M12 L1
Web Configuration File

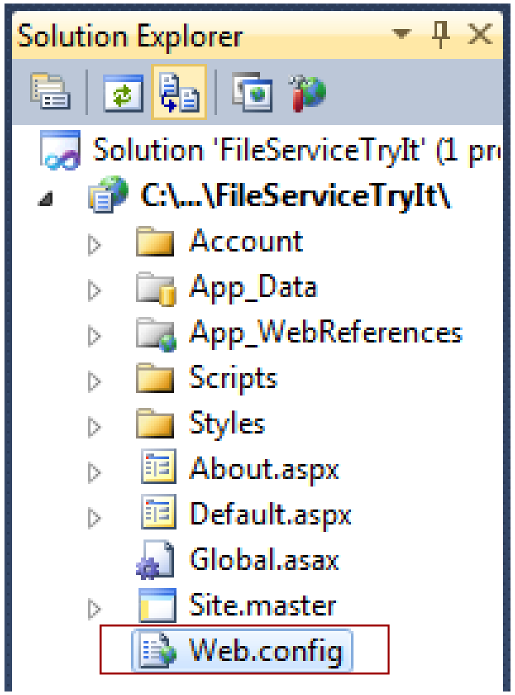
Ira A. Fulton Schools of Engineering
Arizona State University

Lecture Outline

- Web Configuration File and Its Elements
- Reading & Writing AppSettings in User Program
- Policy-Based Computing
- Web Configuration File Inheritance

ASP.Net Application and Its Components

- ASPX files containing Web forms
- ASCX files containing user controls
- **Web.config files containing configuration settings**
- A Global.asax file containing global application elements
- DLL (dynamic link library) files containing custom types employed by the application

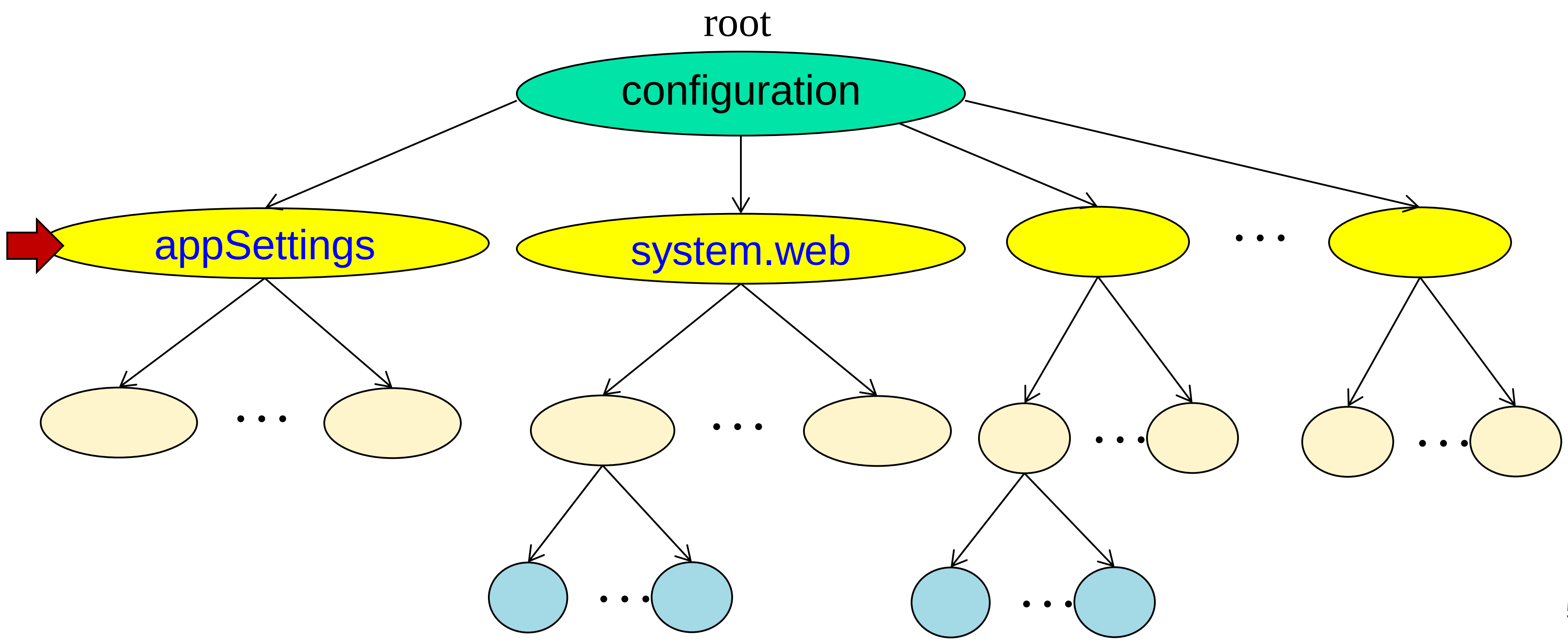


Web.config File

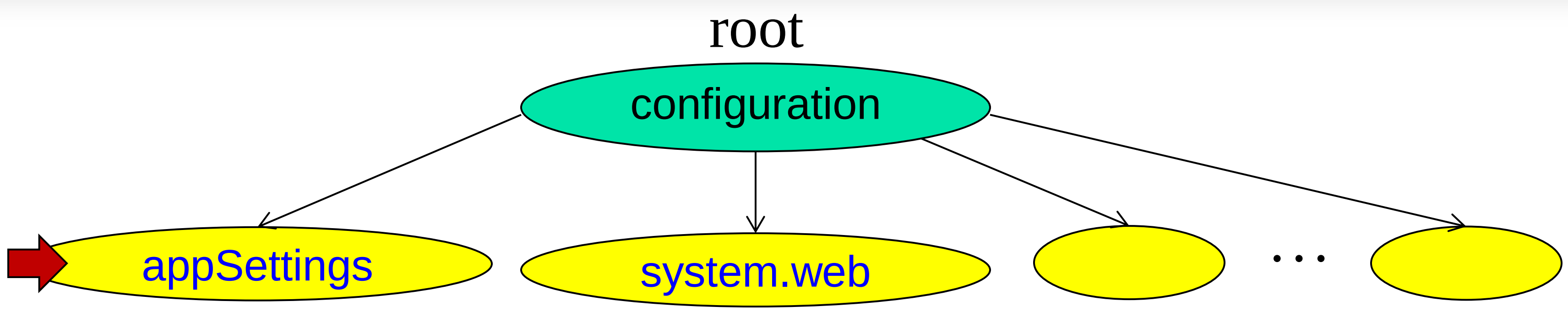
- .Net supports XCOPY deployment: Install software by simply drag and drop the software folder, uninstall software by deleting the folder;
- Keep the text-based configuration file with the application, instead of putting it in the system registry;
- Every Web service and Web application automatically includes a Web.config file;
- The file is never locked when the application is running, and it can be edited at any time, statically and dynamically;
- The files is transferable to other applications: simply copy and paste;
- The file is easily readable for both human and machine.

Web.config File Structure

- Web.config file is an XML file
- It controls various settings, including
  - Application data and parameter values
  - Communication protocols
  - Authentication and authorization setting (to be discussed in chapter 6)



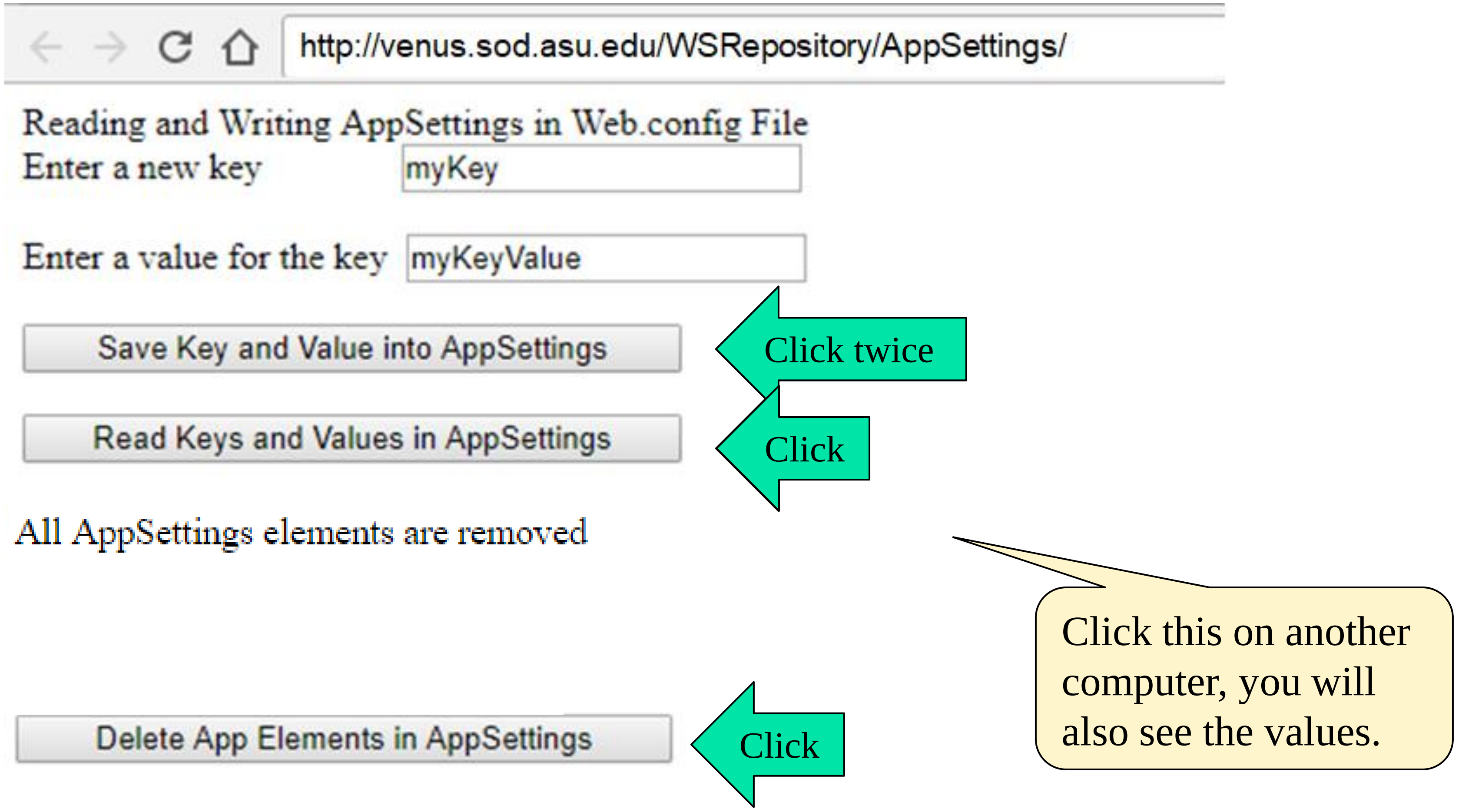
Web.config File: appSettings



```
<?xml version = “1.0” ?>
<configuration>
    <appSettings> <!-- appSetting here --> </appSettings>
    <connectionStrings> <!-- cS setting --> </connectionStrings>
    <system.web>
          <!--ASP.Net configuration settings -->
    <system.web>
    <system.codedom> <!-- setting --> </ system.codedom>
    <system.webServer> <!-- setting --> </ system.webServer>
<configuration>
```

Reading & Writing AppSettings in Program

- Data stored in AppSettings can be accessed in the program.
- There are different ways of accessing the data;
- Example: Design a Web Site App to Modify appSettings



http://venus.sod.asu.edu/WSRepository/AppSettings/

Reading and Writing AppSettings in Web.config File

Enter a new key myKey

Enter a value for the key myKeyValue

All AppSettings elements are removed

Save Key and Value into AppSettings
Click twice
Read Keys and Values in AppSettings
Click
Delete App Elements in AppSettings
Click
Click this on another computer, you will also see the values.

Code Behind the Page: Write

```
protected void btnSave_Click(object sender, EventArgs e) {
    System.Configuration.Configuration config =
        System.Web.Configuration.WebConfigurationManager.
            OpenWebConfiguration("~/"); // Open Web.config file
    // Create a new element into appSettings.
    int index =
        System.Configuration.ConfigurationManager.AppSettings.Count;
    string newKey = txtKey.Text + index.ToString(); // from textbox
    string newValue = txtValue.Text; // from textbox
    // Modify the appSettings in Web.config file.
    config.AppSettings.Settings.Add(newKey, newValue);
    // Save the changes into the Web.config file.
    config.Save(System.Configuration.ConfigurationSaveMode.Modified);
 }
```

Open the Web.config file

Add the new pair into AppSettings

Save the changes to the disk

Code Behind the Page: Read

```
protected void btnRead_Click(object sender, EventArgs e)
{
    System.Collections.Specialized.NameValueCollection myKeys =
        System.Web.Configuration.WebConfigurationManager.AppSettings;
    lblDisplay.Text = "";
    for (int i = 0; i < myKeys.Count; i++)
    {
        string appEntry = String.Format("Key {0}: {1} Value: {2} <br/>",
            i, myKeys.GetKey(i), myKeys[i]);
        lblDisplay.Text += appEntry;
    }
}
```

Create an object of AppSettings

Formating the AppSettings object for display

Code Behind the Page: Delete

```
using System;
using System.Xml;
protected void btnDelete_Click(object sender, EventArgs e)
{
     XmlDocument myCF = new XmlDocument();
     myCF.Load(AppDomain.CurrentDomain.SetupInformation.ConfigurationFile);
     foreach (XmlElement appElement in myCF.DocumentElement)
     {
        if (appElement.Name.Equals("appSettings"))
        {
               appElement.RemoveAll();
               lblDisplay.Text = "All AppSettings elements are removed";
        }
     }
     myCF.Save(AppDomain.CurrentDomain.SetupInformation.ConfigurationFile);
  }
```

Use XmlDocument class

Find appSettings element

Remove all child elements

Save the file

Policy-Based Computing

- **Policy-based computing** refers to a software development model that incorporates a set of decision-making parameters into a separate management component (called *policy data store* or *policy-base*) in order to simplify and automate the administration of computer systems;
- **Policies** are items returned from a *policy data store* and used at runtime by application software. Examples of policies:
    - Password must use between 8 and 12 characters AND at least one letter and one digit.
    - Door unlocks at 7am and locks at 6pm
- Instead of hard coding the specific parameter values into the program, the values are stored in policy data store, which can be modified while the processing program is running.

