
  - **Underlying object model allows the master page to be customized from individual content pages**

Demo:MasterPage.Master

- Luo MasterPage.Master
- Lisää sille haluamasi kuva tai JAMKin logo storagen Jakoon\iio13200
- Tallenna Masterpage
- Luo uusi sivu, johon määrittelet että se käyttää luomaasi Masterpagea.

Public Property in the MasterPage

- Content pages can reference any public property declared in the master page code-behind file.
  - the following code sample defines the property SharedInfo in the MasterPage page code-behind file. The master page provides strong typing for the session variable, SharedInfo.

```
public String SharedInfo
{
  get { return (String)Session["SharedInfo"]; }
  set { Session["SharedInfo"] = value; }
}
```

and at the page where we use SharedInfo we add a directive:

```
<%@ MasterType VirtualPath="~/MasterPage.master" %>
```

Teemat (Themes)

- An ASP.NET Theme enables you to apply a consistent style to the pages in your website.
- A theme is a collection of property settings
- A Theme can control the appearance of:
  - HTML elements
  - ASP.NET controls

http://msdn.microsoft.com/en-us/library/ykzx33wh%28v=vs.100%29.aspx

Master Pages vs Themes

- A Master Page enables you to share content (and functionality) across multiple pages
- **A Theme** enables you to control the appearance of the content

ASP.NET Themes and Skins

- A theme is a collection of property settings that allow you to define the look of pages and controls, and then apply the look consistently across pages in a Web application, across an entire Web application, or across all Web applications on a server.
- Themes are made up of a set of elements skins, cascading style sheets (CSS), images, and other resources. At a minimum, a theme will contain skins.
  - Themes are defined in special directories in your Web site or on your Web server.
- Read more: http://msdn.microsoft.com/en-us/library/ykzx33wh.aspx

Demo: TestaaTeema

Teema-kansio: Punainen

Outlook of controls

RedStyleSheet.css

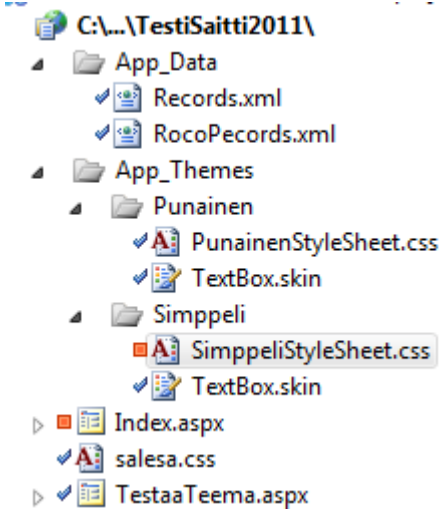
Pictures

Teema-kansio: Simppeli

Outlook of controls

SimpleStyleSheet.css

Pictures



Defines outlook of pages, controls, pictures

WEB APPLICATION PROJECTS VS WEB SITE PROJECTS

Web Application Projects versus Web Site Projects

- In Visual Studio you can create: *Web application projects* or *Web site projects*.
- Each type of project has advantages and disadvantages
- **Select** the appropriate project type **before** you create a project, because it is not practical to convert from one project type to the other.

http://msdn.microsoft.com/en-us/library/dd547590.aspx

Web Applications project preferred

- For ”bigger” projects, gives more possibilites
- Project file structure
  - A Visual Studio project file (.csproj or .vbproj) stores information about the project, such as the list of files that are included in the project, and any project-to-project references.
- Compilation
  - You explicitly compile the source code on the computer that is used for development or source control.

Web Sites projects preferred

- For smaller or medium project, simpler
- Project file structure
  - There is no project file (.csproj or .vbproj). All the files in a folder structure are automatically included in the site.
- Compilation
  - By default, compilation of code files (excluding .aspx and .ascx files) produces a single assembly.

ASP.NET
PROTECTED FOLDERS

ASP.NET Special Protected Folders

- ASP.NET uses a number of special directories below the application root to maintain application content and data.
  - Earlier in ASP.NET 1.x there was only Bin –folder
  - From 2.x -> additional protected directories
- None of these folders are automatically created by ASP.NET or VS, nor are the directories necessarily required to exist → Each directory needs to created either manually or on demand through of VS feature

ASP.NET Special Protected Folders

Directory	Intended Goal
Bin	For precompiled assemblies
App_Browsers	Browser capability information
App_Code	Source class files (vb or c#) used by pages, all the files must be in same language
App_Data	Data files for the application (xml, mdb, sql…
App_GlobalResources	.resx resource files global to the application
App_LocalResources	All .resx resource files that are specific to a particular page
App_Themes	Secu… est
App_WebReferences	

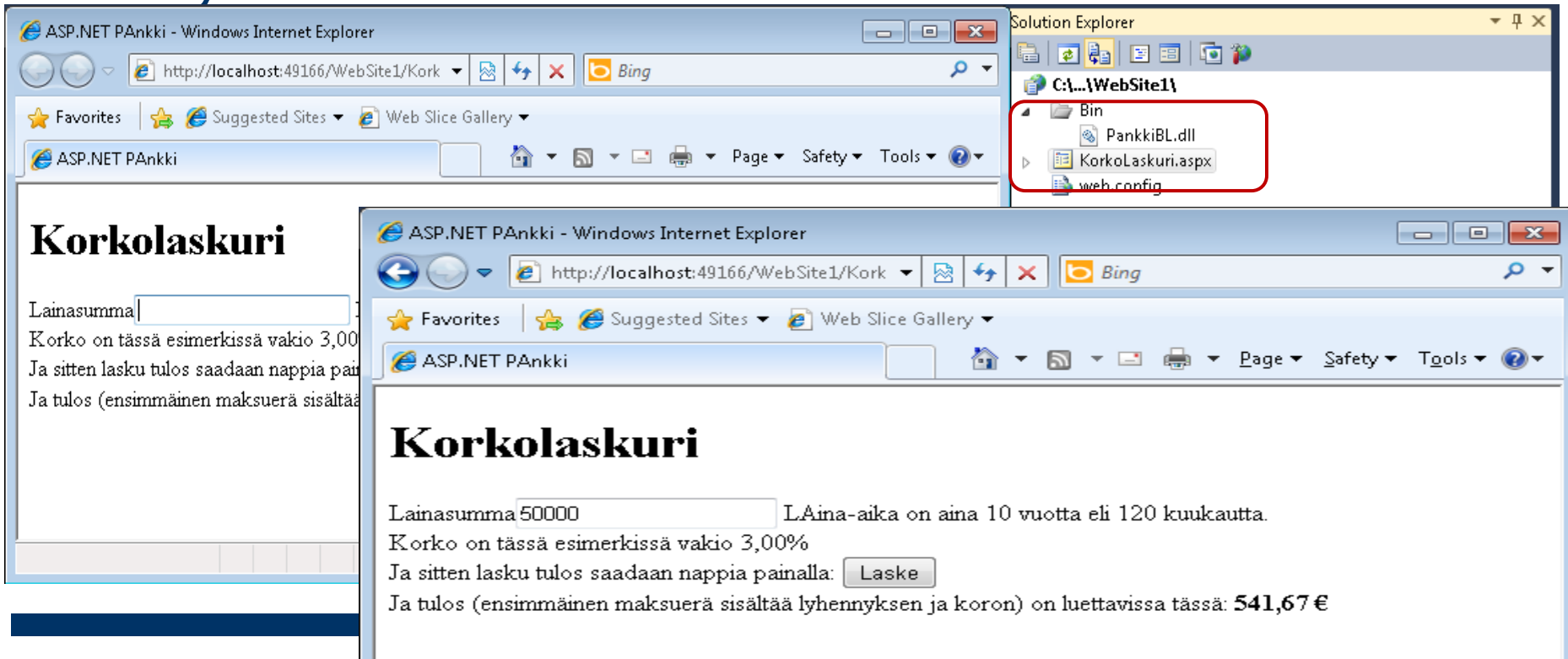
Info: The names of Folders are not customizable

The Bin directory
Valmiiden .NET komponenttien käyttö

- Kaikki komponentit virtuaalihakemistossa **Bin** ovat automaattisesti referoituja.
- Käännetyille .NET-komponenteille.
- Komponenteista tehdään ns *shadow copy*, jolloin komponenttitiedostot eivät ole varattuina vaan niitä voidaan päivittää kopioimalla päälle

Demo: Korkolaskuri.aspx Valmiin DLL:n käyttö

- Käytetään valmista PankkiBL.DLL:ää…

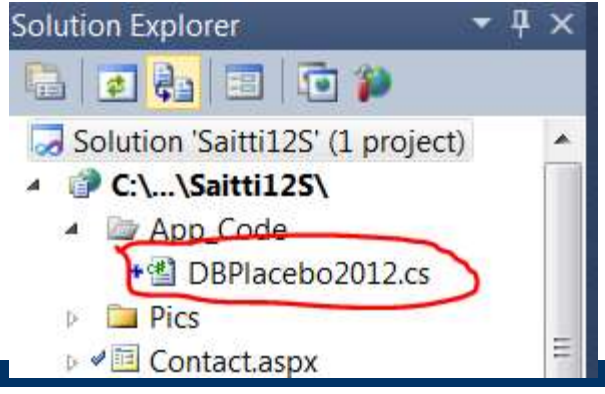


The App_Code directory

- App_Code directory is used to group your helper and business classes.
- Deploy as source files, and ASP.NET runtime ensures that **classes will be automatically compiled on demand** → the resulting assembly is automatically referenced
- Put only components int the App_Code directory, DO NOT put pages, Web user controls

Demo: NaytaOppilaat.aspx App_Coden käyttö

- Kopioi App_Code kansioon DBPlacebo2012.cs → käytä sen staattista metodia **Get3Students**



The App_Data directory

- Data files for the application
- A Reference to a file in code

```
String connStr = "Provider=Microsoft.ACE.OLEDB.12.0;"
  + "Data Source=" + Server.MapPath("~/App_Data/Placebo.accdb")
```

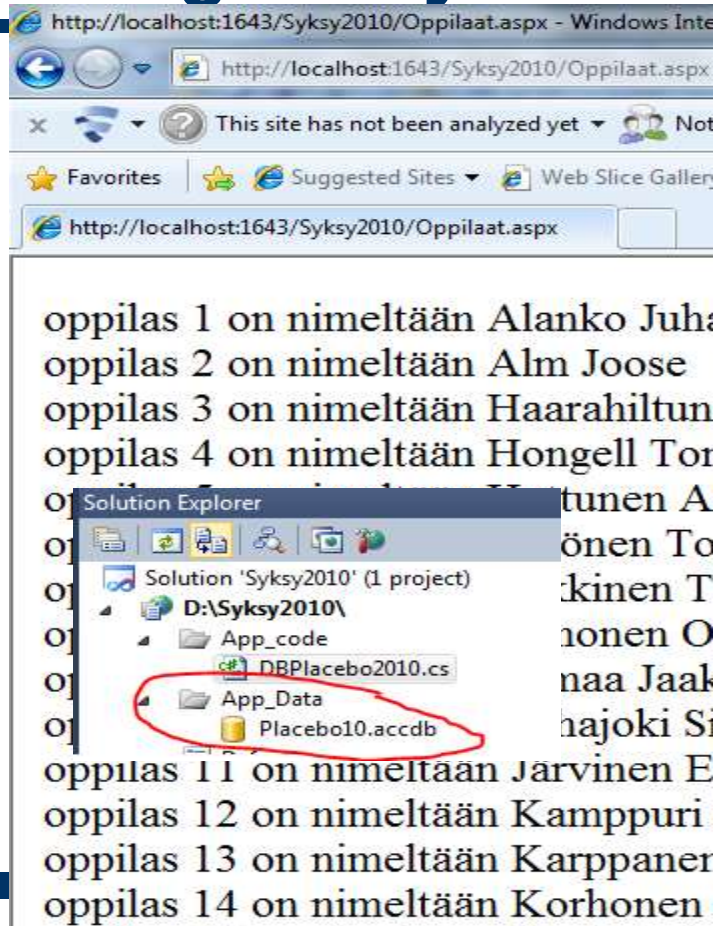
- References to a file/datastore in Web.Config

```
<connectionStrings>
<add name="oppilaatAbsoluuttinen"
  connectionString="Provider=Microsoft.ACE.OLEDB.12.0;
  Data Source=D:\Syksy2010\App_Data\Placebo10.accdb"/>
<add name="oppilaatSuhteellinen"
  connectionString="Provider=Microsoft.ACE.OLEDB.12.0;
  Data Source=|DataDirectory|\Placebo10.accdb"/>
</connectionStrings>
```

Demo: Oppilaat.aspx

App_Datan ja web.configin käyttö

- Kopioi **App_Data** kansioon tiedosto **Placebo.accdb** aseta Web.configiin tietokannan ja taulun nimi
- Käytä DBPlacebo-luokan staattista metodia **GetAllStudents**



Agenda

- Background
- ASP.NET Overview
- Programming Model
- **Programming Basics**
- Server Controls
- Data Binding
- Advanced topics
- Conclusion

STATE MANAGEMENT

A FAQ: How to Pass Values between ASP.NET Web Pages?