Using <appSettings> vs. Hard Coding

- The purpose is to parameterize an application’s behavior, and to allow the behavior to be modified without changing & recompiling the source code.
- <**appSettings**> section holds the application specific values (strings) that can be read during execution.
- It is the basic form of policy-based computing.

Example: Hard-coded SQL access string

```
SqlDataAdapter adapter = new SqlDataAdapter
     (“select * from titles where price != 0”,
      “server=hawkeye; database=pubs; uid=sa; pwd=”);
DataSet ds = new DataSet ();
adapter.Fill (ds);
```

Hard coded

Hard coded

Hard coded

Example: Store Custom Setting in <AppSetting>

```
<appSettings>
        <add key= “xDataFile” value= “c:%myAspApps%Docs%xmlDoc=” />
</appSettings>


Public partial class myApp : System.Web.UI.Page
{
    protected void Page_Load( );
    {
        lblResult.Text = “Display data in xmlDoc here: <br />”;
        lblResult.Text += WebConfigurationManager.appSettings[“xDataFile”];
        lblResult.Text += “<br />”
    }
}
```

Supporting policy-based computing.

Using <appSettings> instead of Hard Coding

```
String conn =
ConfigurationSettings.appSettings["MyConnectionString"];
SqlDataAdapter adapter = new SqlDataAdapter
    ("select * from titles where price != 0", conn);
DataSet ds = new DataSet ();
adapter.Fill (ds);
```

Page_Load extracts the connection string from the Web.config file

```
<configuration>
    <appSettings>
        <add key= "MyConnectionString" value=
            "server=hawkeye;database=pubs;uid=sa;pwd=" />
    </appSettings>
</configuration>
```

<system.web> Element

system.web

- The *system.web* section of Web.config holds configuration settings used by the **system** -- ASP.NET.
- Its content is categorized by subsections. Developers are free to define custom subsections.
- The following subsections are supported by default and can be used without writing custom configuration handlers.

authentication	Sets the authentication mode and specifies settings for the mode selected
authorization	Specifies who is allowed to access resources in this directory and its subdirectories
browserCaps	Maps user-agent data to browser capabilities
clientTarget	Maps user-agent data to browser types
compilation	Specifies run-time compilation settings such as whether executables should be compiled with debug symbols, maps file name extensions and *Language* attributes to compilers, and identifies the assemblies that ASP.NET links to.

<system.web> Element -- continued

customErrors	Enables the use of custom error pages and specifies how errors should be reported on clients and servers
httpRuntime	Specifies request time-outs and other settings used by ASP.NET runtime
globalization	Specifies character encodings for requests and responses
httpHandlers	Maps URLs to HVFP handlers (for example, maps requests for ASPX files to *System. Web.UI.PageHandlerFactoiy*)
httpModules	Identifies HTTP modules called in response to HUP requests
identity	Controls the identity that ASP.NET assigns to individual requests
machineKey	Specifies encryption and validation settings (for example, the key and algorithm used to encrypt authentication cookies)
pages	Specifies page-level configuration settings such as whether output buffering, session state, and view state are enabled
processModel	Specifies configuration settings for ASP.NET worker processes
securityPolicy	Maps trust levels to CONFIG files containing security policies
sessionState	Specifies session state settings (e.g., where session state is stored)
trace	Enables and disables tracing and specifies trace settings
trust	Specifies the code access security trust level
webControls	Identifies the location on the server of client scripts used by ASP.NET Web controls
webServices	Contains Web service settings

Example of <system.web> Element

```
<configuration>
        <system.web>
                <trace enabled="true" />
        </system.web>
</configuration>
```

Empty element

```
<configuration>
   <system.web>
      <trace enabled="true" />
      <sessionState mode="SQLServer"
           sqlConnectionString="server=localhost;uid=sa;pwd="/>
      <compilation debug="true" defaultLanguage="c#" />
      <pages enableViewStateMac="true" />
   </system.web>
</configuration>
```

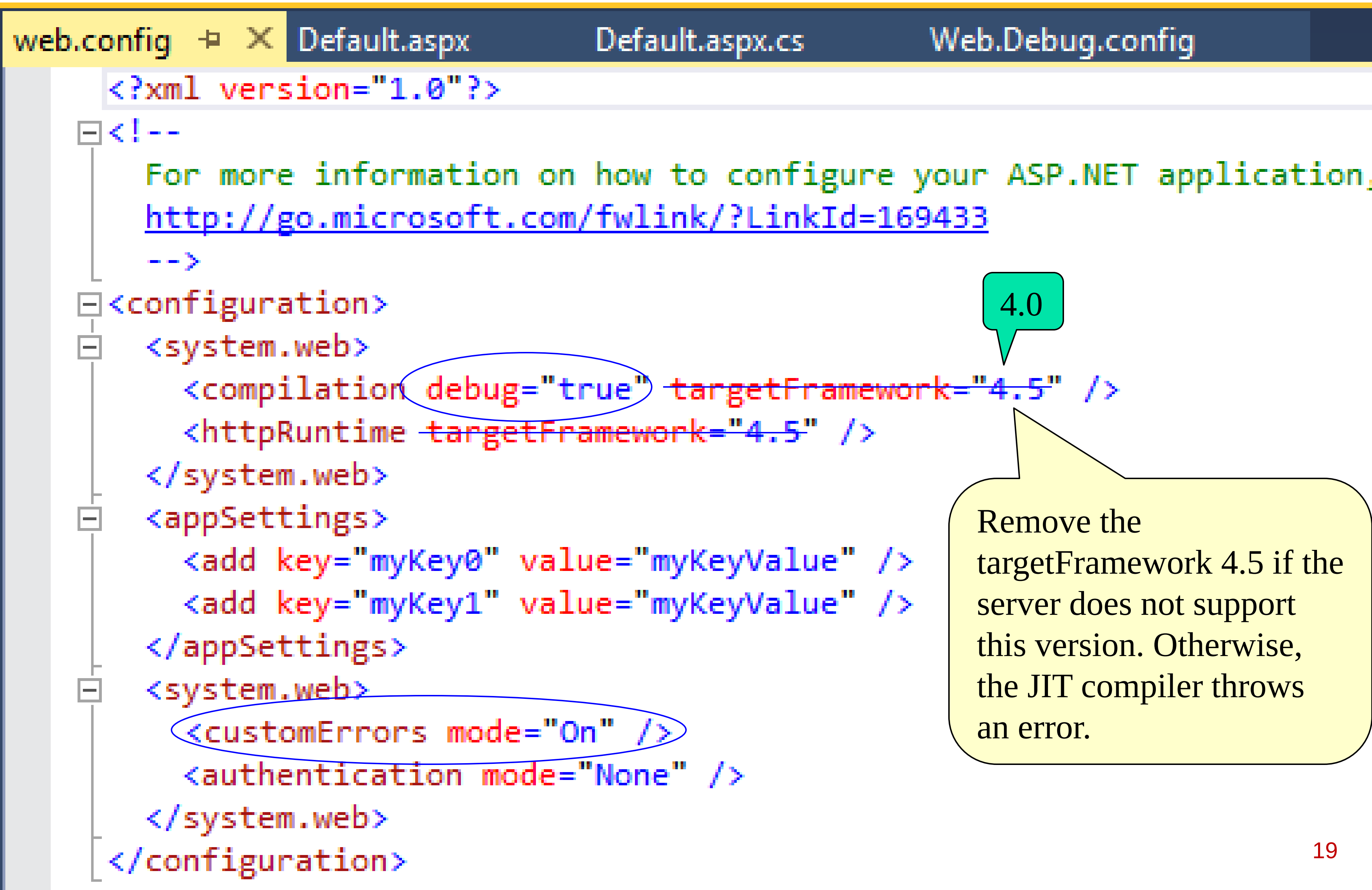
Example of <system.web> Element

```
<configuration>
   <system.web>

        <machineKey ... />

   </system.web>
</configuration>
```

The machineKey is normally generated automatically. However, if we use a Web Farm, the auto-generated key does not work. We need to change the setting, See WebStrar Tutorial.

https://support.microsoft.com/en-us/kb/2915218?wa=wsignin1.0#AppendixA

Debugging Information in Web.config File



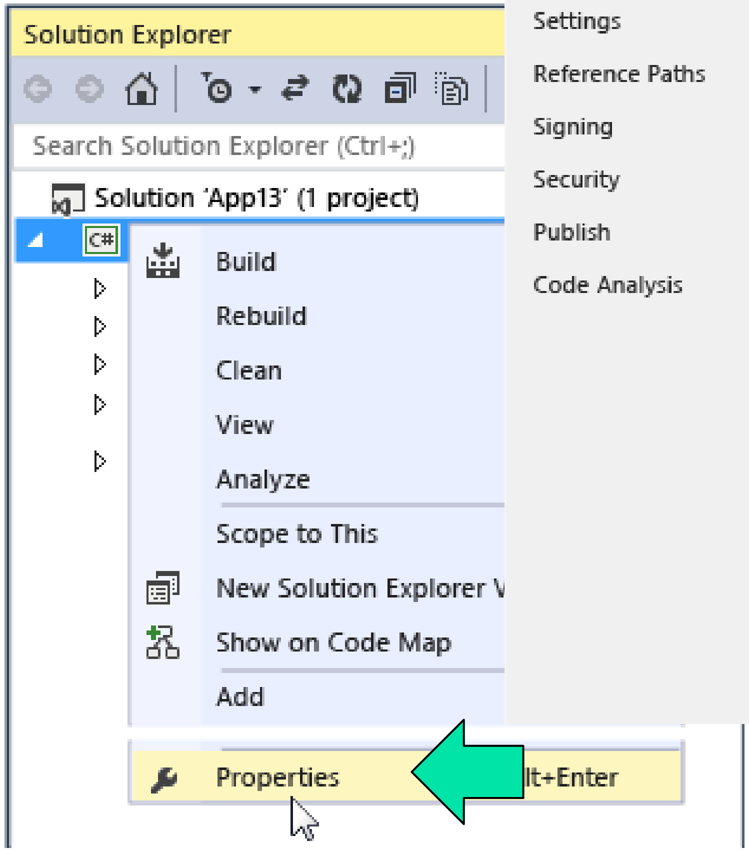
```
<?xml version="1.0"?>
<!--
  For more information on how to configure your ASP.NET application
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
  <system.web>
    <compilation debug="true" ~~targetFramework="4.5"~~ />
    <httpRuntime ~~targetFramework="4.5"~~ />
  </system.web>
  <appSettings>
    <add key="myKey0" value="myKeyValue" />
    <add key="myKey1" value="myKeyValue" />
  </appSettings>
  <system.web>
    <customErrors mode="On" />
    <authentication mode="None" />
  </system.web>
</configuration>
```

4.0

Remove the targetFramework 4.5 if the server does not support this version. Otherwise, the JIT compiler throws an error.

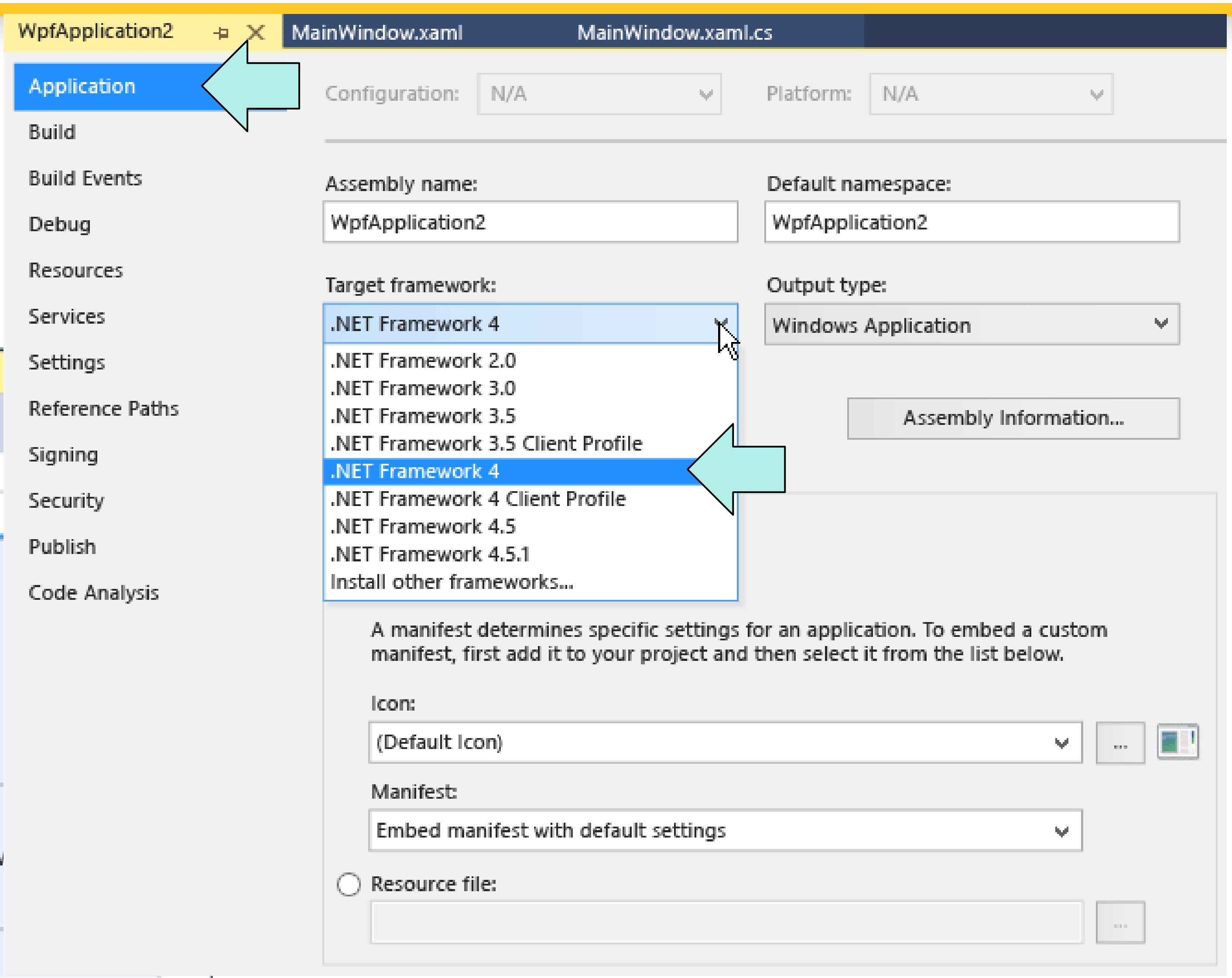
Define Your targetFramework

Right-Click Solution


Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'App13' (1 project)



Configuration: N/A Platform: N/A

Assembly name: WpfApplication2

Default namespace: WpfApplication2

Target framework: .NET Framework 4

Output type: Windows Application

.NET Framework 2.0
.NET Framework 3.0
.NET Framework 3.5
.NET Framework 3.5 Client Profile
.NET Framework 4
.NET Framework 4 Client Profile
.NET Framework 4.5
.NET Framework 4.5.1
Install other frameworks...