- You can pass information between pages in various ways, some of which depend on how the redirection occurs.
  - 1) Use a query string

    available even if the source/target page is in a different ASP.NET Web application from the target page, or if the source/target page is not an ASP.NET Web page
  - 2) Use session state

    available only when the source and target pages are in the same ASP.NET Web application.
  - 3) Cookies
- Yleisemmin kyseessä on “tilan hallinnasta” = state management →

ASP.NET State Management Overview

- ASP.NET is based on the stateless HTTP protocol, so:
  - each request from the client browser to the web server is understood as a independent request and
  - a new instance of the Web page class is created each time the page is posted to the server.

    →typically means that all information associated with the page and the controls on the page would be lost with each round trip

ASP.NET State Management Overview #2

- State Management is one of the most important concepts is ASP.NET.
- It is a technique used to maintain state information for ASP.NET web pages across multiple requests.
- To overcome this inherent limitation of traditional Web programming, ASP.NET includes several options that help you preserve data on both a per-page basis and an application-wide basis

ASP.NET State Management Features

ASP.NET features to preserve data on both a per-page basis and an application-wide basis are:

- View state
- Control state
- Hidden fields
- Cookies
- Query strings
- Application state
- Session state
- Cache object

Client-side state management

Server-side state management

CLIENT-SIDE STATE MANAGEMENT

Client-Side State management

- To manage state information **on the client side** you can use:
  - ViewState
  - Hidden fields
  - Query strings
  - Cookies

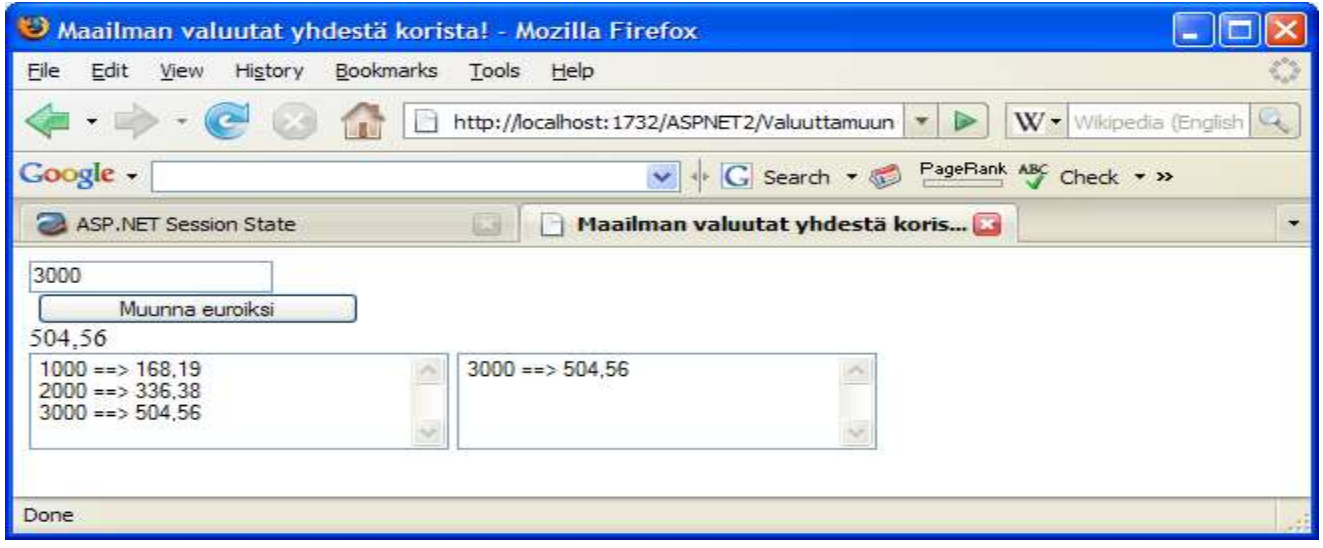
Programming Basics
Maintaining State

- By default. controls maintain their state across multiple postback requests
  - Implemented using a hidden HTML field: `__VIEWSTATE`
  - Works for controls with input data (e.g. `TextBox`, `CheckBox`), non-input controls (e.g. `Label`, `DataGrid`), and hybrids (e.g. `DropDownList`, `ListBox`)
- Can be disabled **per control** or **entire page**
  - Set `EnableViewState="false"`
  - Lets you minimize size of `__VIEWSTATE`

Demo: Valuuttamuunnin1.aspx

Maintaining State

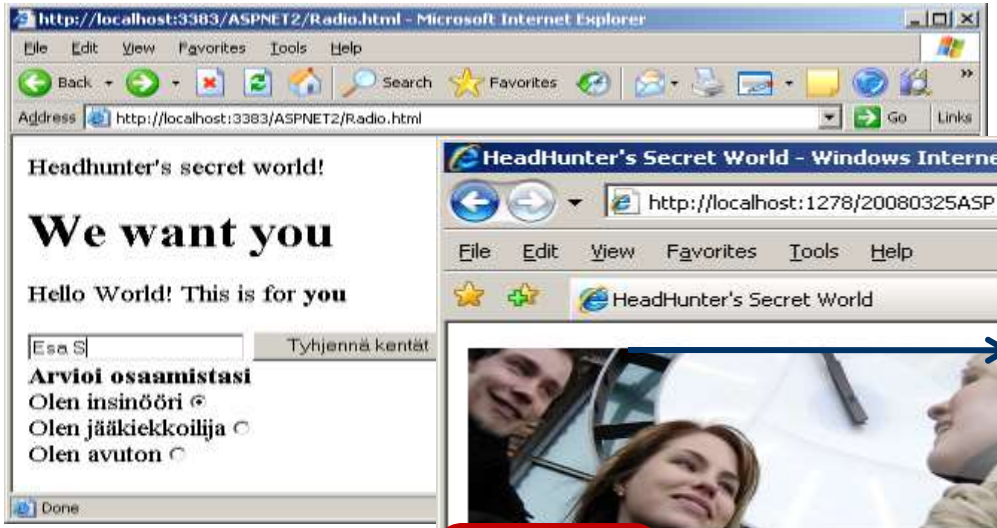
- Demo: Valuuttamuunnin1.aspx
  - *Default* EnableViewState = True

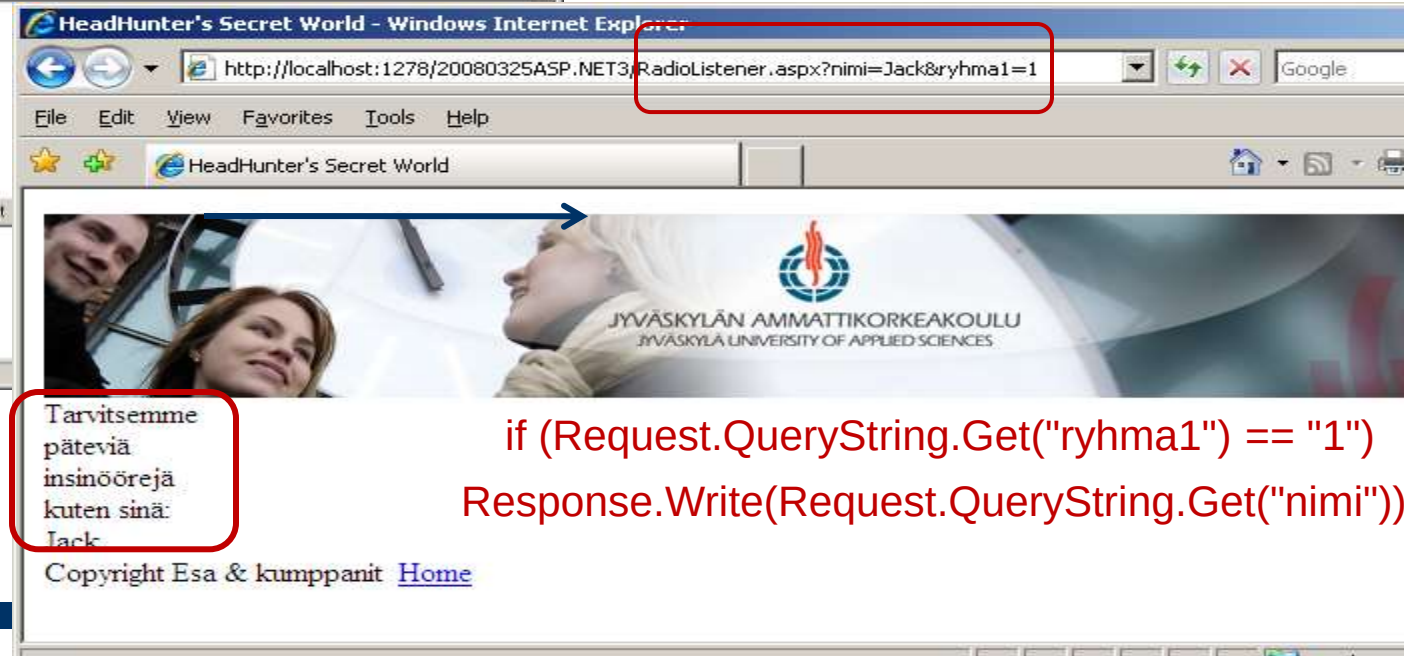


Tutki miten toteutettu?

Demo: RadioListener.aspx Query strings HTML → ASPX

Radio.html → **RadioListerner.aspx**






```
if (Request.QueryString.Get("ryhma1") == "1")
Response.Write(Request.QueryString.Get("nimi"))
```

Demo: HttpRequest.aspx Query strings

- HttpRequest luokka

  Enables ASP.NET to read the HTTP values sent by a client during a Web request

http://localhost:1831/Syksy2010/HttpRequest.aspx - Windows Internet Explorer

http://localhost:1831/Syksy2010/HttpRequest.aspx

Favorites | MSDN AA lataukset | Etusivu Yle Areena yle | Suggested Sites | Web Slice Gallery

Page | Safety | Tools

Pyyntö osoitteesta: 127.0.0.1 parametreillä

Done

http://localhost:1831/Syksy2010/HttpRequest.aspx?user=Esa - Windows Internet Explorer

http://localhost:1831/Syksy2010/HttpRequest.aspx?user=Esa

Favorites | MSDN AA lataukset | Etusivu Yle Areena yle | Suggested Sites | Web Slice Gallery

Page | Safety | Tools

Pyyntö osoitteesta: 127.0.0.1 parametreillä user=Esa

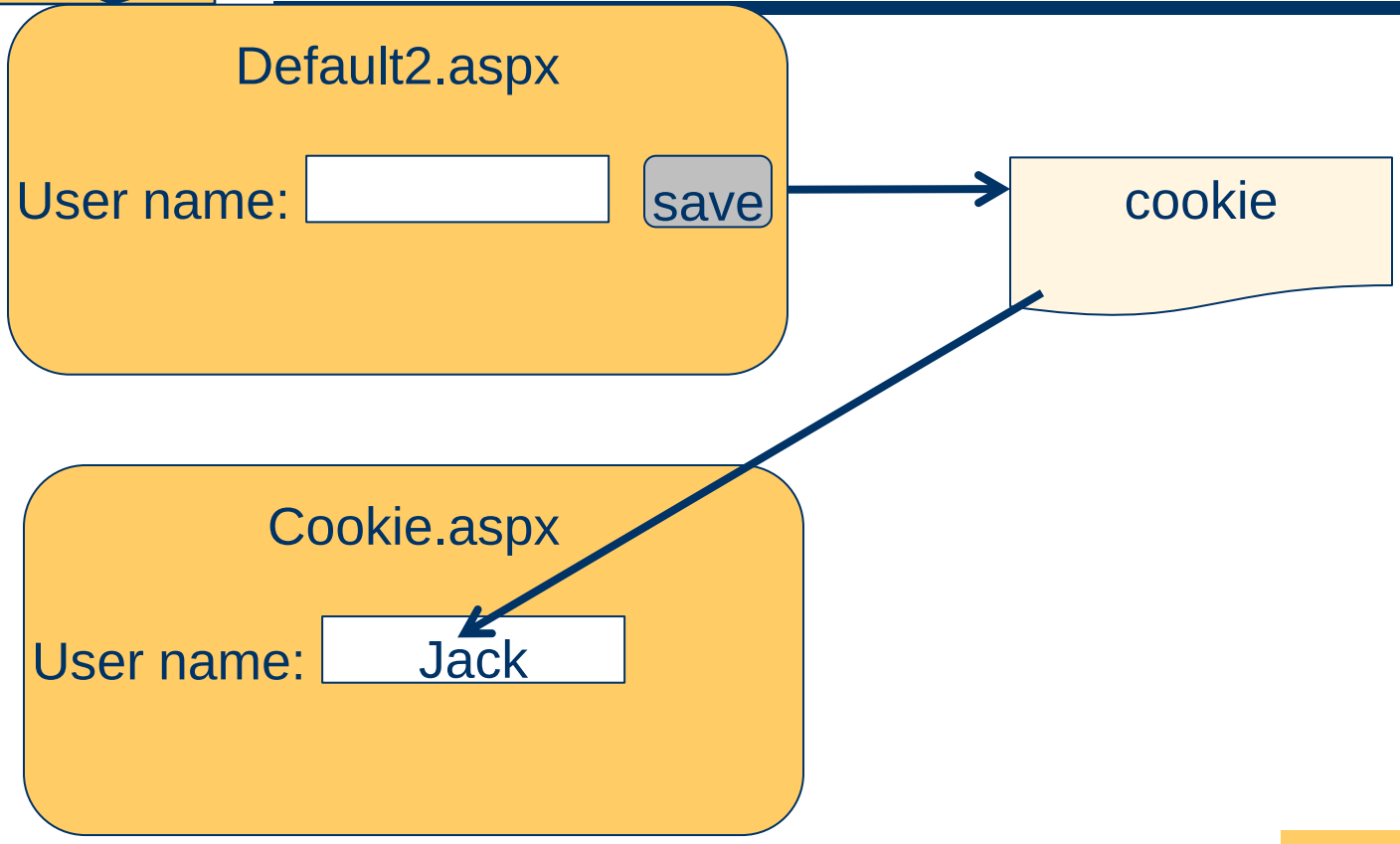
Key: user
Value 0: Esa

Cookies

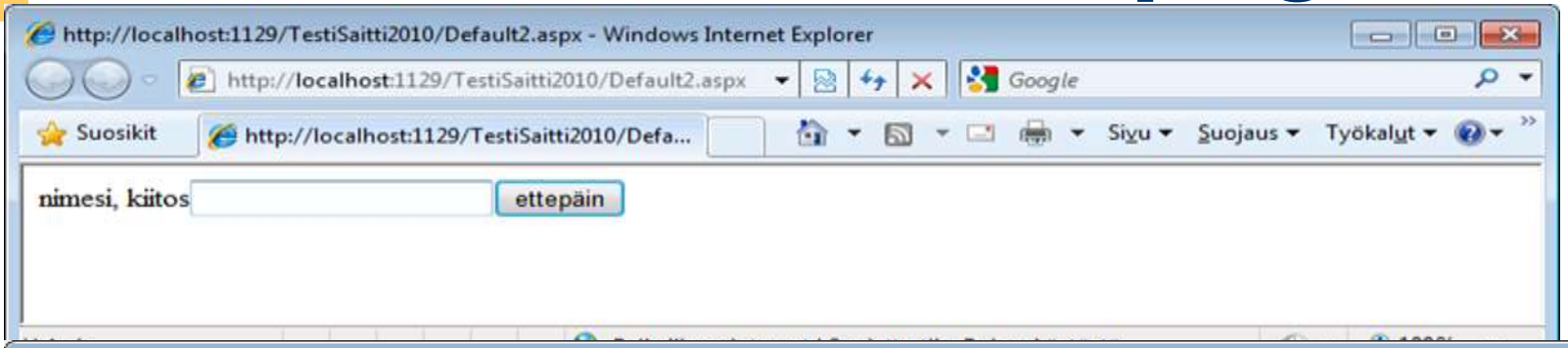
- Cookies provide a means in Web applications **to store user-specific information**.
- Cookies are user as a client-side state management
- **A cookie** is a small bit of text
  - Stored on the client side
  - accompanies requests and pages as they go between the Web server and browser
- Two types in AS
  - Temporary coo
  - Permantent co

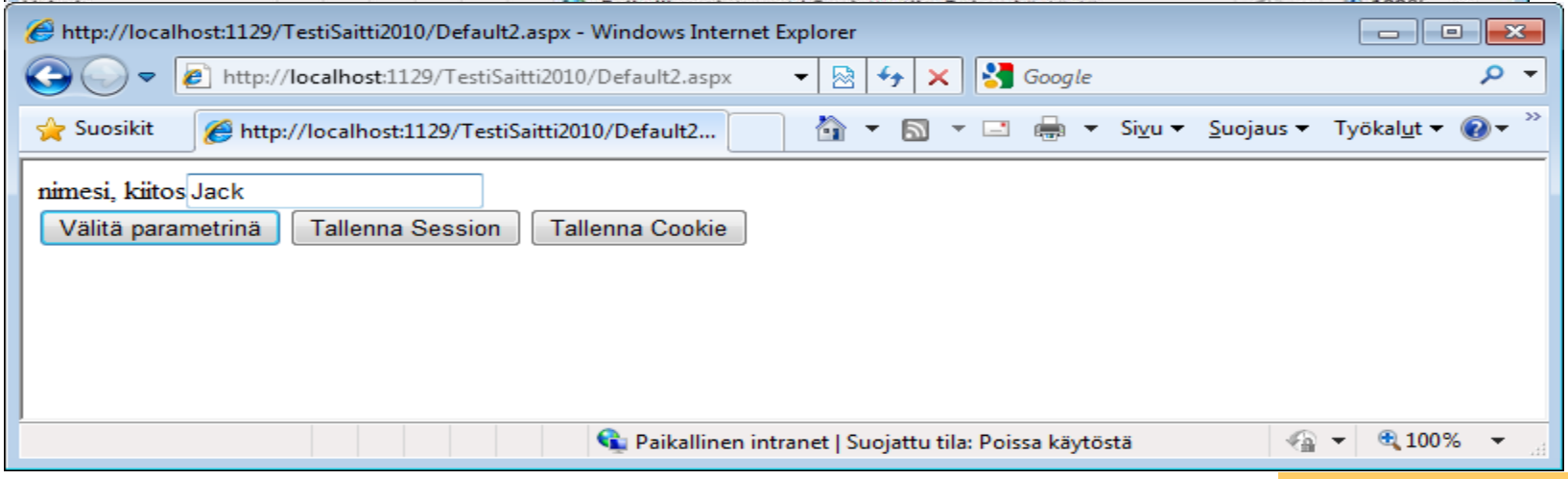
Although cookies can be very useful in your application, the application should not depend on being able to store cookies.
Do not use cookies to support critical features.
Do not use for sensitive data.
If your application must rely on cookies, you can test to see whether the browser will accept cookies.

Demo: Cookien käyttö



Demo: 3 ways to pass values between pages





SERVER-SIDE STATE MANAGEMENT

Server-side State Management

- You can manage state information **on the server** using with:
  - Application Object
  - Session Object
  - Cache Object

Maintain Application State in Application Object

- Scenario: You need to store data on an application-wide basis
- Application data is exactly that: it is same across the entire app, **for all users in all sessions**.
  - It is equivalent of a global varible
  - Global Application static class
- An Example

```
// in global.asax
Application[”LastLoaded”] = DateTime.UtcNow;
//..later
DateTime dt = (DateTime)Application[”LastLoaded”];
```

Session State

- HTTP is a **stateless** protocol, meaning that your Web server treats each HTTP request for a page as ***an independent request***;
  - by default, the server retains ***no knowledge*** of variable values used during previous requests.
- ASP.NET session state enables you to **store and retrieve** values for a user as the user navigates the different ASP.NET pages within a Web app
  - See more MSDN ASP.NET Session State

Maintain User Data in a Session

- If You need to store data associated with a specific user across page loads use **Session**
  - ASP.NET session state is **enabled** by default for all ASP.NET applications.
- Session state variables are easily set and retrieved using the Session property, which stores session variable values as a collection indexed by name.
- `Session["käyttäjä"] = txtName.Text;`

Demo: Pass Values Between ASP.NET Web Pages

Pistetään talteen Session-objektiin sivulla A

```
Session["käyttäjä"] = txtName.Text;
```

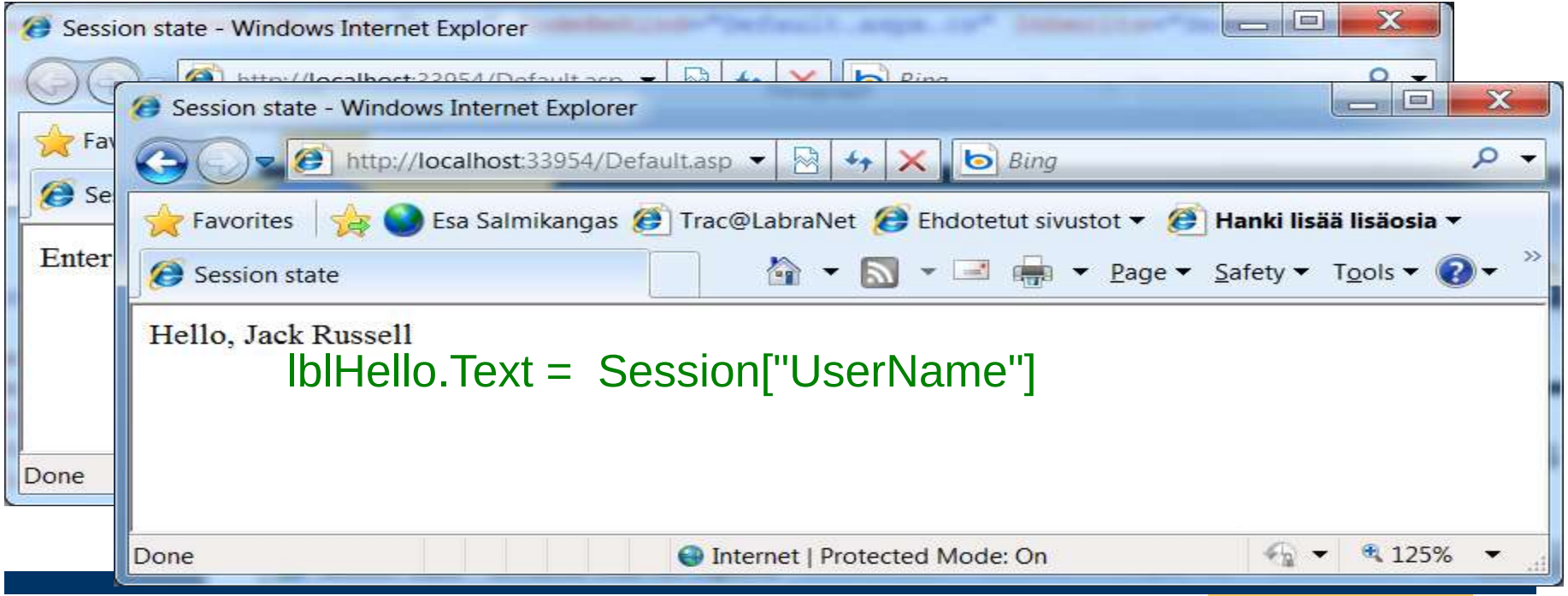
Ja käytetään sitten kun tarvitaan sivulla B, C jne:

```
txtToinen.Text = Session["käyttäjä"];
```

http://msdn.microsoft.com/en-us/library/6c3yckfw.aspx

Demo: Session State

- A user session generally starts when your user hits your site for the first time (and possibly logs in)…



Programming Basics
Global.asax

- Global.asax = ASP.NET Application File
- Contains code for responding to application-level and session-level events
  - ASP.NET automatically binds application events to event-handler methods in the Global.asax file using a naming convention of **Application**_*event*, such as **Application_BeginRequest** and **Application_Error**
- Is optional
- Safe: any direct URL requests are rejected
  - See more: http://msdn.microsoft.com/en-us/library/2027ewzw.aspx

ASP.NET caching

- Caching is a state management technique that can dramatically improve the application performance
- two types of caching that you can use to create high-performance Web applications:
  - page output caching
  - application data caching

Page output caching

- the entire page is cached in memory so subsequent requests for the same page are addressed from the cache itself
- is implemented with OutputCache –directive

```
<% OutputCache Duration=“30” …%>
```

Data Caching → Cache object

- enables to store relatively stale data in the cache for retrieval later so as to reduce the load on the server's resources.
- `DataSet ds = Cache["myDS"] as DataSet;`
- A demo later…
  - look more: **Caching Application Data**

THE PAGE

Programming Basics
Page Syntax

- The most basic page is just static text
  - Any HTML page can be renamed `.aspx`
- Pages may contain:
  - Directives: `<%@ Page Language="C#" %>`
  - Server controls: `<asp:Button runat="server">`
  - Code blocks: `<script runat="server">…</script>`
  - Data bind expressions: `<%# %>`
  - Server side comments: `<%-- --%>`
  - Render code: `<%= %>` and `<% %>`
    - Use is discouraged; use `<script runat=server>` with code in event handlers instead

Programming Basics
The Page Directive

- Lets you specify page-specific attributes, e.g.
  - `Buffer`: Controls page output buffering
  - `CodePage`: Code page for this .aspx page
  - `ContentType`: MIME type of the response
  - `ErrorPage`: URL if unhandled error occurs
  - `Inherits`: Base class of Page object
  - `Language`: Programming language
  - `Trace`: Enables tracing for this page
  - `Transaction`: COM+ transaction setting
  - `Debug`=”true”
  - `MasterPageFile`="~/MasterPage.master"
- Only one page directive per .aspx file

Programming Basics
`Import` Directive

- Adds code namespace reference to page
  - Avoids having to fully qualify .NET types and class names
  - Equivalent to the C# `using` directive

in html so in aspx

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Net" %>
<%@ Import Namespace="System.IO" %>
```

in code so in aspx.cs