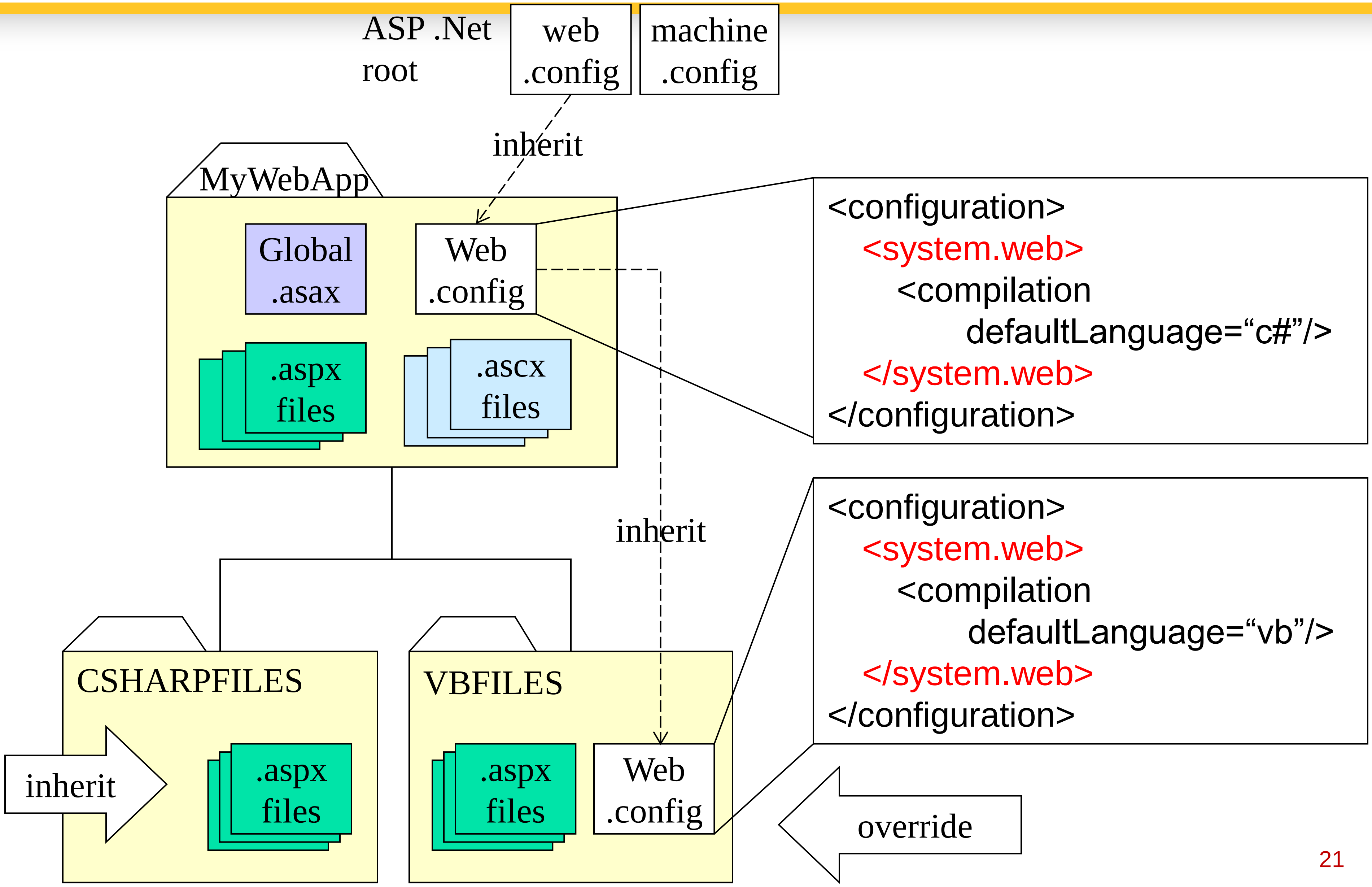
A manifest determines specific settings for an application. To embed a custom manifest, first add it to your project and then select it from the list below.

Icon: (Default Icon)

Manifest: Embed manifest with default settings

Resource file:

Web Configuration File Inheritance

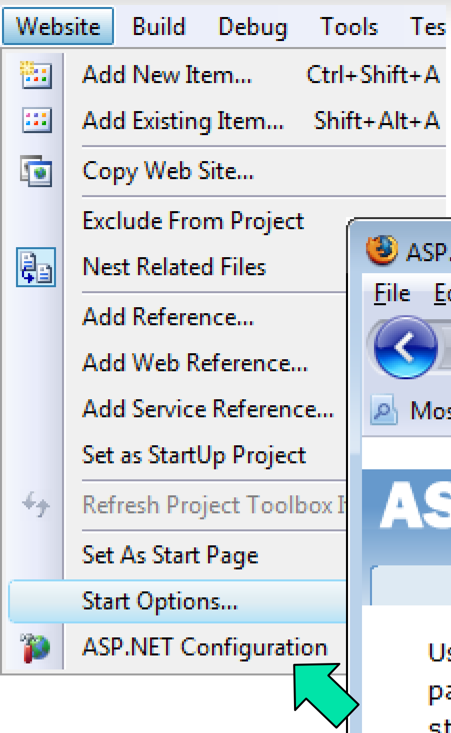


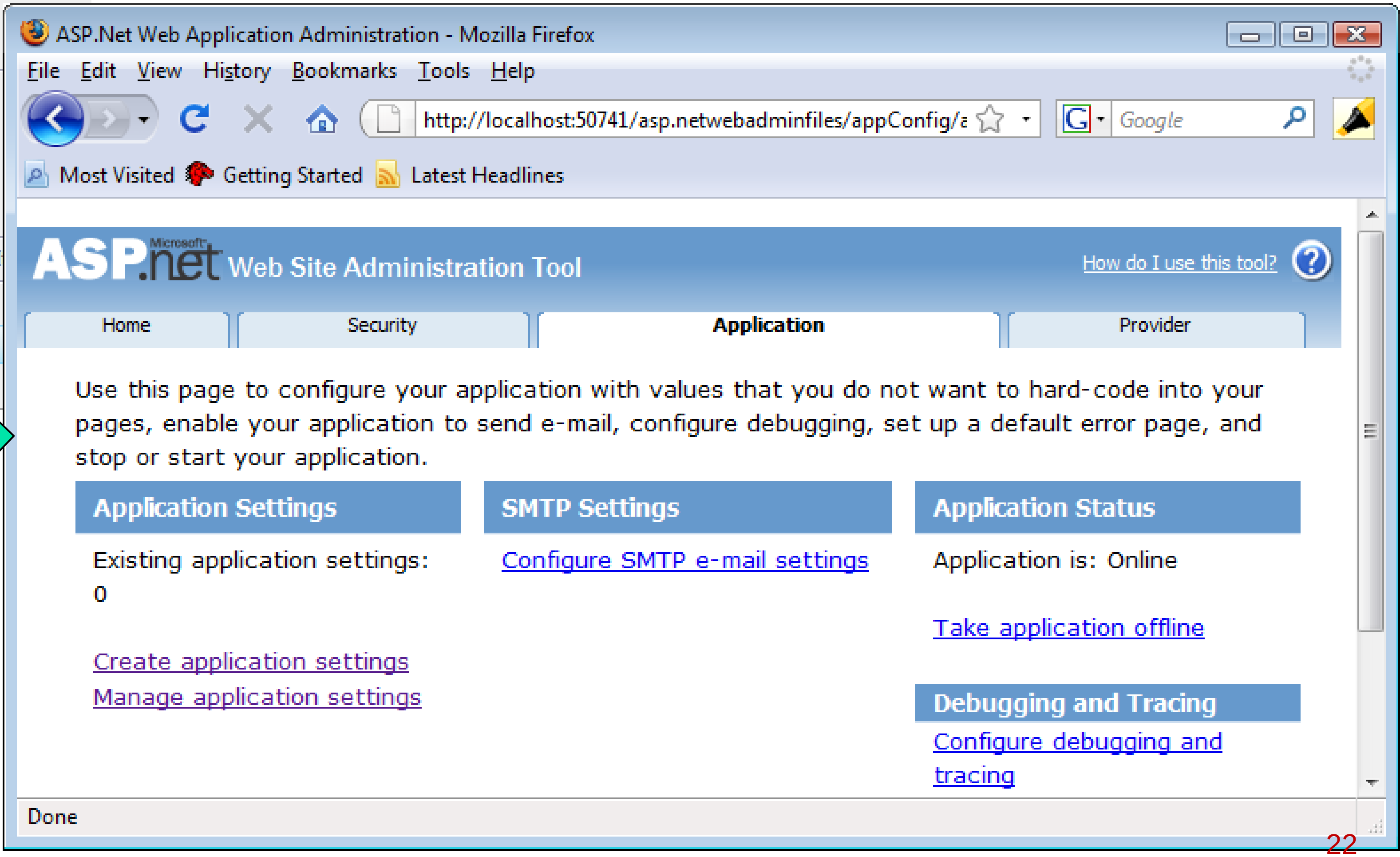
```
<configuration>
   <system.web>
      <compilation
            defaultLanguage="c#"/>
   </system.web>
</configuration>
```

```
<configuration>
   <system.web>
      <compilation
            defaultLanguage="vb"/>
   </system.web>
</configuration>
```

Using ASP.Net Website Admin Tool

In VS, select Website → ASP.Net Configuration




ASP.Net Web Application Administration - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:50741/asp.netwebadminfiles/appConfig/a
Google
Most Visited
Getting Started
Latest Headlines
ASP.net Web Site Administration Tool
How do I use this tool?
Home
Security
Application
Provider

Use this page to configure your application with values that you do not want to hard-code into your pages, enable your application to send e-mail, configure debugging, set up a default error page, and stop or start your application.

Application Settings

Existing application settings: 0

Create application settings

Manage application settings

SMTP Settings

Configure SMTP e-mail settings

Application Status

Application is: Online

Take application offline

Debugging and Tracing

Configure debugging and tracing

Done

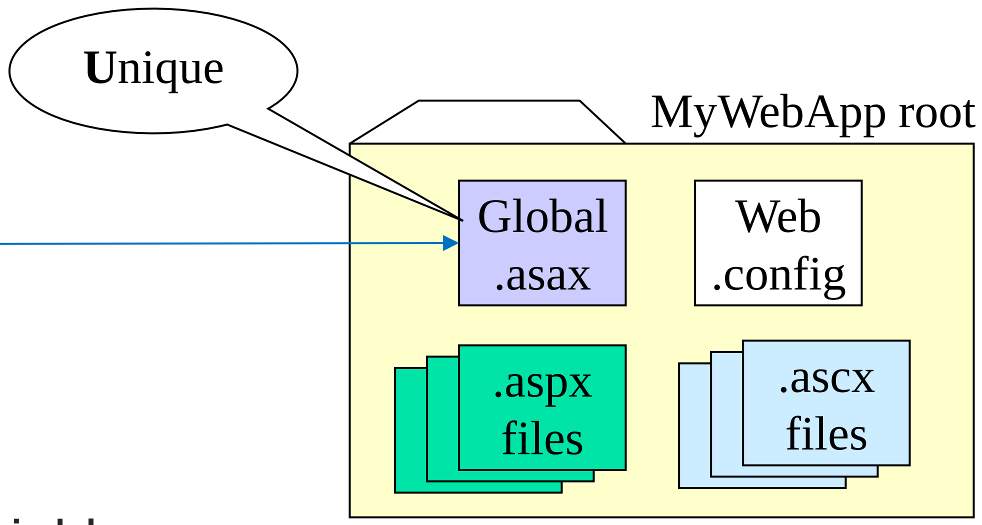
Ira A. Fulton Schools of Engineering

Arizona State University

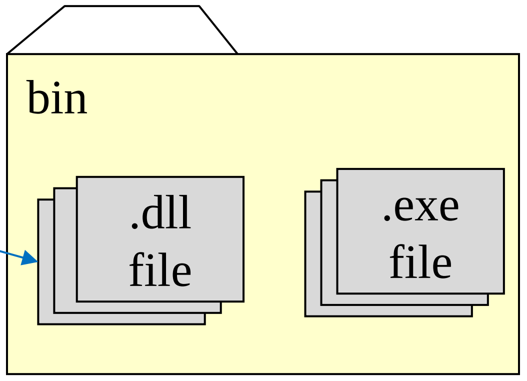
M12 L2
Global and DLL

Lecture Outline

- **Global File**
  - Global Directives
  - Global Event Handler
  - Global Objects and Variable



- **DLL: Dynamic Linking Library**



Global.asax File

Global.asax is a text file that houses global directives, application-level event handlers, declarations that apply to all parts of the application, and other global application elements.

- Global directives
  - @ *Application* directives
  - @ *Import* directives
  - @ *Assembly* directives
- Global event handlers: particularly important and are the main reason why developers include Global.asax files in their applications
- Global objects (variables)

@ Application Directives in Global.asax

- *@ Application* directives serve two purposes:
  - Enable developers to add descriptive text (comments) to applications, and
  - Facilitate programming in Global.asax files.

```
<%@ Application Description = “MY ASP.NET Application with Global Directives” %>
<%@ Import Namespace= “System.Data” %>
<script language=“C#” runat=“server”>
      void Application_Start ()
      {
            DataSet ds = new DataSet ();
            ds.ReadXml (Server.MapPath (“GlobalData.xml”));
            Application[“GlobalData”] = ds;
      }
</script>
```

Support DataSet

This code will be compiled at runtime:
Pro: flexibility
Con: cold start

@ Application Directives in Global.asax

- Write the code as a C# program (.cs) and pre-compile the code into .DLL file, e.g., MyStarter.dll

```
using System.Web;
using System.Data;
public class MyStarter : HttpApplication
{
        void Application_Start ()
        {
                DataSet ds = new DataSet ();
                ds.ReadXml (Server.MapPath ("GlobalData.xml"));
                Application["GlobalData"] = ds;
        }
}
```

- In Global.asax file, use this line to invoke the program

```
<%@ Application inherits = "MyStarter" %>
```

Global Event Handlers in Global.asax

- ASP.NET fires global events named *Start* and *End*, when an application starts and stops. To process these events, include handlers named *Application_Start* and *Application_End* in Global.asax:

```
<script language="C#" runat= "server">
    void Application_Start()
    {
        . . . // Display Welcome message, initialization, …
     }
     void Application_End()
     {
        Response.Write("<hr />This page was last accessed
        at " + DateTime.Now.ToString());
     }
</script>
```

Global.asax file Example for Web API

```
using System;
using System.Web.Http;
using HelloWebAPI.Configuration;

namespace MyWebApi
{
    public class Global : System.Web.HttpApplication
    {
        protected void Application_Start(object sender, EventArgs e)
        {
            GlobalConfiguration.Configure(HelloWebAPIConfig.Register);
        }
    }
}
```

Application_Start event handler:
When the application starts, it runs the code to register the api.

Additional Application Events

- There are many other application events that you can handle by writing your handlers:

Event/handler	**Description**
Application_Start( )	Called the beginning of the application
Application_End( )	Called the end of the application
Session_Start( )	Called the beginning of the session
Session_End( )	Called the end of the session
Application_Error( )	Called when an unhandled error occurs

Per Request Event Handlers

Application_BeginRequest( )	Called at the beginning of each request the appl. received, before the page is executed;
Application_EndRequest	Called after the page is executed at the end of each request the application received;
Application_AuthenticateRequest( )	Called to authenticate the caller
Application_AuthorizeRequest( )	Called to determine whether the caller is authorized to access the requested resource
Application_ResolveRequestCache( )	Called to resolve the current request by providing content from a cache
Application_AcquireRequestState( )	Called to associate the current request with a session and populate session state
Application_ReleaseRequestState( )	Called to release (store) any State associated with this session
Application_UpdateRequestCache( )	Called to update a cache with content returned in the response

Global Object / Variable in Global.asax

- A global object/variable can facilitate the communication among the
  - Different sessions from different clients
  - Pages within the same session (There are other better ways for this purpose: session state)
- Need to address the monitoring/synchronization issues, as we discussed in text Chapter 2.

Global Object / Variable

In Global.asax file

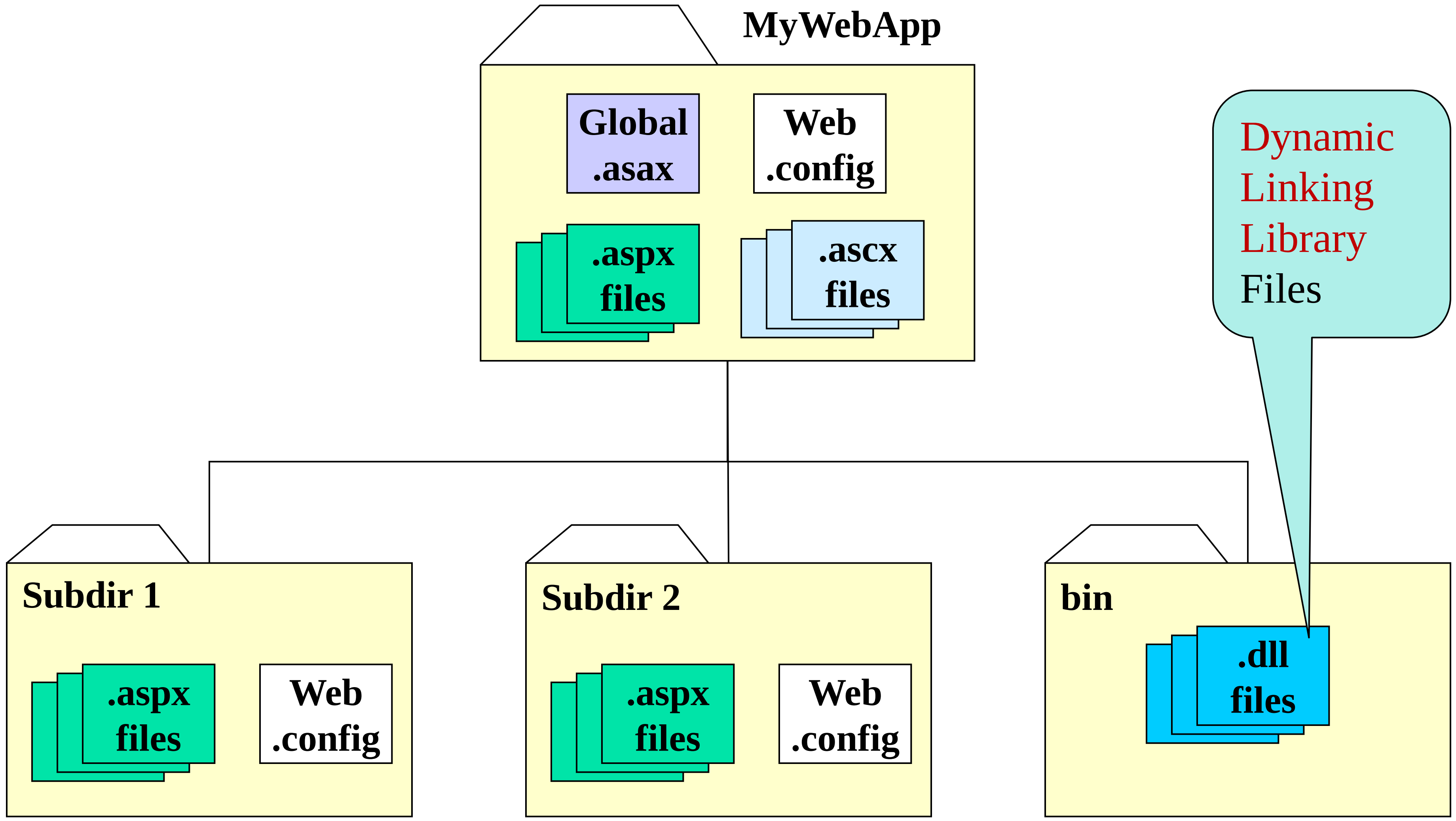
```
<script language="C#" runat="server">
    public static Int32 globalCounter = 0;
    </script>
```

Challenge 1: Simultaneous write?

```
<script language="C#" runat="server">
    private static Int32 globalCounter = 0;
    public void increment(Int32 newValue) {
        lock(this) {
            globalCounter = globalCounter + newValue;
        }
</script>
```

Challenge 2: Performance?

DLL Files in an ASP.Net Web Application

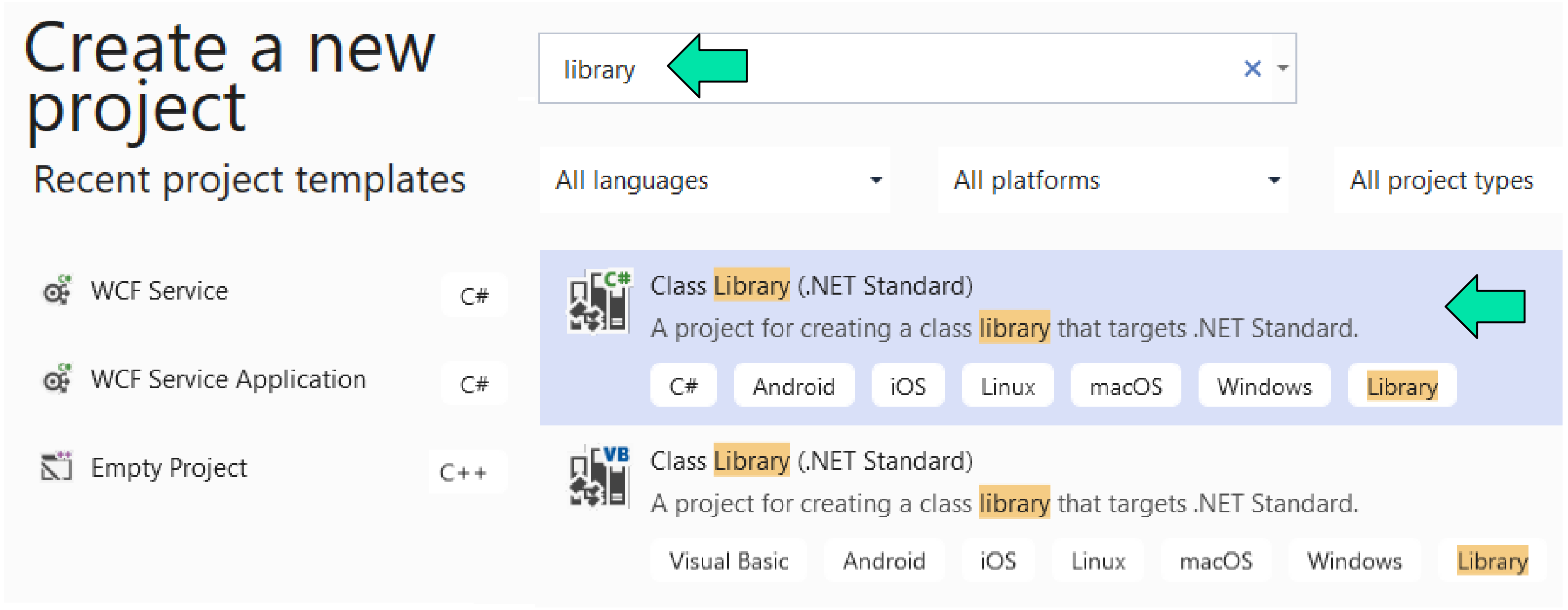


Creating DLL Components

- An application can contain multiple classes (pages);
- The code of the classes (aspx.cs files) are not reusable in other applications;
- The .aspx.cs file provides event handlers for the controls in aspx page;
- In order to reuse the code, you can make this component a service – a remote component;
- You can also create your own DLL library to collect all your reusable classes. They are local components and have better performance.
- Your library will form a namespace;
- Include your library in your application.

Creating a Class Library Project

Source: https://learn.microsoft.com/en-us/dotnet/core/tutorials/library-with-visual-studio?pivots=dotnet-7-0



Create a new project

library

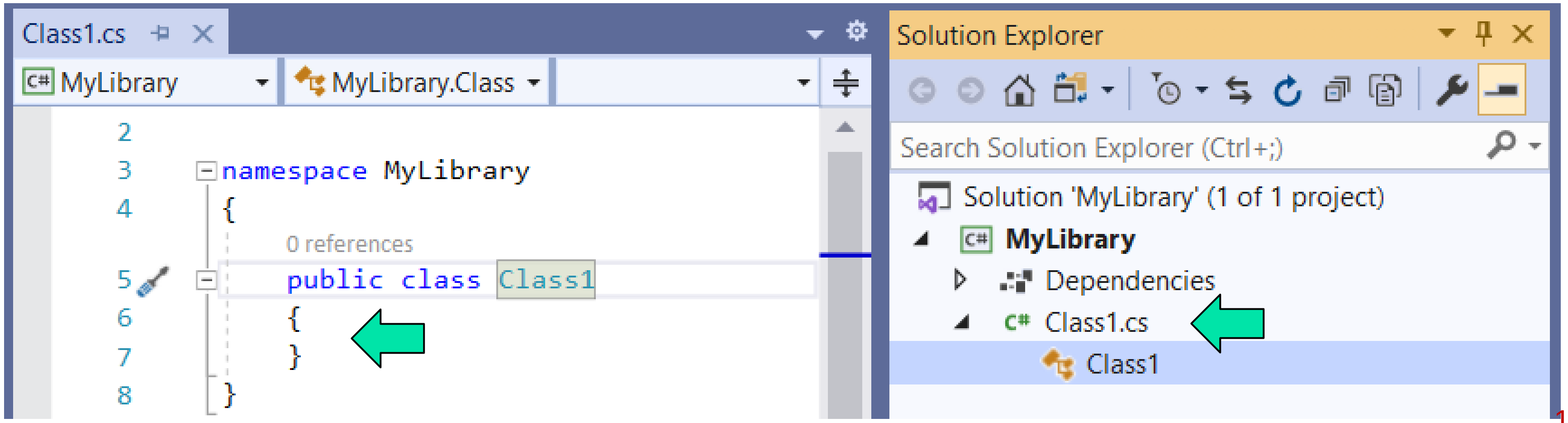
Recent project templates

WCF Service	C#
WCF Service Application	C#
Empty Project	C++

All languages | All platforms | All project types

Class Library (.NET Standard)
A project for creating a class library that targets .NET Standard.
C# Android iOS Linux macOS Windows Library

Class Library (.NET Standard)
A project for creating a class library that targets .NET Standard.
Visual Basic Android iOS Linux macOS Windows Library



Class1.cs

MyLibrary | MyLibrary.Class

```
namespace MyLibrary
{
    0 references
    public class Class1
    {
    }
}
```

Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'MyLibrary' (1 of 1 project)
  - MyLibrary
    - Dependencies
    - Class1.cs
      - Class1

Add the DLL File into your Web Project

- Create a new project of type "Class Library";
- Create a .cs page of classes that you will be using in your other applications.
- After classes are created and compiled, go back to your website project and do "Add Reference";
- Browse to the class library project, find the library, and add it to your application project.
- A copy of the DLL file will be copied and pasted in the "bin" directory in your application folder.
- Note, you must use the same .Net Framework version, e.g., 4.72, to create the DLL library project and the Application project.