```
using System.Data;
…
```

Programming Basics
Page Events

- Pages are structured using events
  - Enables clean code organization
  - Avoids the “Monster IF” statement
  - Less complex than ASP pages
- Code can respond to page events
  - e.g. `Page_Load`, `Page_Unload`
- Code can respond to control events
  - `Button1_Click`
  - `Textbox1_Changed`

Programming Basics

Page Event Lifecycle

Initialize	`Page_Init`
Restore Control State	
Load Page	`Page_Load`
Control Events	
1. Change Events	`Textbox1_Changed`
2. Action Events	`Button1_Click`
Save Control State	
Render	
Unload Page	`Page_Unload`

Programming Basics
Page Loading

- `Page_Load` fires at beginning of request after controls are initialized
  - Input control values already populated

```
protected void Page_Load(Object s, EventArgs e) {
  message.Text = textbox1.Text;
}
```

Programming Basics
Page Loading

- `Page_Load` fires on every request
  - Use `Page.IsPostBack` to execute conditional logic
  - If a Page/Control is maintaining state then need only initialize it when `IsPostBack` is false

```
protected void Page_Load(Object s, EventArgs e) {
  if (! Page.IsPostBack) {
    // Executes only on initial page load
    Message.Text = "initial value";
  }
  // Rest of procedure executes on every request
}
```

Programming Basics
Page Class

- The `Page` object is always available when handling server-side events
- Provides a large set of useful properties and methods, including:
  - `Application`, `Cache`, `Controls`, `EnableViewState`, `EnableViewStateMac`, `ErrorPage`, `IsPostBack`, `IsValid`, `Request`, `Response`, `Server`, `Session`, `Trace`, `User`, `Validators`
  - `DataBind()`, `LoadControl()`, `MapPath()`, `Validate()`

Programming Basics
Page Unloading

- `Page_Unload` fires after the page is rendered
  - Don't try to add to output
- Useful for logging and clean up

```
protected void Page_Unload(Object s, EventArgs e) {
  MyApp.LogPageComplete();
}
```

Programming Basics
Server Code Blocks

- Server code lives in a script block marked `runat="server"`

```
<script language="C#" runat=server>
<script language="VB" runat=server>
<script language="JScript" runat=server>
```

- Script blocks can contain
  - Variables, methods, event handlers, properties
  - They become members of a custom `Page` object

Agenda

- Background
- ASP.NET Overview
- Programming Model
- Programming Basics
- **Server Controls**
- Data Binding
- Conclusion

THE ASP.NET SERVER CONTROLS

Overview of ASP.NET controls

- ASP.NET ships with ~90 built-in controls
  - Standard controls
  - Validation controls
  - Rich controls
  - Data controls
  - Navigation controls
  - Login controls
  - HTML controls

THE SYNTAX OF SERVER CONTROLS

Server Control Syntax

- Controls are declared as HTML tags with `runat="server"` attribute

```
<input type=text id=text2 runat="server" />
<asp:calendar id=myCal runat="server" />
```

- Tag identifies which type of control to create
  - Control is implemented as an ASP.NET class
- The `id` attribute provides programmatic identifier
  - It names the instance available during postback
  - Just like Dynamic HTML

Server Control Properties

- Tag attributes map to control properties

```
<asp:button id="c1" Text="Foo" runat="server">
<asp:ListBox id="c2" Rows="5" runat="server">
```

- Control properties can be set programmatically

```
c1.Text = "Foo";
c2.Rows = 5;
```

Server Control Events

- Change Events
  - By default, these execute only on next action event
  - E.g. `OnTextChanged`, `OnCheckedChanged`
  - Change events fire in random order
- Action Events
  - Cause an immediate postback to server
  - E.g. `OnClick`
- Works with any browser
  - No client script required, no applets, no ActiveX® Controls!

Wiring Up Control Events

- Control event handlers are identified on the tag

```
<asp:button onclick="btn1_click" runat=server>
<asp:textbox onchanged="text1_changed" runat=server>
```

- Event handler code

```
protected void btn1_Click(Object s, EventArgs e) {
  Message.Text = "Button1 clicked";
}
```

Event Arguments

- Events pass two arguments:
  - The sender, declared as type `object`
    - Usually the object representing the control that generated the event
    - Allows you to use the same event handler for multiple controls
  - Arguments, declared as type `EventArgs`
    - Provides additional data specific to the event
    - `EventArgs` itself contains no data; a class derived from `EventArgs` will be passed

THE HTML CONTROLS

Server Controls
HTML Controls

- Work well with existing HTML designers
- Properties map 1:1 with HTML

```
table.bgcolor ="red“;
```

- Can specify client-side event handlers
- Good when quickly converting existing pages
- Derived from `System.Web.UI.HtmlControls.HtmlControl`
- Supported controls have custom class, others derive from `HtmlGenericControl`

Server Controls
HTML Controls

- Supported controls
  - `<a>`
  - `<img>`
  - `<form>`
  - `<table>`
  - `<tr>`
  - `<td>`
  - `<th>`
  - `<select>`
  - `<textarea>`
  - `<button>`
  - `<input type=text>`
  - `<input type=file>`
  - `<input type=submit>`
  - `<input type=button>`
  - `<input type=reset>`
  - `<input type=hidden>`

Server Controls
HTML Controls

- Can use controls two ways:
  - Handle everything in action events (e.g. button click)
    - Event code will read the values of other controls (e.g. text, check boxes, radio buttons, select lists)
  - Handle change events as well as action events

THE ASP: CONTROLS

Server Controls
Web Controls

- Consistent object model

```
Label1.BackColor = Color.Red;
Table.BackColor = Color.Blue;
```

- Richer functionality
  - E.g. `AutoPostBack`, additional methods
- Automatic uplevel/downlevel support
  - E.g. validation controls
- Strongly-typed; no generic control
  - Enables better compiler type checking

Server Controls
Web Controls

- Web controls appear in HTML markup as namespaced tags
- Web controls have an `asp:` prefix