Create myLibrary in Class Library Project

```
namespace myLibrary{
Class 1 →   public class TemperatureConversion {
        public static Int32 getFahrenheit(Int32 c)  {
            Double f = c * 9 / 5 + 32;
            return Convert.ToInt32(f);
        }
            public static Int32 getCelsius(Int32 f) {
            Double c = (f - 32) * 5 / 9;
            return Convert.ToInt32(c);
        }
    }
Class 2 →   public class myMath {
        public static long abs (long x) {
            if (x >= 0) return (x); else return (-x);
        }
    }
}
```

Use the Functions in myLibrary

Include myLibrary

Add Reference to copy the code into the application.

Class Name

Method Name

```
using myLibrary;
class myApplication {
    static void Main(string[ ] args)  {
    Int32 Ctemp = 23;
    Int32 Ftemp = 121;
    double x = TemperatureConversion.getFahrenheit(Ctemp);
    double y = TemperatureConversion.getCelsius(Ftemp);
    System.Console.WriteLine("C-temp {0}  is F-temp {1}", Ctemp , x);
    System.Console.WriteLine("F-temp {0}  is C-temp {1}", Ftemp , y);
   }
}
```

```
double x = myLibrary.TemperatureConversion.getFahrenheit(Ctemp);
```

Wrapping Legacy Software into Web Service

- There are many useful software components are developed before Web service standards;
- They are in the form of library classes and functions, such as DLL classes and functions;
- To wrap a library class into a service:
  - Use a Web service template to start service development;
  - Add Reference and load a library class into your service;
  - Use the library class to implement your service;
  - After you deploy the service, the library class becomes a service;
  - You may need re-implement a number of mechanisms, such as input, output, and state management.

Ira A. Fulton Schools of Engineering

Arizona State University

M12 L3
Web State Management: Cookies

Ira A. Fulton Schools of Engineering
Arizona State University

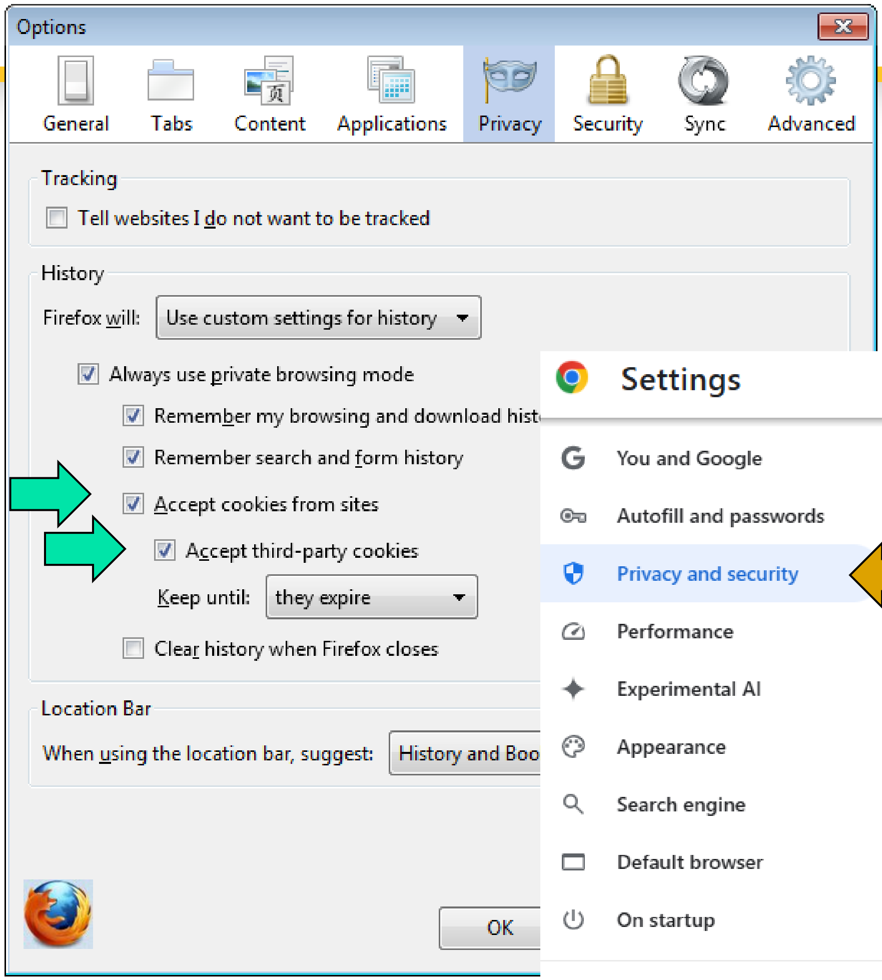
Lecture Outline

- What are Cookies
- Saving and Retrieving Cookies
- Cookie Application

Cookies: Storage on Client Computer

- Cookies provide a way of storing user’s information
  - in the browser (temporary, disappear after closing browser)
  - on the hard drive of client’s computer (longer term)
- Cookies are transparent to the users, as long as the cookies are enabled in the browser;
- Cookies can store string type of data only, often used for storing user’s preferences of the application;
- Other data types need to be converted to strings;
- The syntax of Cookies are similar to View State.

Configure Your Browser to Enable Cookies



Tracking

[ ] Tell websites I do not want to be tracked

History

Firefox will: Use custom settings for history

[✓] Always use private browsing mode

[✓] Remember my browsing and download hist

[✓] Remember search and form history

[✓] Accept cookies from sites

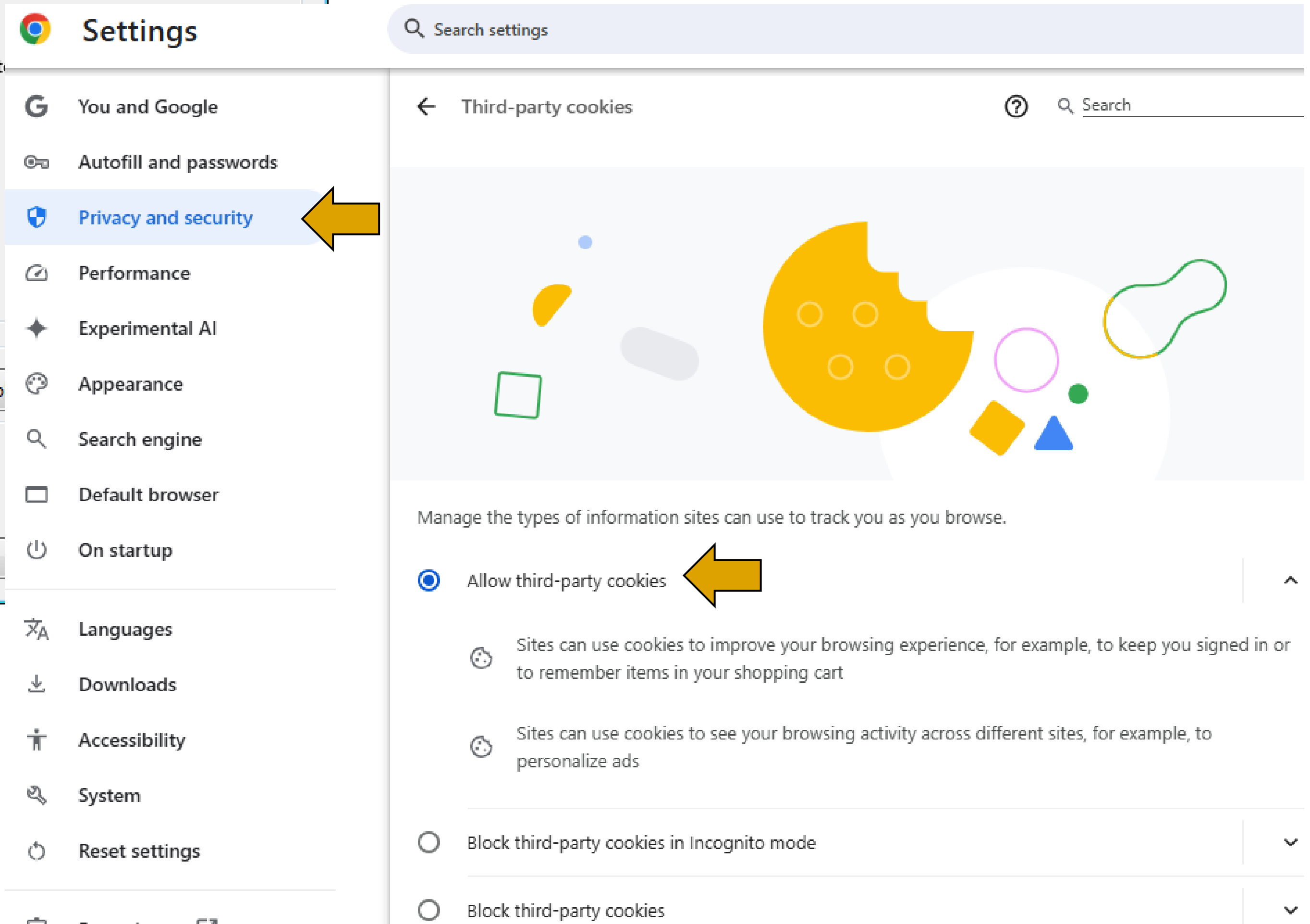
[✓] Accept third-party cookies

Keep until: they expire

[ ] Clear history when Firefox closes

Location Bar

When using the location bar, suggest: History and Boo



Manage the types of information sites can use to track you as you browse.

● Allow third-party cookies

Sites can use cookies to improve your browsing experience, for example, to keep you signed in or to remember items in your shopping cart

Sites can use cookies to see your browsing activity across different sites, for example, to personalize ads

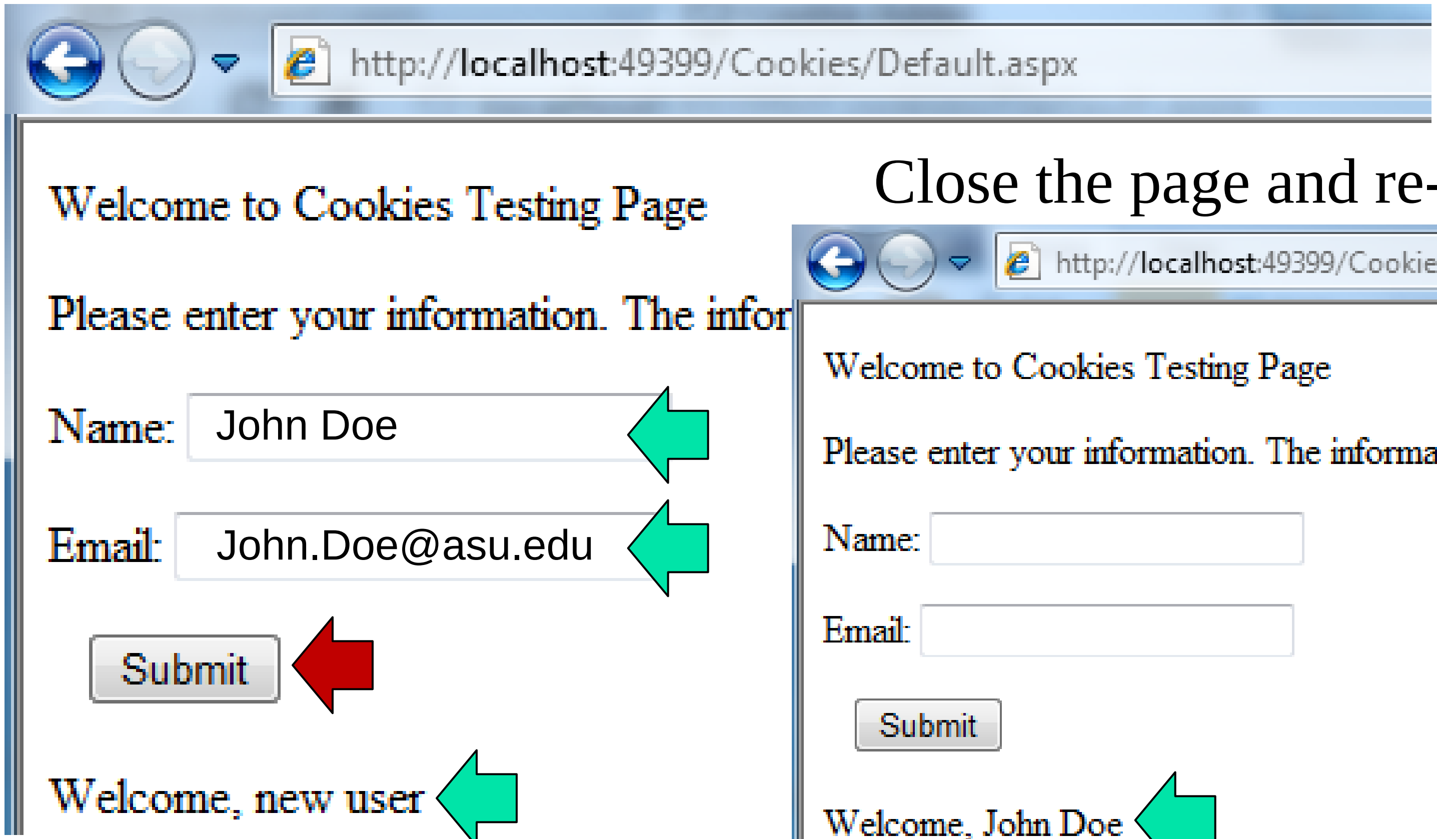
○ Block third-party cookies in Incognito mode