```
<asp:button onclick="button1_click" runat=server>
<asp:textbox onchanged="text1_changed" runat=server>
```

- Defined in the `System.Web.UI.WebControls` namespace
- This namespace is automatically mapped to the `asp:` prefix

Server Controls
Web Controls

- Web Controls provide extensive properties to control display and format, e.g.
  - `Font`
  - `BackColor`, `ForeColor`
  - `BorderColor`, `BorderStyle`, `BorderWidth`
  - `Style`, `CssClass`
  - `Height`, `Width`
  - `Visible`, `Enabled`

Server Controls
Web Controls

- Four types of Web Controls
  - HTML (Intrinsic) controls
  - Rich controls
  - List controls
  - Validation controls

Server Controls
HTML (Intrinsic) Controls

- Correspond to HTML controls
- Supported controls
  - `<asp:button>`
  - `<asp:imagebutton>`
  - `<asp:linkbutton>`
  - `<asp:hyperlink>`
  - `<asp:textbox>`
  - `<asp:checkbox>`
  - `<asp:radiobutton>`
  - `<asp:image>`
  - `<asp:label>`
  - `<asp:panel>`
  - `<asp:table>`

Intrinsic = olennainen

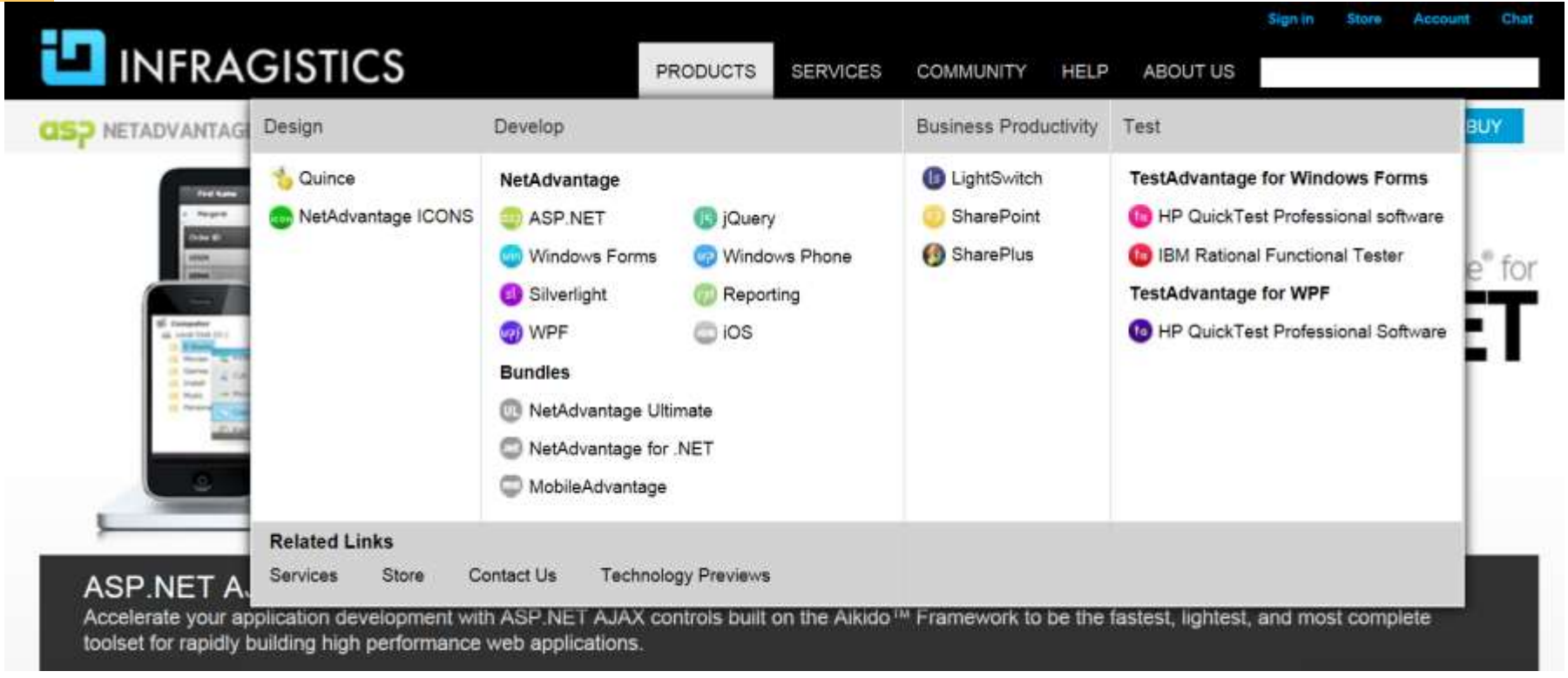
Server Controls
HTML (Intrinsic) Controls

- `TextBox`, `ListControl`, `CheckBox` and their subclasses don't automatically do a postback when their controls are changed
- Specify `AutoPostBack=true` to make change events cause a postback

Server Controls
Rich Controls

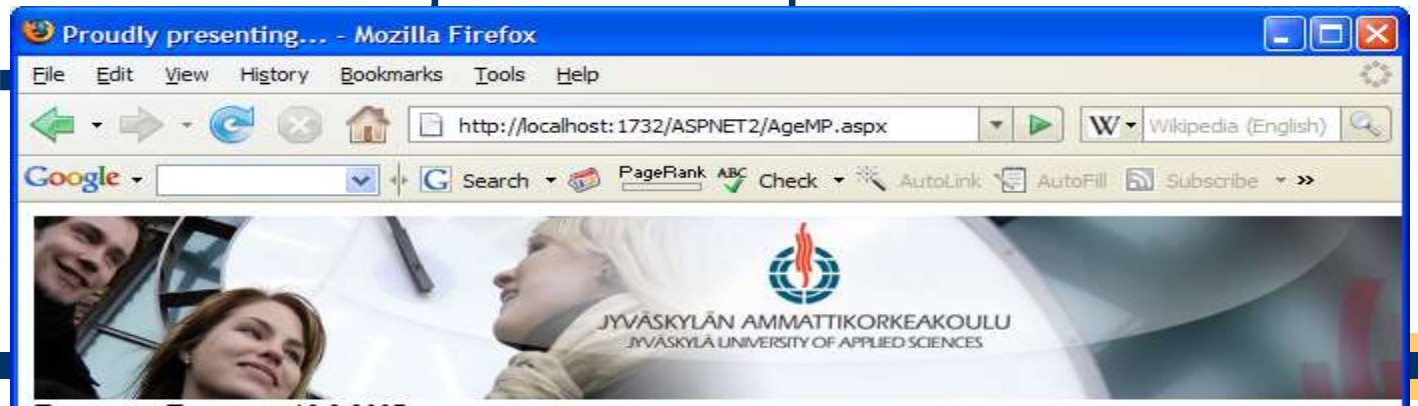
- Custom controls with rich functionality
- Supported Controls
  - `<asp:calendar>`
  - `<asp:adrotator>`
- More will be added
- Lot of 3rd party controls

Want some more very rich controls?



Demo: Age.aspx

- Tee sovellus joka näyttää käyttäjälle kalenterikomponentin, josta käyttäjä voi valita haluamansa päivämäärän.
- Valinnan jälkeen sovellus näyttää valitun päivämäärän ja kuluvan päivämäärän erotuksen:

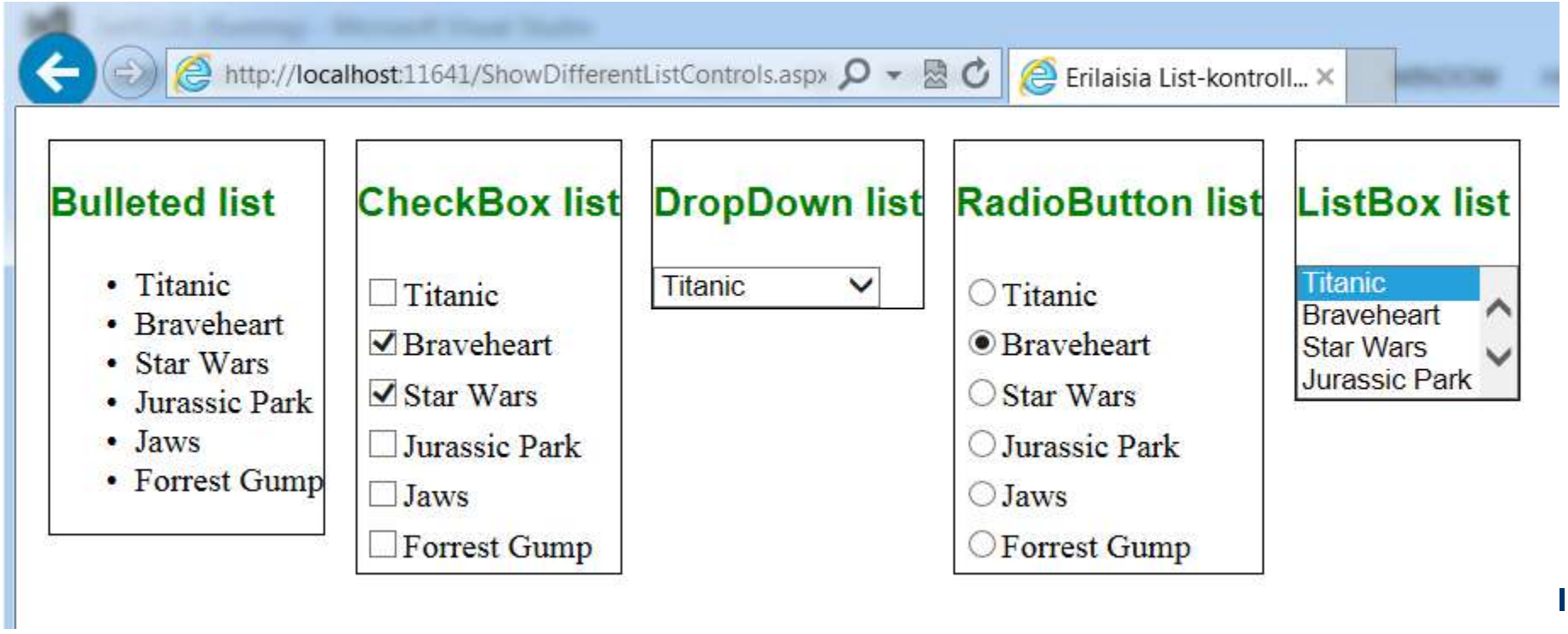


Server Controls
List Controls

- Controls that handle repetition
- Supported controls
  - `<asp:dropdownlist>`
  - `<asp:listbox>`
  - `<asp:radiobuttonlist>`
  - `<asp:checkboxlist>`
  - `<asp:repeater>`
  - `<asp:datalist>`

mo: DifferentListControls.aspx

- Yksi tietolähde **SqlDataSource**, useita erilaisia esitystapoja

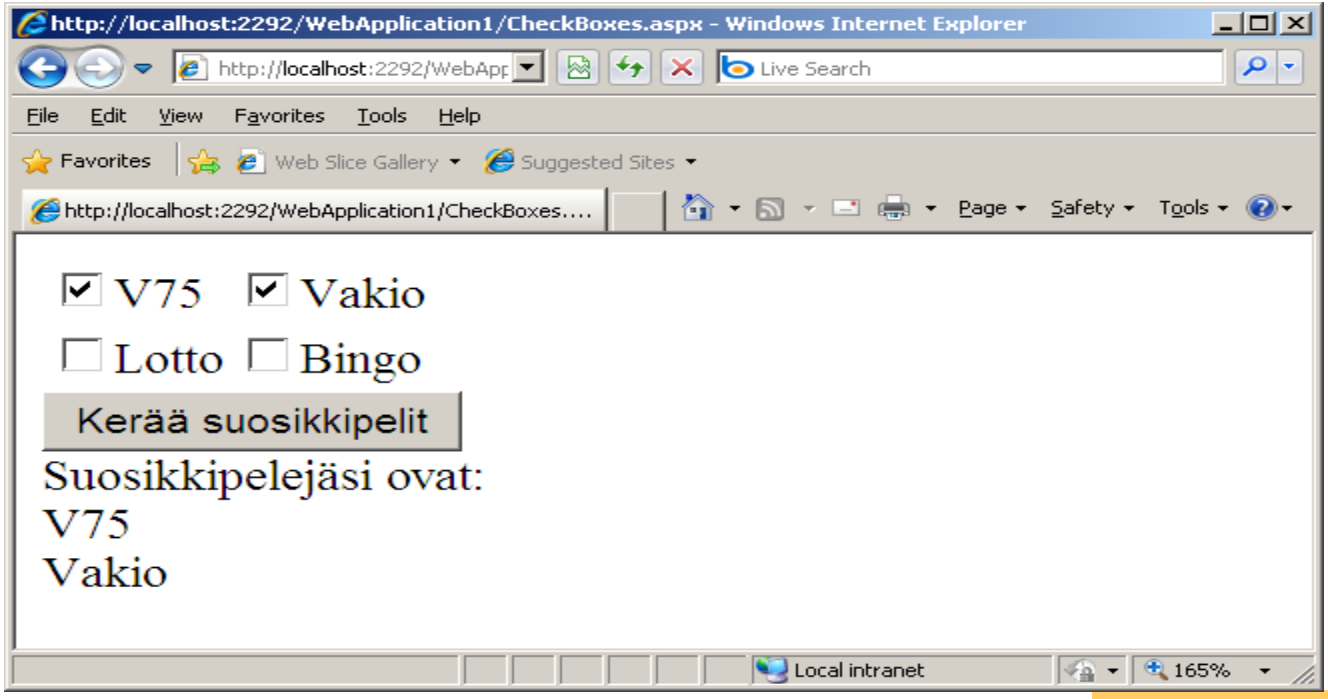


Server Controls
CheckBoxList & RadioButtonList

- Provides a collection of check box or radio button controls
- Can be populated via data binding

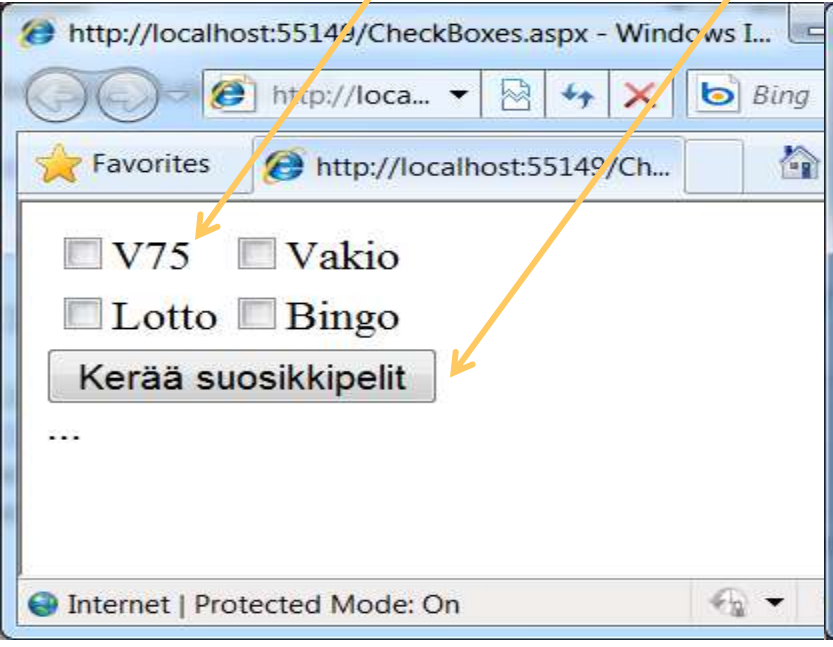
```
<asp:CheckBoxList id=Check1 runat="server">
  <asp:ListItem>Item 1</asp:ListItem>
  <asp:ListItem>Item 2</asp:ListItem>
  <asp:ListItem>Item 3</asp:ListItem>
  <asp:ListItem>Item 4</asp:ListItem>
  <asp:ListItem>Item 5</asp:ListItem>
</asp:CheckBoxList>
```

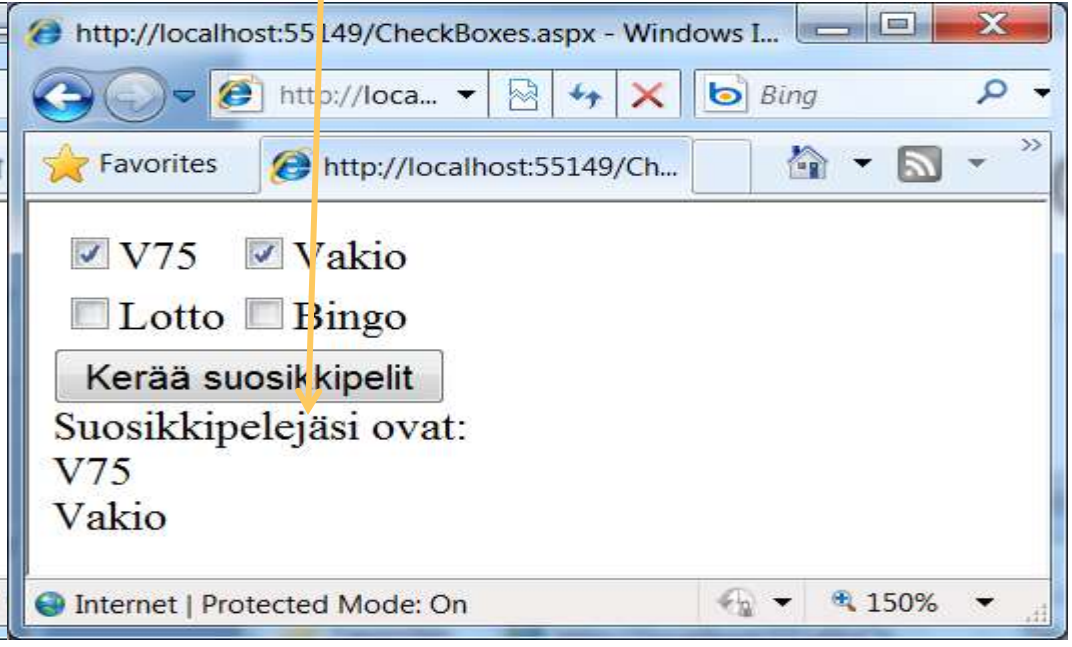
Demo: CheckBoxList.aspx



Demo: CheckBoxList.aspx

- CheckBoxList Button Label





Server Controls
List Controls

- `Repeater`, `DataList` controls
  - Powerful, customizable list controls
  - Expose templates for customization
  - Can contain other controls
  - Provide event bubbling through their `OnItemCommand` event
  - More about these controls and templates later

Demot myöhemmin Repeaterista ja DataList –kontrollista Data Binding -osiossa

Web Site Navigation

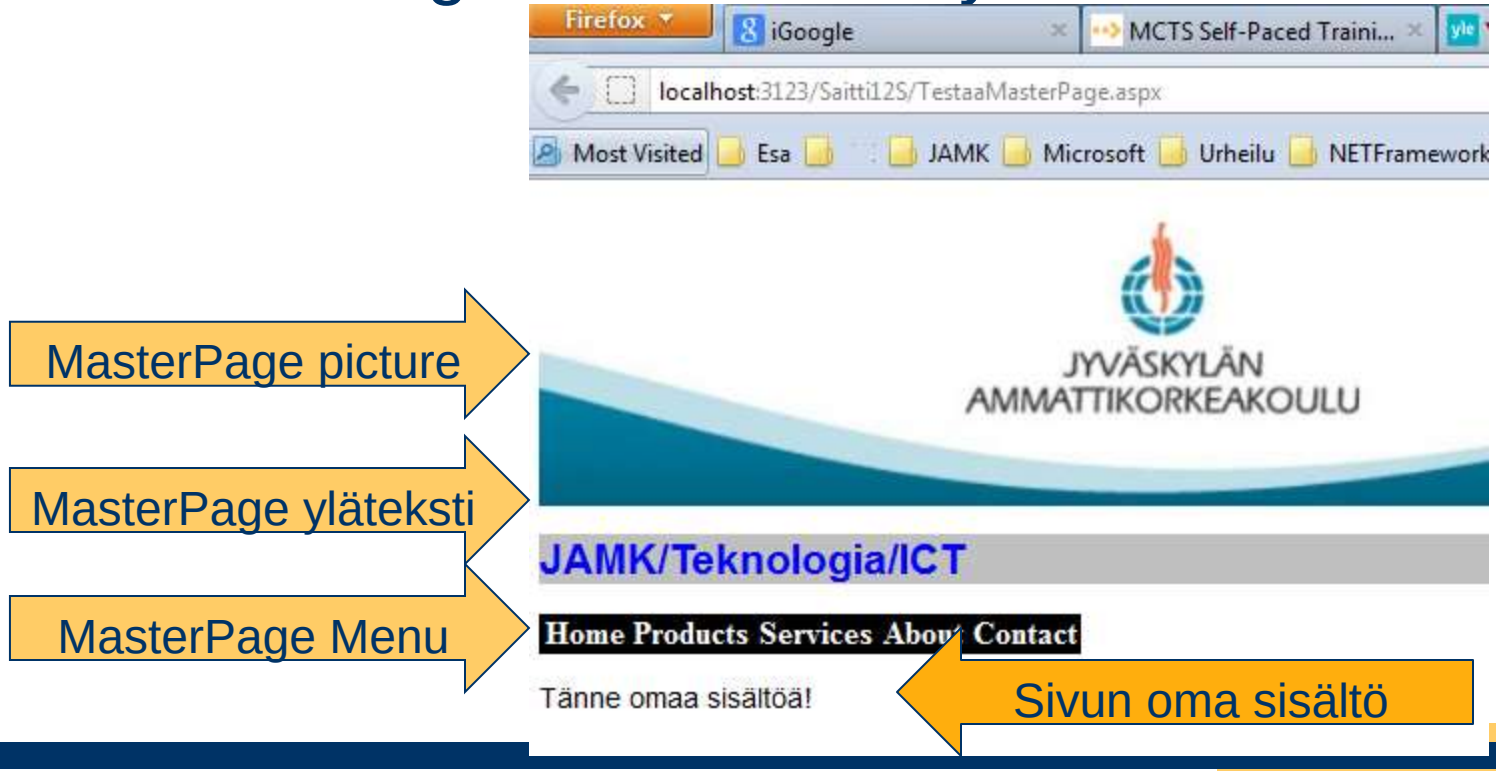
- Maintaining the menu of a large web site is difficult and time consuming.
- In ASP.NET the menu can be stored in a file **web.sitemap** to make it easier to maintain
  - stored in the root directory of the web.
- In addition, ASP.NET has three navigation controls:
  - Dynamic menus
  - TreeViews
  - Site Map Path

ASP:Menu Control

- The Menu control allows develop both statically and dynamically displayed menus for Web pages.
  - **Static display** means that the Menu control is fully expanded all the time. The entire structure is visible, and a user can click on any part.
  - In a dynamically displayed menu, only the portions you specify are static, while their child menu items are displayed when the user holds the mouse pointer over the parent node.
- the contents of the Menu control:
  - can be configured directly in the control
  - Or can be bind the control to a data source (xml, data).

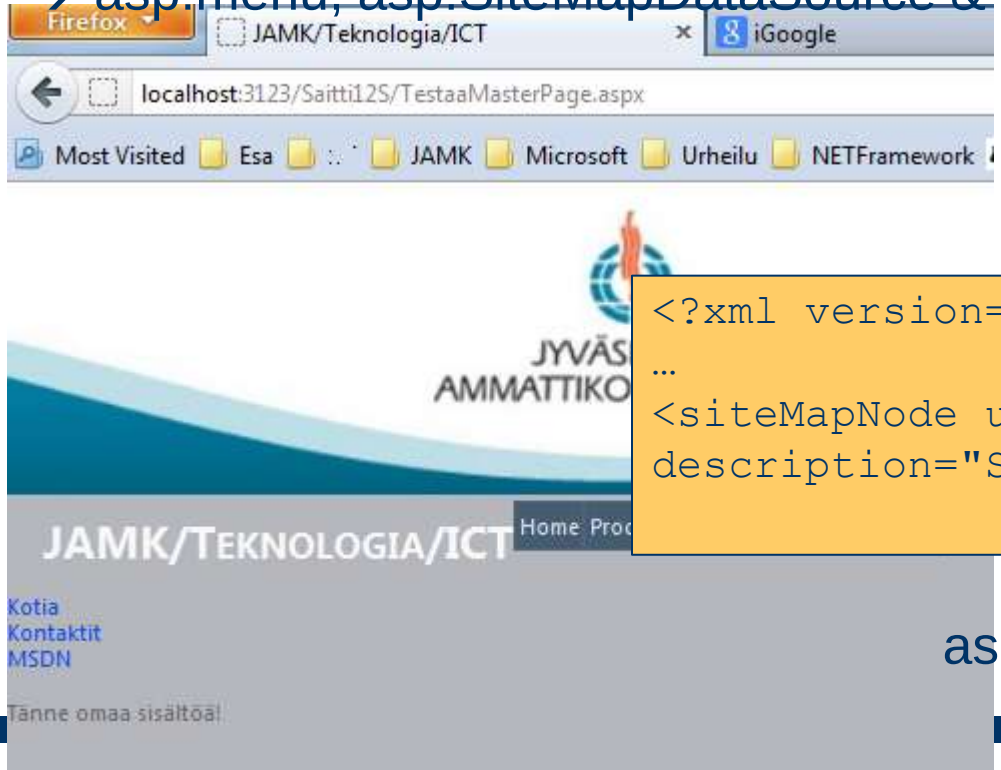
Demo: staattinen menu

- lisää MasterPageen staattinen ylämenu



Demo: Staattinen menu tiedostosta SiteMapMenu.aspx

- We want to provide a menu for site navigation
  - → asp:menu, asp:SiteMapDataSource & Web.sitemap file



Web.sitemap