○ Block third-party cookies

Testing a Website with cookies: (1) Enter data (2) Close Browser (3) Reopen

http://venus.sod.asu.edu/WSRepository/CookiesTest/

Demo



http://localhost:49399/Cookies/Default.aspx

Welcome to Cookies Testing Page

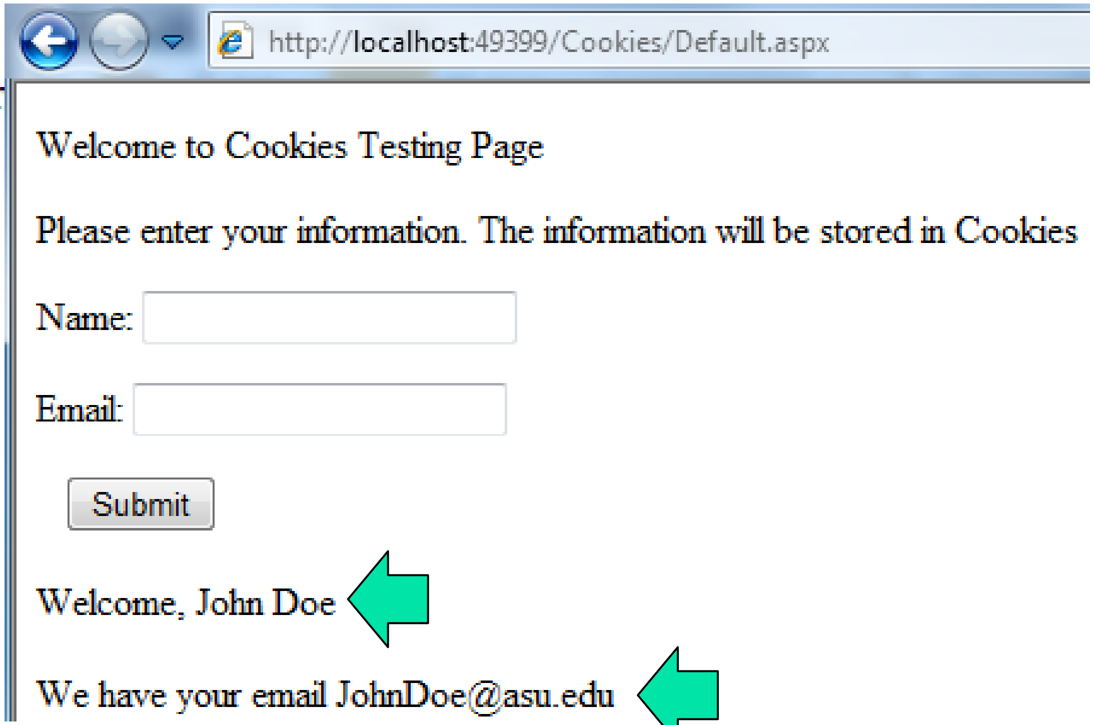
Please enter your information. The infor

Name: John Doe

Email: John.Doe@asu.edu

Welcome, new user

Close the page and re-open the page



http://localhost:49399/Cookies/Default.aspx

Welcome to Cookies Testing Page

Please enter your information. The information will be stored in Cookies

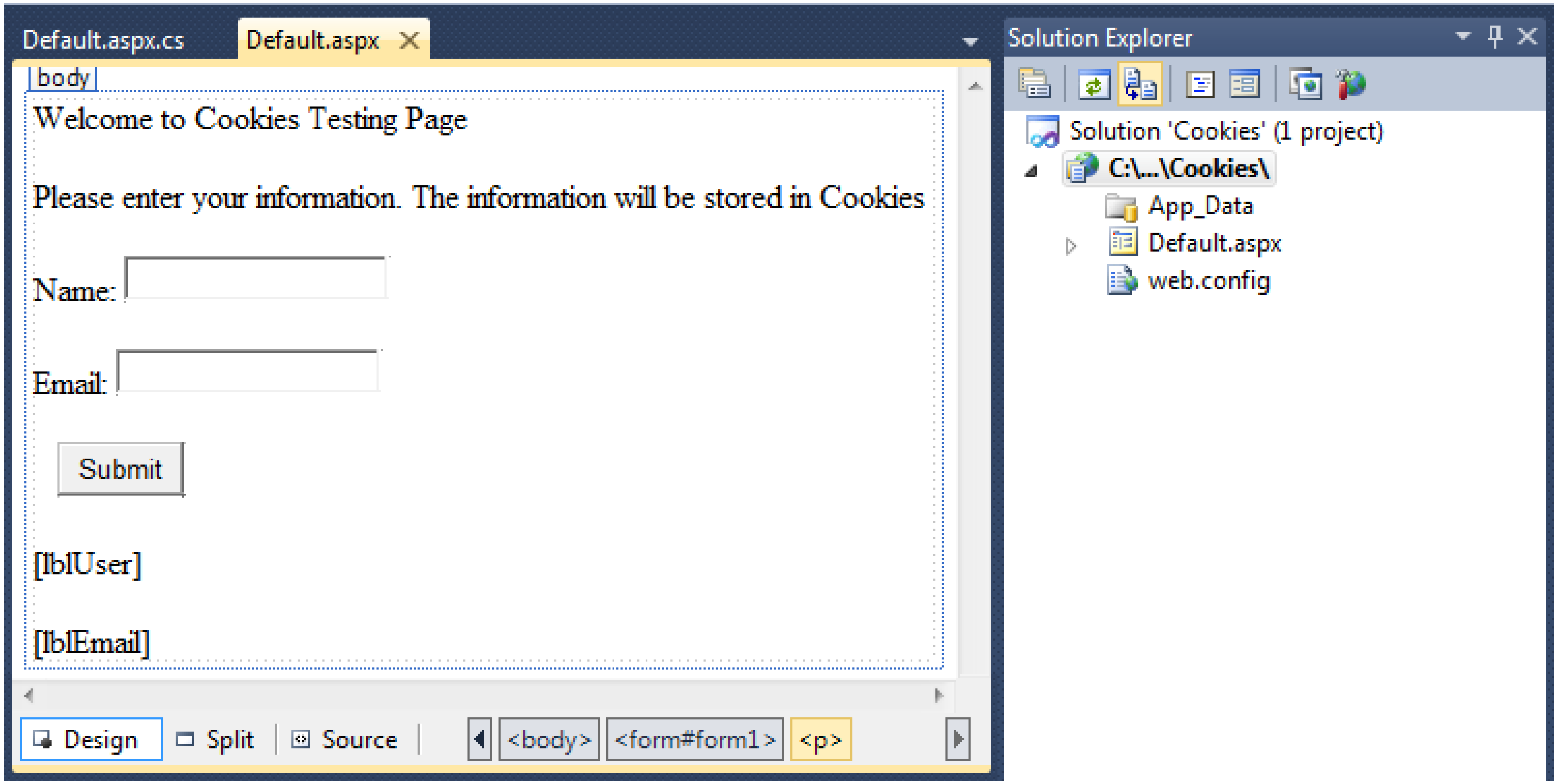
Name:

Email:

Welcome, John Doe

We have your email JohnDoe@asu.edu

Page Design


Welcome to Cookies Testing Page

Please enter your information. The information will be stored in Cookies

Name:

Email:

Submit

[lblUser]

[lblEmail]

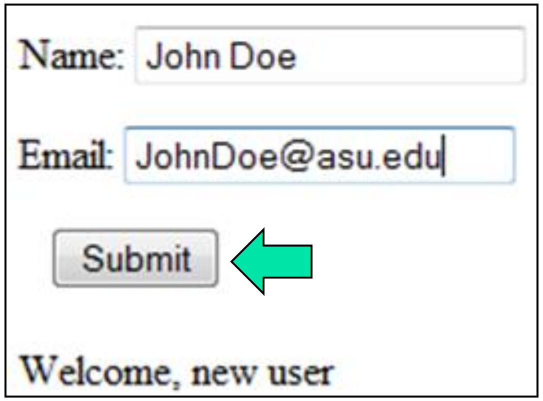
Code Saving and Retrieving Cookies

```
    protected void  Button1_Click(object sender, EventArgs e)
    {
        HttpCookie myCookies = new HttpCookie("myCookieId");
        myCookies["Name"] = TextBox1.Text;
        myCookies["Email"] = TextBox2.Text;
        myCookies.Expires = DateTime.Now.AddMonths(6);
        Response.Cookies.Add(myCookies);
        lblUser.Text = "Name stored in cookies " + myCookies["Name"];
        lblEmail.Text = "Email stored in cookies " + myCookies["Email"];
   }
}
```

Create a new cookie object with a key

Add my content to the cookies collection

Read the content and display at the given label



Name: John Doe

Email: JohnDoe@asu.edu

Welcome, new user

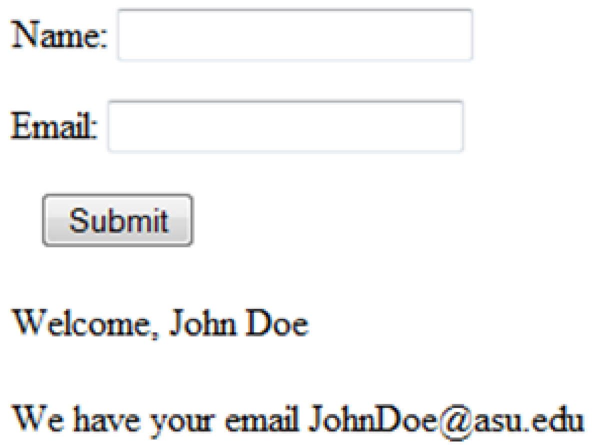
Code Saving and Retrieving Cookies

```
using System.Net; // needed for Cookies
public partial class _Default : System.Web.UI.Page  {
    protected void Page_Load(object sender, EventArgs e) {
        HttpCookie myCookies = Request.Cookies["myCookieId"];
        if ((myCookies == null) || (myCookies["Name"]=="")) {
           lblUser.Text = "Welcome, new user";
    } else {
        lblUser.Text = "Welcome, " + myCookies["Name"];
        lblEmail.Text = "We have your email " + myCookies["Email"];
        }
    }
```

Access a cookie object using a key

Check if the cookie exist or is empty

Read cookie content and display at given label

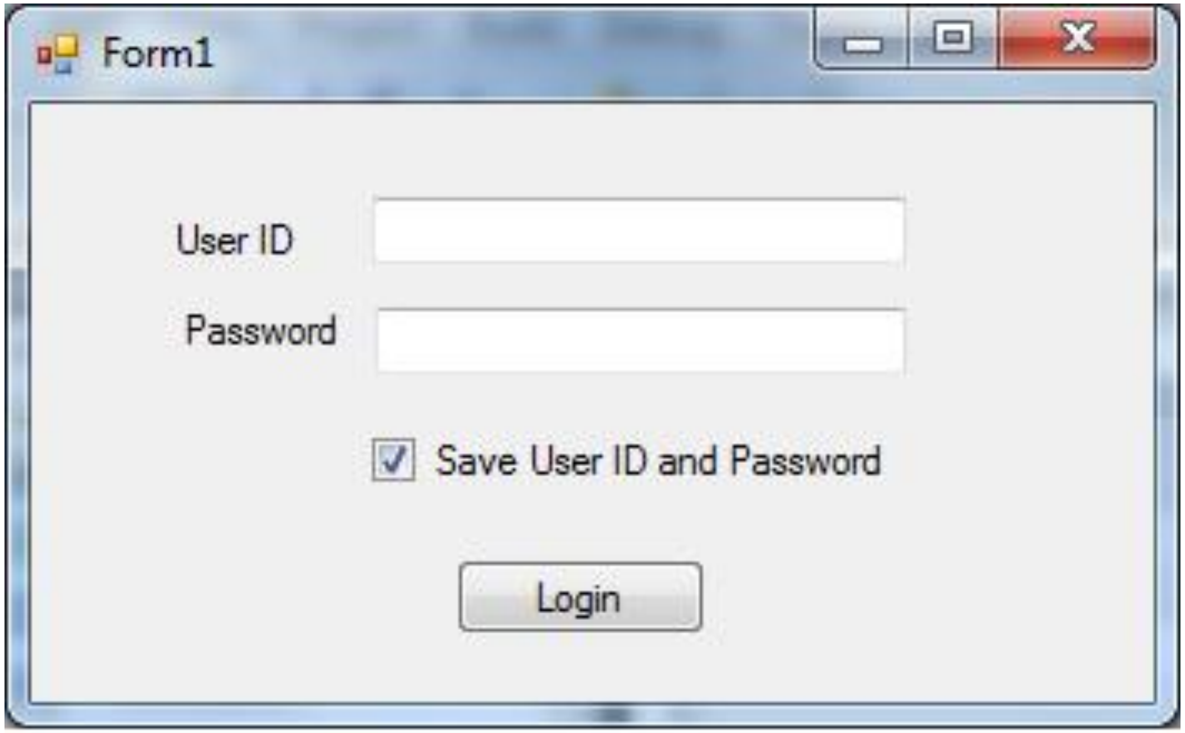


Welcome, John Doe

We have your email JohnDoe@asu.edu

Application of cookies in Login

- Cookies are often used in saving the credentials



Code behind the Login Button

```
namespace LoginCookie {
    public partial class LoginPage : Form {
        public LoginPage()
        { InitializeComponent(); }
        private void btnLogin_Click(object sender, EventArgs e)  {
            if ((txtUserId.Text != "") && (txtPassword.Text != ""))
             {
                 String c = txtUserId.Text + " " + txtPassword.Text;
                 FormsAuthentication.RedirectFromLoginPage(c, ckbChecked);
             }
            else
                Output.Text = "Invalid login, try again";
        }
    }
}
```

This method will compare the user ID and the password saved in Web.config file. To discuss in Chapter 6.