```
<?xml version="1.0" encoding="utf-8" ?>
…
<siteMapNode url="~/Default.aspx" title="Ho
description="Site default page" />
```

asp:SiteMapDataSource ID

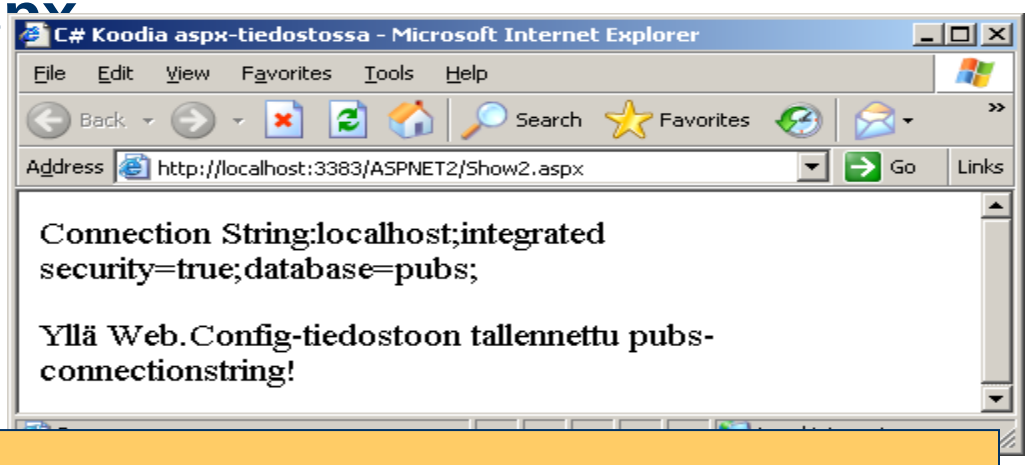
THE WEB.CONFIG

Web.config

- XML-pohjainen ASP.NET sovellusten konfigurointitiedosto (vrt App.Config)
- Eri tasoilla:
  - Sovelluskohtainen konfigurointi
  - Hakemisto ja sen alihakemistot
  - Koneen yleinen konfigurointi
- erilaisia asetuksia:
  - Tietokantayhteydet, sovelluksen asetukset, tilanhallinta, tietoturva, omia parametreja jne

Demo: Show2.aspx WebConfig (CodeBehind)

- Sama toiminnallisuus
  - ShowCodeBehind.aspx
  - Show2.aspx

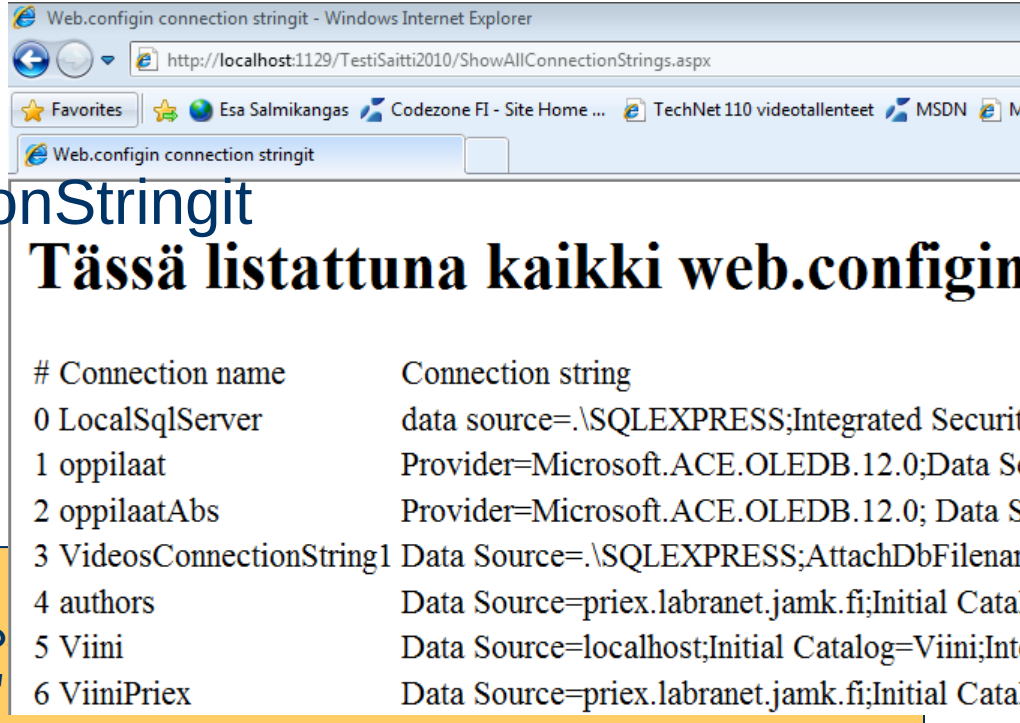


Web.Config

```
<connectionStrings>
<add name="pubs"    connectionString="localhost;integrated
security=true;database=pubs;" />
</connectionStrings >
```

Demo: ShowAllConnectionStrings.aspx

Tee sivu joka näyttää WebConfigin kaikki ConnectionStringit



Web.Config