FormsAuthentication.RedirectFromLoginPage

```
FormsAuthentication.RedirectFromLoginPage
(txtUserId.Text, Persist);
```

→ true: create cookies to save the credential

→ false: no cookies for the credential

Applications of cookies will be further discussed in later section and in Chapter 6 on security

Are Cookies Browser Dependent?

- They are stored in a browser-specified location and thus, the browser will search that location only. Typically, the browser application folder;
- Cookies are normally not available cross browsers;
- However, cookies are stored in standard format and can be transferred between the browsers.
  - When you start to use a new browser, you often receive this question: Do you want to copy your user profiles from X browser?

Are Cookies Secure?

- Cookies are stored in local hard drive. It is as secure as other data on your computer;
- Cookies (e.g., username and password) will be sent from your local computer to the server for validation when you login. It is not secure during the transmission. However, if you enter the username and password, they are not secure either.
- The only solution is to have SSL connection (https). Chapter 6 will discuss how can you install SSL to enable secure connection between client and server.
- HttpCookie class has a "Secure" property for user to check if SSL is available:

```
if (MyCookie.Secure)
{
        // Use cookies, otherwise not
}
```

Ira A. Fulton Schools of Engineering

Arizona State University

M12 L4
Web State Management:
Session State

Ira A. Fulton Schools of Engineering

Arizona State University

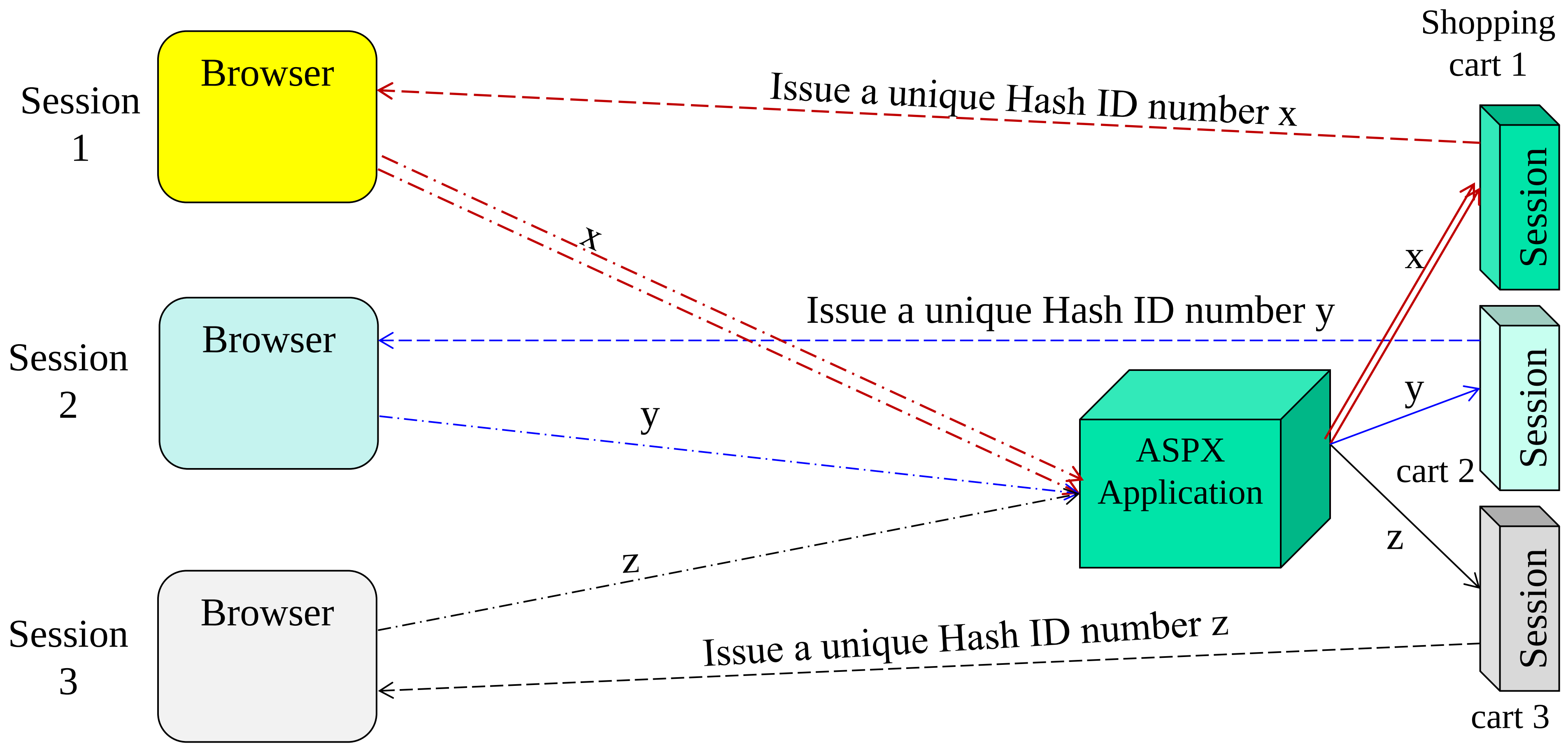
Lecture Outline

- What is Session State?
- Case Study: Creating a Shopping Site
- Cookie Support to Session State
- From Session State to Application State

Session State

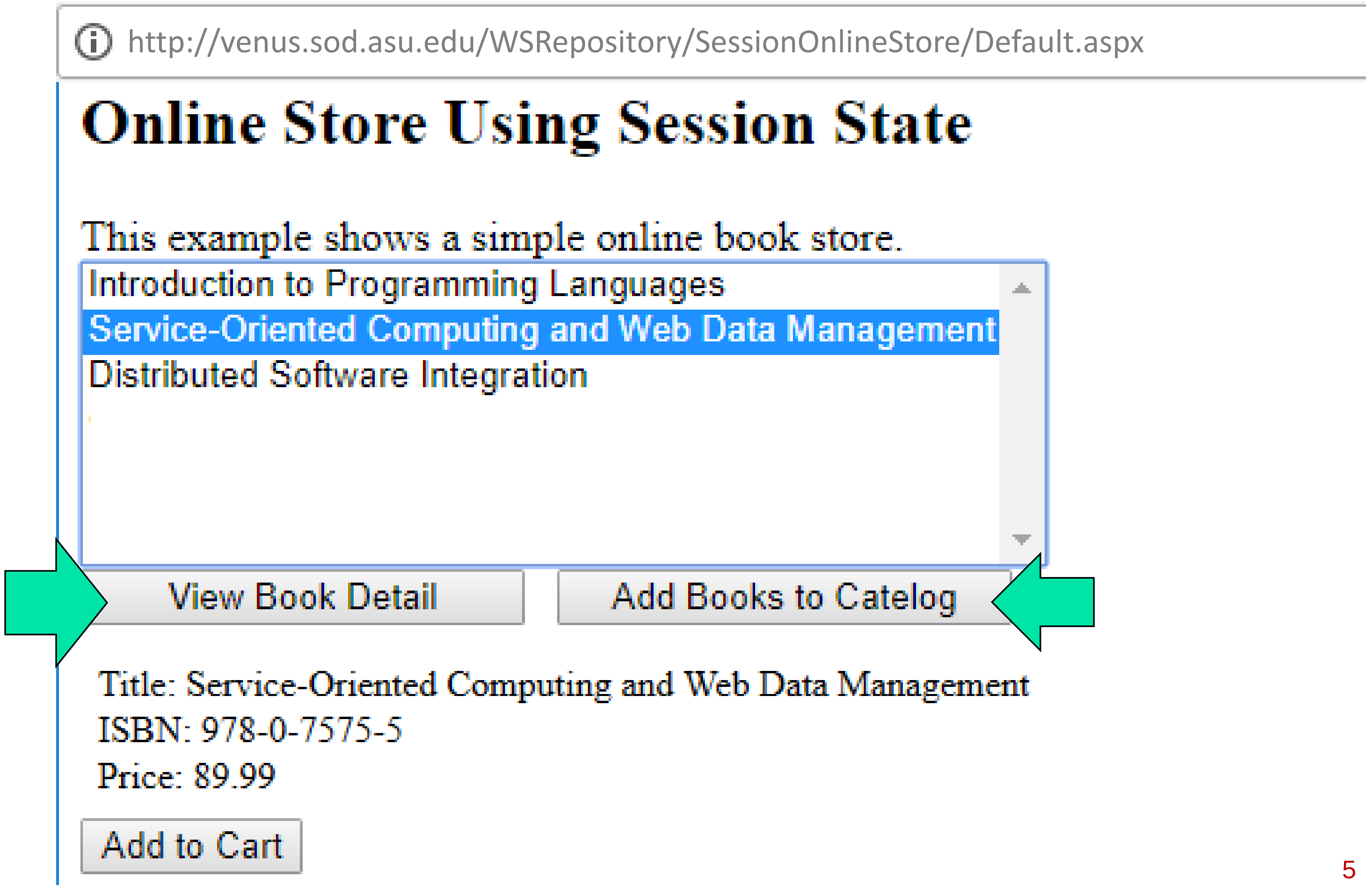
- View State and Cookie can store string data on client side:
  - View State: in hidden fields in html page in web browser
  - Cookie: in client machine's hard drive
- Session state allows you to store structured objects in the server;
- The scope of session state is within all pages of the session, but not cross different sessions;
- The syntax is similar to ViewState and Cookies
- The information in Session state is secure
  - The information is linked to the session. Other sessions of the same application cannot access the information;
  - A unique 120-bit hash number is generated to associate the user to the session: The number is sent to user as an id, and the user session must carry the ID in order to revisit the session. Try the service that can generate the ID: http://venus.sod.asu.edu/WSRepository/Services/HashSha512/Service.svc

Understanding Session and State



Use Session State To Store Objects

http://venus.sod.asu.edu/WSRepository/SessionOnlineStore/Default.aspx



http://venus.sod.asu.edu/WSRepository/SessionOnlineStore/Default.aspx

Online Store Using Session State

This example shows a simple online book store.

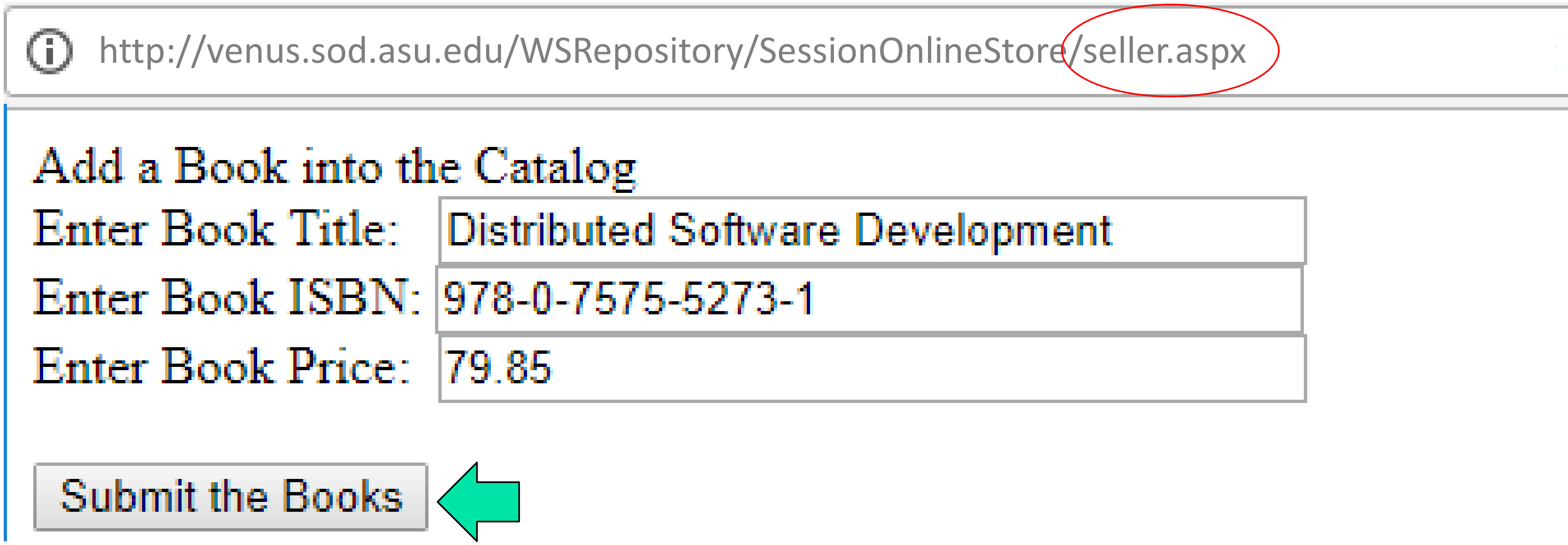
Introduction to Programming Languages
Service-Oriented Computing and Web Data Management
Distributed Software Integration
View Book Detail
Add Books to Catelog

Title: Service-Oriented Computing and Web Data Management
ISBN: 978-0-7575-5
Price: 89.99