```
<connectionStrings>
<add name="pubs"     connectio
security=true;database=pubs;"
</connectionStrings >
```

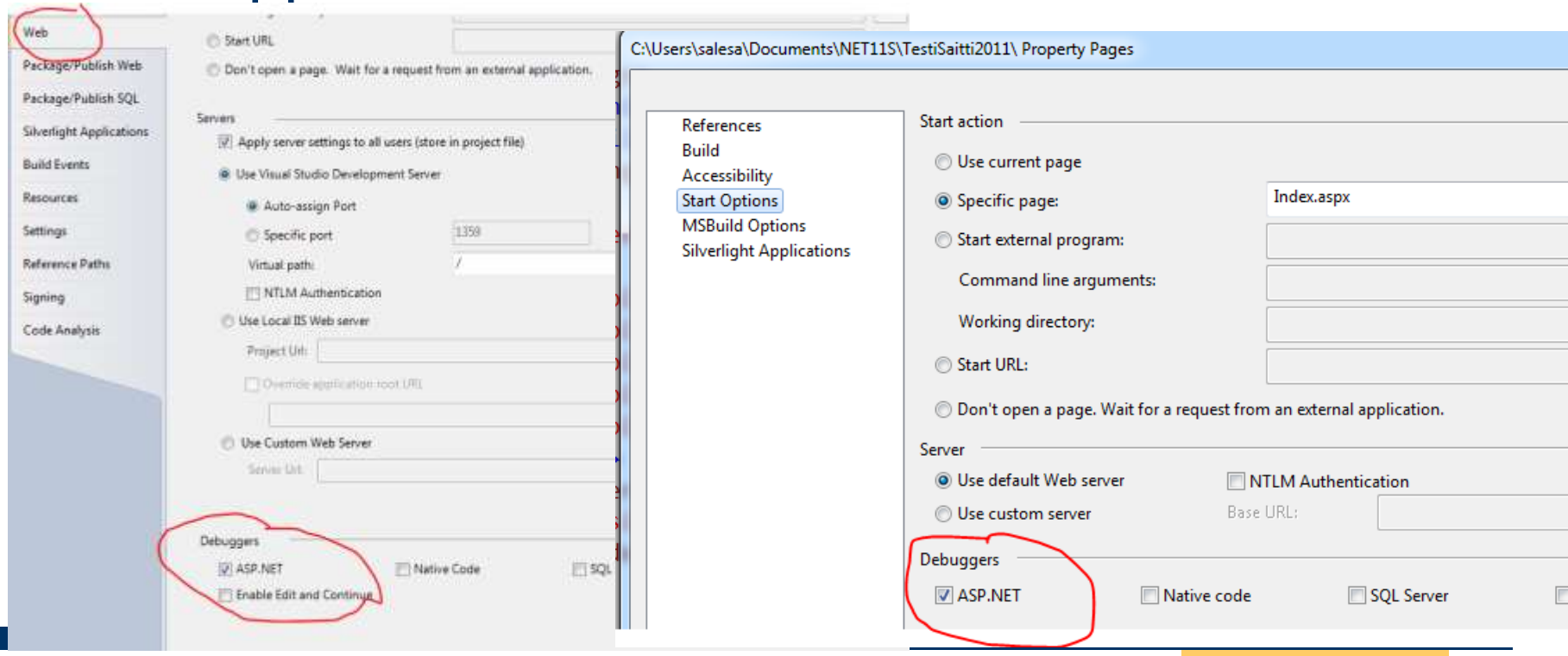
DEBUG, TRACE, REQUESTS

Debug

- The Visual Studio debugger provides powerful tools for debugging ASP.NET Web applications locally or on a remote server
- To debug a ASP.NET application:
  - You must have required permissions
  - ASP.NET debugging must be enabled in Project Properties.
  - The configuration file of your application (Web.config) must be set to debug mode.

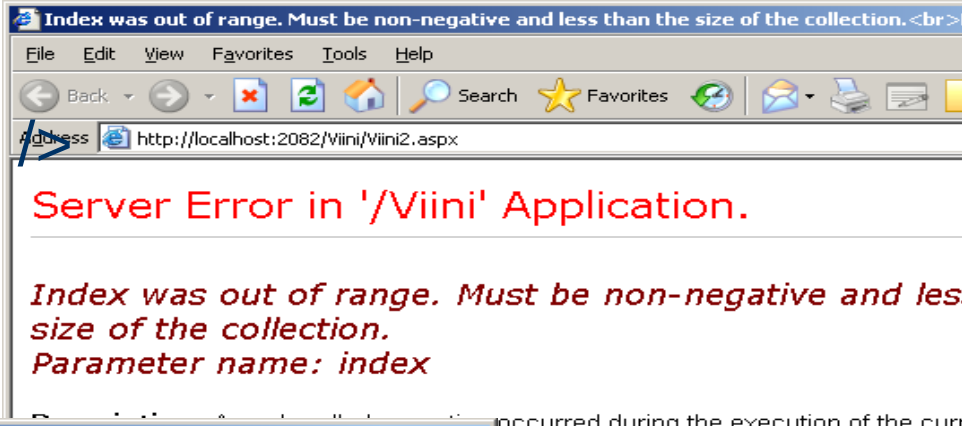
Debug WebApplication

- WebApplication / WebSite toimivat eri tavalla



Debug = ”true”

- How to: Enable Debugging for ASP.NET Applications
- Web.Config
  - <compilation debug="true" />
- In Page:
  - Debug="true"

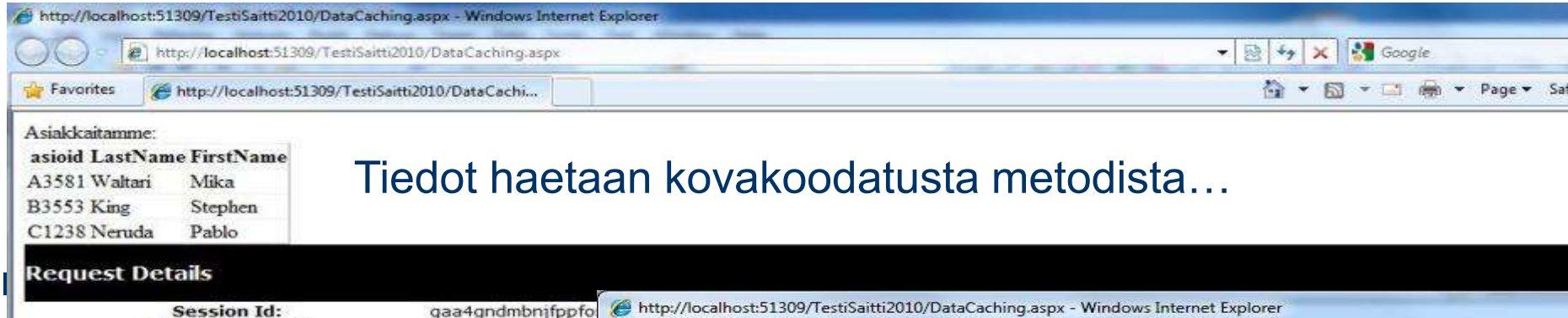


Trace asetukset

- Sivu ja sovellustasolla
- Ei tarvita omia Response.Write-tulosteita, vaan Trace päälle → valmiit trace-tulosteet
- Trace-tulosteessa:
  - kontrolliluettelo, tapahtumat aikaleimoin, server variables, form/query string parametrit
- Sivukohtainen Trace-direktiivi
  - `<%@ Page Trace="True" %>`
- omat tulosteet Trace.Write-metodilla

Demo: DataCaching.aspx
Tracing Data Handling

- Halutaan tutkia kuinka paljon DataTablen tallentaminen **Cacheen** nopeuttaa sivun päivitystä:
  - Trace asetettu päälle sivulle
  - Ennen ja jälkeen tiedonhakumetodin kirjoitetaan Traceen



Tiedot haetaan kovakoodatusta metodista…

Trace: a specific page or all pages

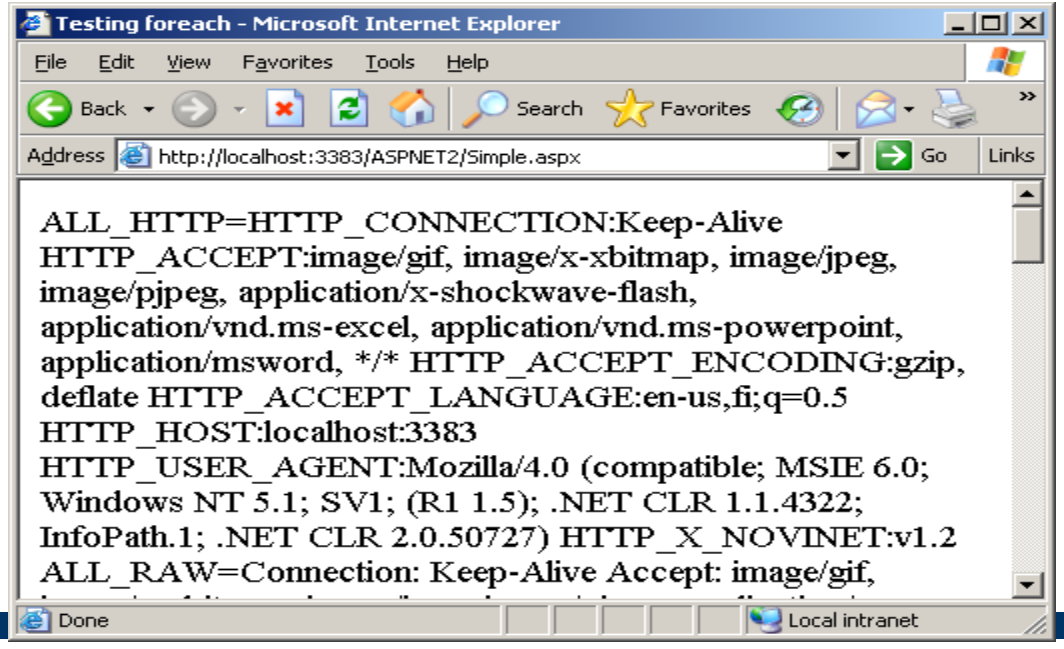
- Enable Tracing for a specific page
  - Add **Trace=”true”** to the @Page
  - `<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs” Inherits=“Some_Default" trace="true"%>`
- Tracing for all pages
  - Modify **web.config**
  - `<trace enabled="true" localOnly="false" pageOutput="true"/>`

SERVER VARIABLES

- HttpRequest.ServerVariables

Demo: Simple.aspx ServerVariables

- Request.ServerVariables
  - Gets a collection of Web server variables

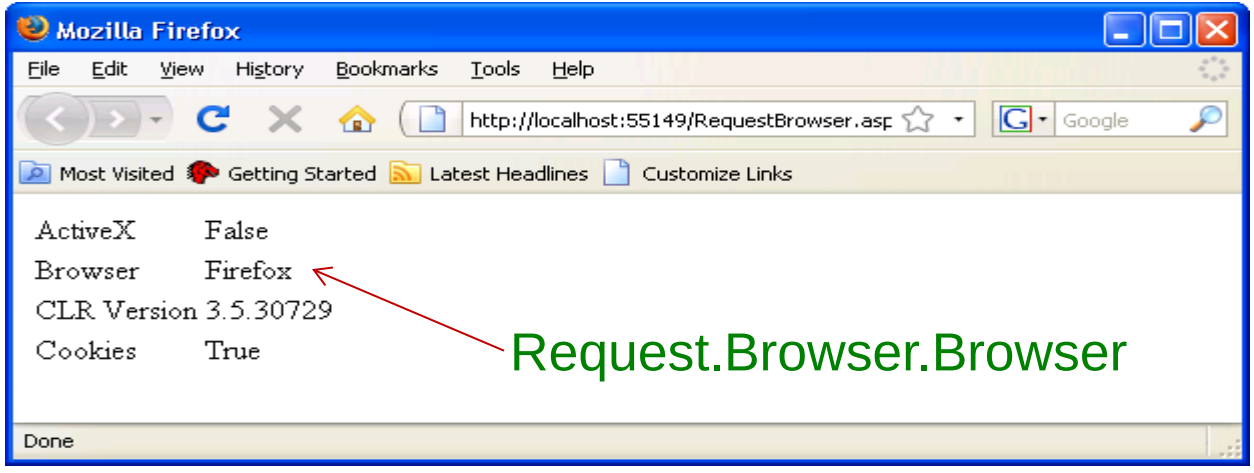


FAQ: Determine Web Browsers Capability

- Scenario:
  - You need to change your application's behavior based on the browser the user is using
- Solution:
  - .NET provides easy way to discover browser capabilitis by Using `Request.Browser`

Demo: RequestBrowser.aspx Browser capabilities

- **Request.Browser** structure contains values for many important properties



Demo: Browser Capabilities All

http://localhost:14709/Default.aspx - Windows Internet Explorer

http://localhost:14709/Default.aspx

Favorites | Esa Salmikangas | Trac@LabraNet | Ehdotetut sivustot | Hanki lisää lisäosia

Page | Safety | Tools

Browser Capabilities

Capability	Enabled
ActiveX	True
AOL	False
Background Sounds	True
Beta	False
Browser	IE
Can Render After Input or Select Element	True
Can Send Mail	True
CLR Version	3.5.30729
Cookies	True
Crawler	False
Default Submit Button Limit	1
ECMA Script Version	3.0
Frames	True
Has Back Button	True
Input Type	keyboard
Is Color	True
Is Mobile Device	False
Java Applets	True

Done | Internet | Protected Mode: On | 125%

Agenda

- Background
- ASP.NET Overview
- Programming Model
- Programming Basics
- Server Controls
- **Validator Controls**
- Data Binding
- Conclusion

THE VALIDATION CONTROLS

Data validation

- Web Security Rule #1

  ”All user input are evil until proven otherwise!”

→ We want to **validate** user input before sending it to the server!

- Two places to do input and data validation:
  1. On the client
  2. On the server
- ASP.NET provides various controls to checking that the input is in a certain range automatically
  - On the client side OR On the server side...

Server Controls
Validation Controls

- Rich, declarative validation
- Validation declared separately from input control
- Extensible validation framework
- Supports validation on client and server
  - Automatically detects uplevel clients
  - Avoids roundtrips for uplevel clients
- Server-side validation is **always** done
  - Prevents users from spoofing Web Forms

Server Controls
Validation Controls

- `<asp:RequiredFieldValidator>`
  - Ensures that a value is entered
- `<asp:RangeValidator>`
  - Checks if value is within minimum and maximum values
- `<asp:CompareValidator>`
  - Compares value against constant, another control or data type
- `<asp:RegularExpressionValidator>`
  - Tests if value matches a predefined pattern
- `<asp:CustomValidator>`
  - Lets you create custom client- or server-side validation function
- `<asp:ValidationSummary>`
  - Displays list of validation errors in one place

Server Controls
Validation Controls

- Validation controls are derived from `System.Web.UI.WebControls.BaseValidator`, which is derived from the `Label` control
- Validation controls contain text which is displayed only if validation fails
  - `Text` property is displayed at control location
  - `ErrorMessage` is displayed in summary

Validation controls

- Inheritance hierarchy
  - WebControl
    - Label
    - BaseValidator
      - Validation Controls

- RequiredFieldValidator
- CompareValidator
- RangeValidator
- RegularExpressionValidator
- CustomValidator
- ValidationSummary

- Validate associated input control
  - ControlToValidate property
  - Can apply multiple validation controls to a single control
- Support client-side validation (default)
  - Avoiding posting to server for validation
  - Delivering JavaScript to the client (no coding necessary!)
- Error message will be displayed if validation fails

Server Controls
Validation Controls

- Validation controls are associated with their target control using the `ControlToValidate` property

```
<asp:TextBox id=TextBox1 runat=server />

<asp:RequiredFieldValidator id="Req1"
  ControlToValidate="TextBox1"
  Text="Required Field" runat=server />
```

- Can create multiple validation controls with the same target control

Server Controls
Validation Controls

- `Page.IsValid` indicates if all validation controls on the page succeed

```
void Submit_click(object s, EventArgs e) {
  if (Page.IsValid) {
    Message.Text = "Page is valid!";
  }
}
```

Server Controls
Validation Controls

- `Display` property controls layout
  - `Static`: fixed layout, display won't change if invalid
  - `Dynamic`: dynamic layout
  - `None`: no display; can still use `ValidationSummary` and `Page.IsValid`
- `Type` property specifies expected data type: `Currency`, `Date`, `Double`, `Integer`, `String`

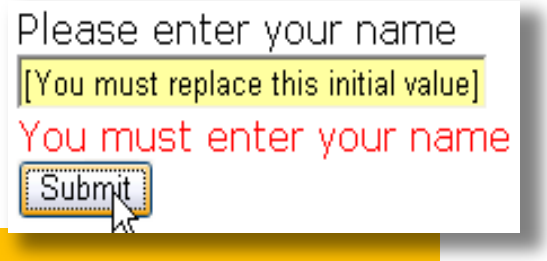
Server Controls
Validation Controls

- Can force down-level option
  - Only server-side validation

```
<% @ Page Language="c#"
          ClientTarget="DownLevel" %>
```

Example: Required field validator

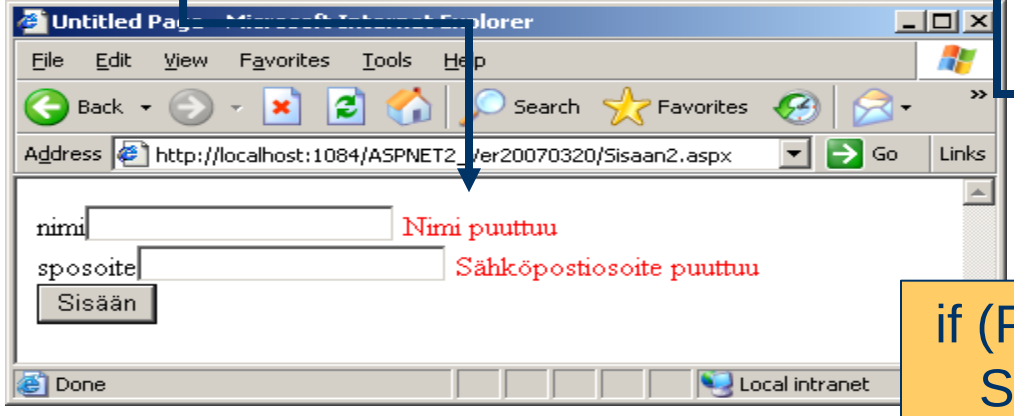
- Compares the user input to initial value of the input control
  - Validation fails if input is same as initial value

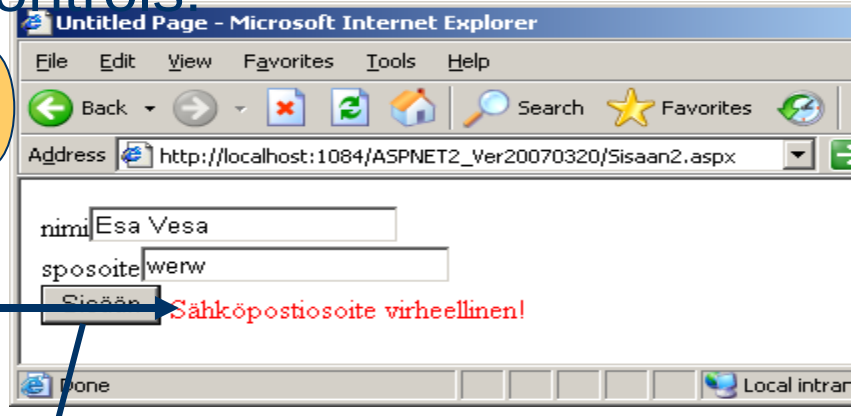



```
<asp:RequiredFieldValidator
    ID="vldReqName"
    runat="server"
    ErrorMessage="You must enter your name"
    InitialValue="[You must replace this initial value]"
    ControlToValidate="txtName">
</asp:RequiredFieldValidator>
```

Demo: Sisaan2.aspx
Validation Controls

- Demo: Sisaan2.aspx
  - Different type of validation controls:
    - RequiredFieldValidator
    - RegularExpressionValidator



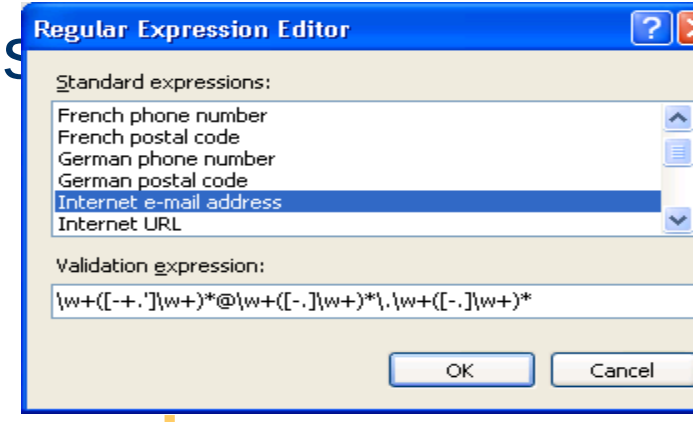



```
if (Page.IsValid)  {
    Server.Transfer("Default.aspx");
```

REGULAR EXPRESSION

Regular Expressions

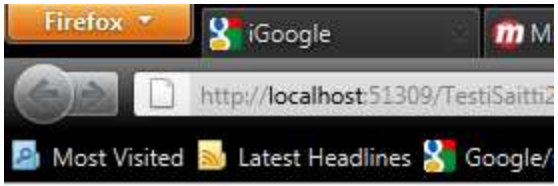
- Regular expressions provide an extremely powerful text processing system
- Gives you the power to find & check complex patterns



- .NET Framework Regular Expressions

Demo: MyRegEx.aspx

- Tee webbisivulle TextBox ja sille tarkistus että käyttäjän syöttämä teksti sisältää vain kirjaimia.
  - Sallittuja ovat vain isot ja pienet kirjaimet, mutta ei numerot tai erikoismerkit.



JYVÄSKYLÄN AMMATTIKORKEAKOULU

Ole hyvä, ja anna nimesi:

Validate

[Tähän kelpaako vai ei]

Home Proudly presents Esa & kni

Ole hyvä, ja anna nimesi:

JackRussell Validate

Kelpaa

Ole hyvä, ja anna nimesi:

Jack++

Ei kelpaa

Ole hyvä, ja anna nimesi:

Jack Russell Validate

Ei kelpaa

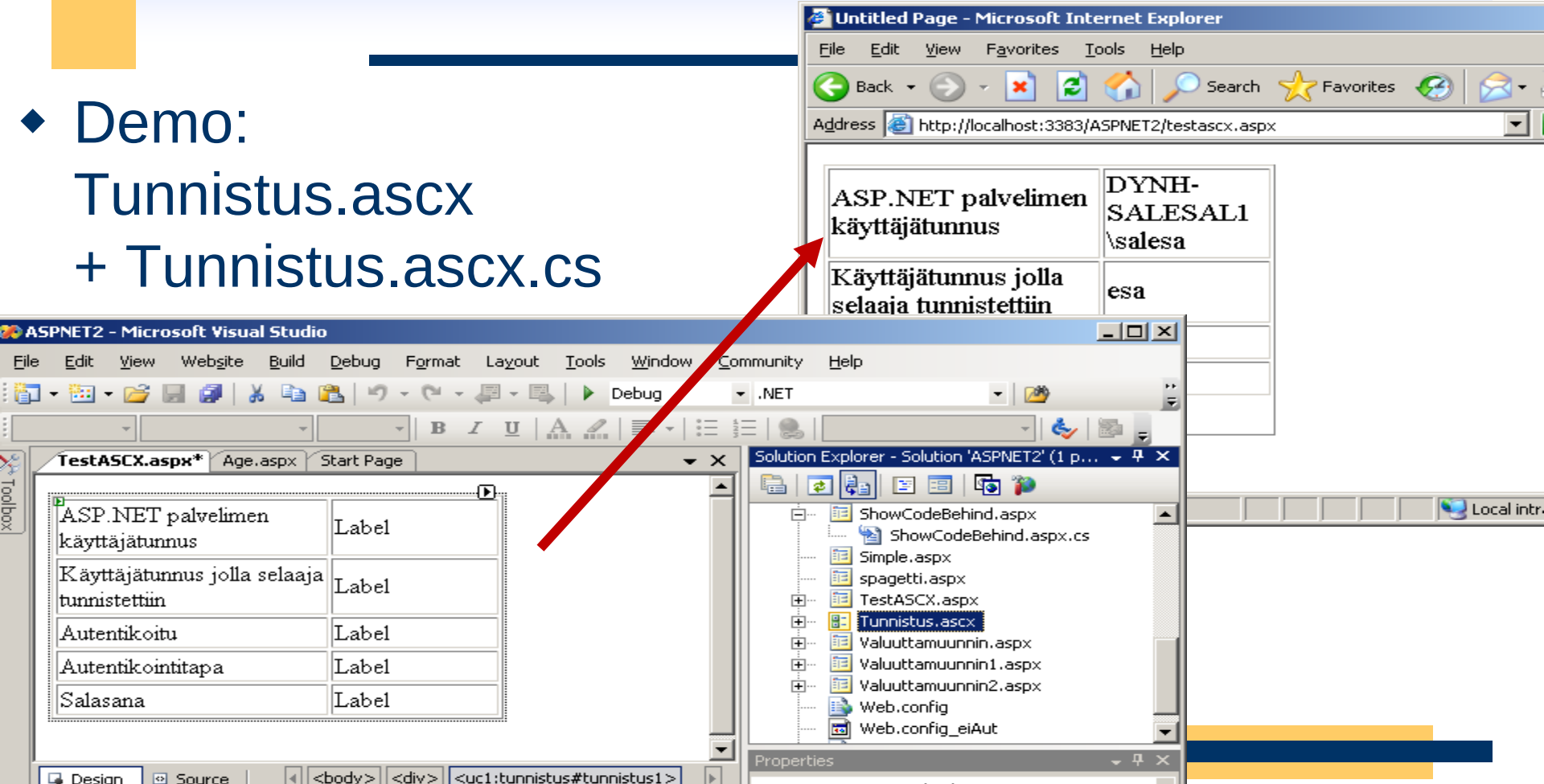
USER AND CUSTOM CONTROLS

Advanced: User control

- If we need functionality in a control that is not provided by the built-in ASP.NET Web server controls → we can create your own controls. You have two options. You can create:
  - **User controls**. User controls are containers into which you can put markup and Web server controls. You can then treat the user control as a unit and define properties and methods for it.
  - **Custom controls**. A custom control is a class that you write that derives from Control or WebControl.
- User controls are substantially easier to create than custom controls, because you can reuse existing controls. They make it particularly easy to create controls with complex user interface elements.

Advanced: User Control

- Demo:
  Tunnistus.ascx
  + Tunnistus.ascx.cs



Advanced topics

- ADO.NET & Data → ASP.NET_Data.pptx
- Security → ASP.NET_Security.pptx
- MVC → ASP.NET_MVC.pptx
- AJAX → ASP.NET_AJAX.pptx

Agenda

- Background
- ASP.NET Overview
- Programming Model
- Programming Basics
- Server Controls
- Data Binding
- Security
- **Conclusion**

Conclusion #1

- We covered
  - What ASP.NET and Web Forms are
  - ASP.NET Programming Essentials
  - Server Controls
  - Data Binding
  - Templates

A Recommended book the further studies:
Spanjaars I.: Beginning ASP.NET 4.5 in C# and VB, Wrox

Conclusion #2

We should continue to the following topics:

- Web Applications
- Configuration
- Tracing
- Session Management
- Error Handling
- Deployment
- Security
- Architecture
- Extensibility (User Controls and Custom Controls)

A Recommended book the further studies:
Evjen, Hanselmamm, Rader:
Professional ASP.NET 4 in C# and VB, Wrox

Jos kaikki ei suju niinkuin pitäisi…

- Stop Bashing Your Head against a Wall.
  1. There is a knowledge gap (*probability 90%*)
  2. The code sample is wrong (*9 %*)
  3. The documentation is wrong (*0,9%*)
  4. You are up against a bug in the .NET platform or ASP.NET (*0,09%*)
  5. World is against me! (*0,0000000009%*)

source: ASP.NET 3.5 for Dummies

Resources
“some often visited”

- http://msdn.microsoft.com/
- http://www.asp.net/
- http://www.devx.com/
- http://www.codeproject.com/
- any many, many others