Seller Page for Entering Information

Seller.aspx

http://venus.sod.asu.edu/WSRepository/SessionOnlineStore/seller.aspx

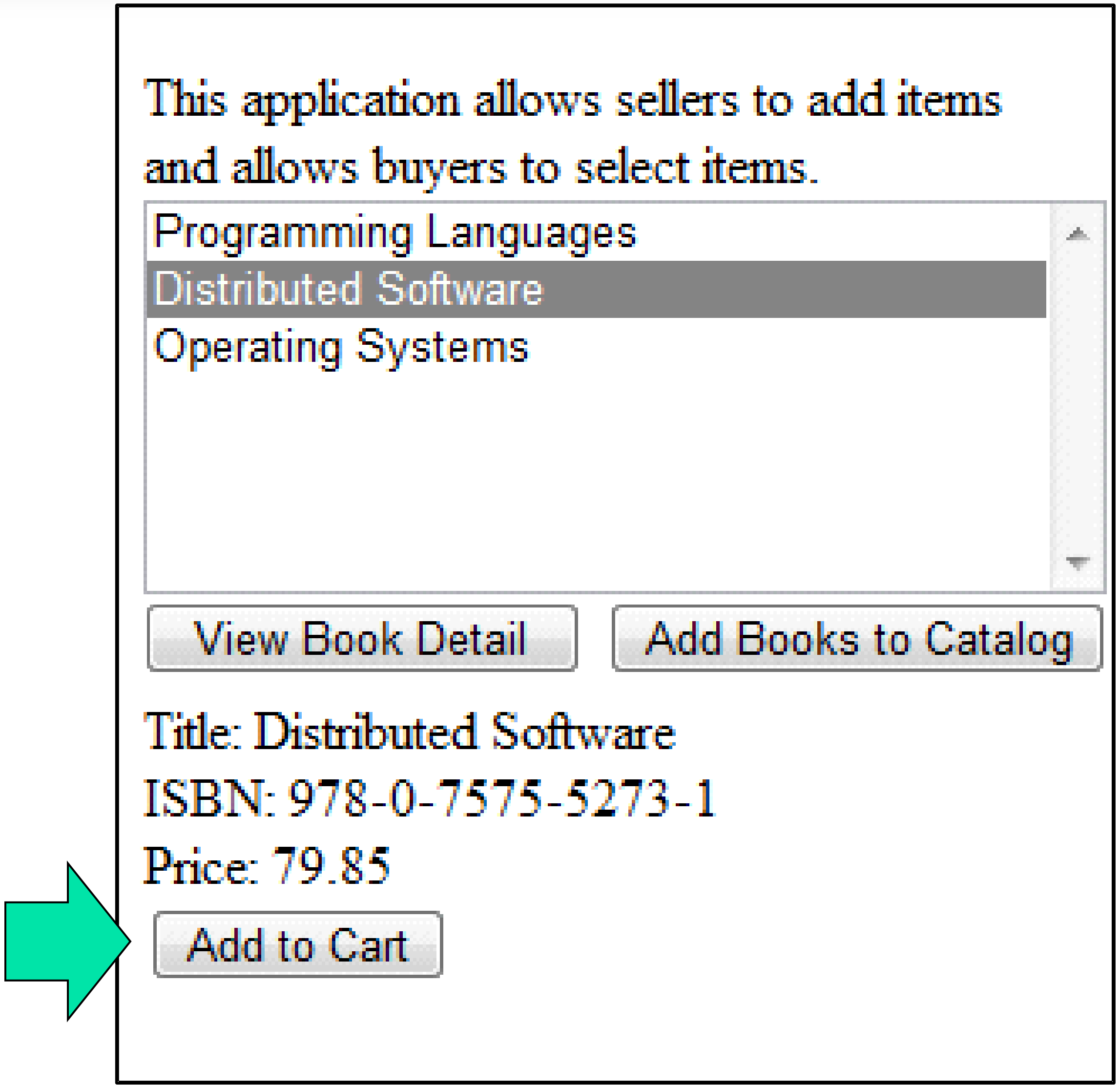


http://venus.sod.asu.edu/WSRepository/SessionOnlineStore/seller.aspx

Add a Book into the Catalog

Enter Book Title:	Distributed Software Development
Enter Book ISBN:	978-0-7575-5273-1
Enter Book Price:	79.85

Returns to Default.aspx Page

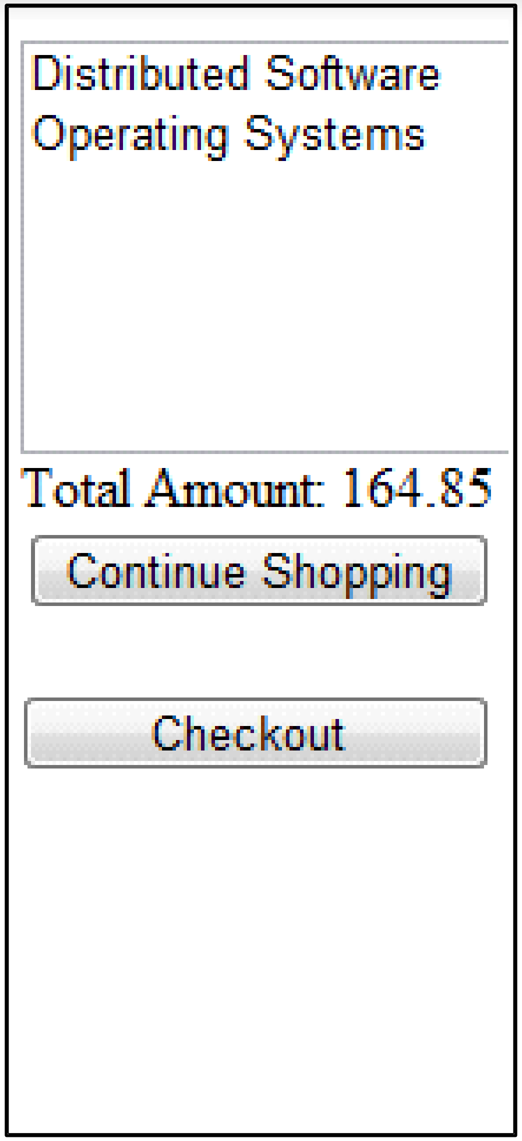


This application allows sellers to add items and allows buyers to select items.

Title: Distributed Software

ISBN: 978-0-7575-5273-1

Price: 79.85



Total Amount: 164.85

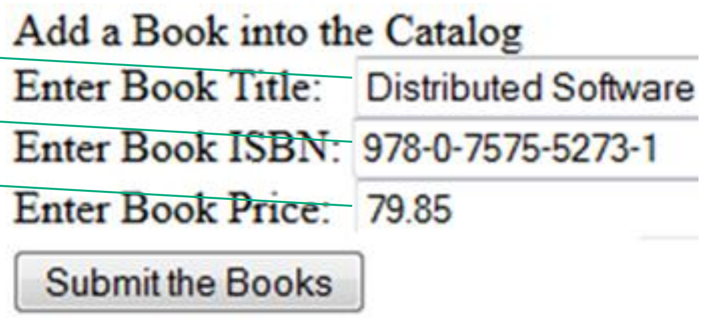
Default.aspx.cs -- Book Class Definition

```
public class Book
{   public string _Title;
    public string _Isbn;
    public double _Price;
    public bool _InCart;   // whether the book is in cart
    public Book(string title, string isbn, double price)
    {     _Title = title;
          _Isbn = isbn;
          _Price = price;
          _InCart = false;
    }
}
```

The constructor loads the parameter values into the class variables

Seller.aspx.cs

```
public partial class Seller : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) {  }
    protected void btnSubmitBook_Click(object sender, EventArgs e) {
        string title = txtTitle.Text;
        string isbn = txtIsbn.Text;
        string sPrice = txtPrice.Text;
        double price = Convert.ToDouble(sPrice);
        Book aBook1 = new Book(title, isbn, price);
        string num = Convert.ToString(Session.Count + 1); // Find the next free spot
        string catalogKey = "sBook" + num;  // Form the index key for next session spot
        Session[catalogKey] = aBook1; // Add an object into session state
        Response.Redirect("Default.aspx");  // Return to catalog page
    }
    protected void txtIsbn_TextChanged(object sender, EventArgs e){
                // text change handler
    }
}
```


Add a Book into the Catalog

Enter Book Title:	Distributed Software
Enter Book ISBN:	978-0-7575-5273-1
Enter Book Price:	79.85

Allow you to write an event handler to response to the change of the text. For example, auto update when a number changes

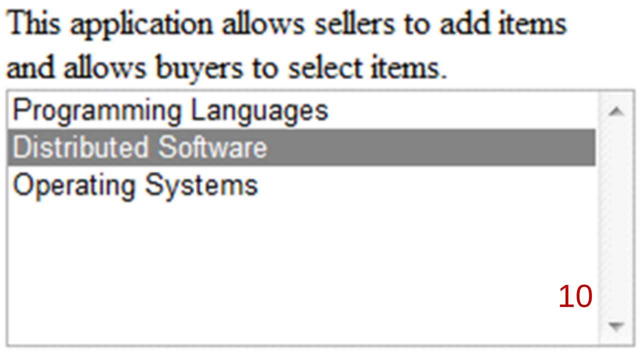
Default.aspx.cs

```
public partial class _Default : System.Web.UI.Page
{
    Book aBook1, aBook2, aBook3;
    string indexKey;
    protected void Page_Load(object sender, EventArgs e)  {
      if ((Session.Count != 0) && (ListBox1.Items.Count == 0))  {
          aBook1 = (Book)Session["sBook1"];
          ListBox1.Items.Add(aBook1._Title);
          aBook2 = (Book)Session["sBook2"];
          ListBox1.Items.Add(aBook2._Title);
          aBook3 = (Book)Session["sBook3"];
          ListBox1.Items.Add(aBook3._Title);
      }
   }
   // Continued next page
```

Code to be executed every time the page is loaded/reloaded.

There is information available in session state

The ListBox is empty


This application allows sellers to add items and allows buyers to select items.

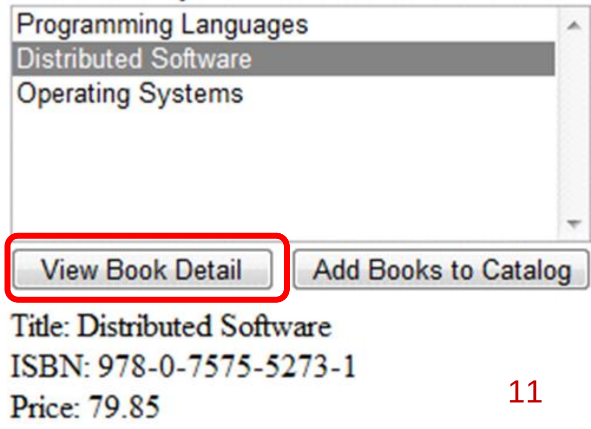
Programming Languages
Distributed Software
Operating Systems

Default.aspx.cs (Contd.)

```
protected void btnSeller_Click(object sender, EventArgs e)
{
   Response.Redirect("Seller.aspx");
}
protected void btnViewBook_Click(object sender, EventArgs e) {
      if (ListBoxCatalog.SelectedIndex < 0 )
         lblTitle.Text = "Please select a book in the list above";
      else {
         string num = Convert.ToString(ListBoxCatalog.SelectedIndex + 1);
         indexKey = "sBook" + num;  // Find the selected book
         Book aBook = (Book)Session[indexKey];
         lblTitle.Text = "<br />Title: " + aBook._Title;
         lblIsbn.Text = "<br />ISBN: " + aBook._Isbn;
         lblPrice.Text = "<br />Price: " + aBook._Price;
      }
   }
```

Jump from Default page to Seller page

No item selected



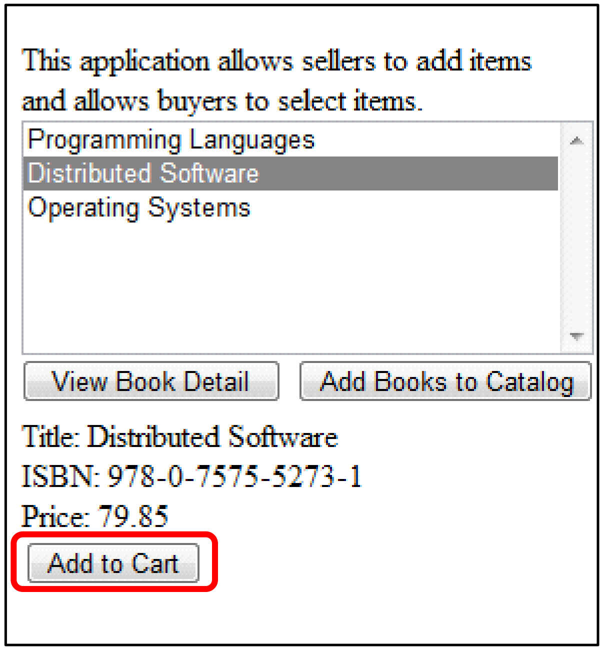
Title: Distributed Software
ISBN: 978-0-7575-5273-1
Price: 79.85

Default.aspx.cs: Add to Cart Button

```
protected void btnAddToCart_Click(object sender, EventArgs e)
{
        string num = Convert.ToString(ListBoxCatalog.SelectedIndex + 1);
        indexKey = "sBook" + num;  // Find selected book
        Book sBook = (Book)Session[indexKey]; // read from state variable
        sBook._InCart = true;                      // add information
        Session[indexKey] = sBook;                 // Write back
        Response.Redirect("MyCart.aspx");
```

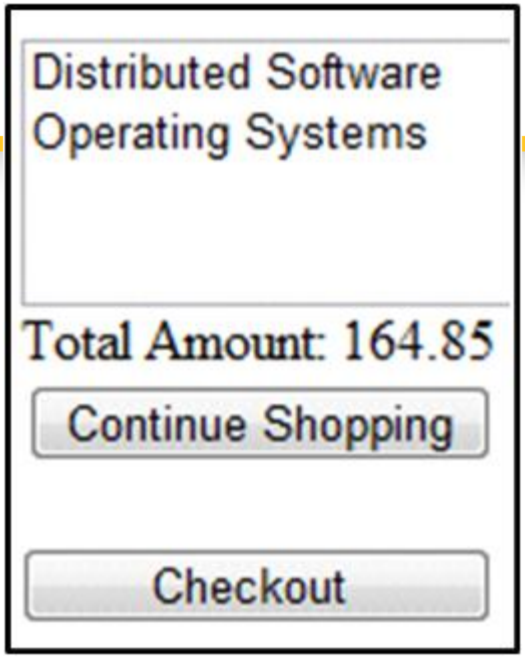
Use a boolean variable here.
You could create a new session state array for the cart.

Jump from current (Default) page to MyCart page


This application allows sellers to add items and allows buyers to select items.

Title: Distributed Software
ISBN: 978-0-7575-5273-1
Price: 79.85

MyCart.aspx.cs

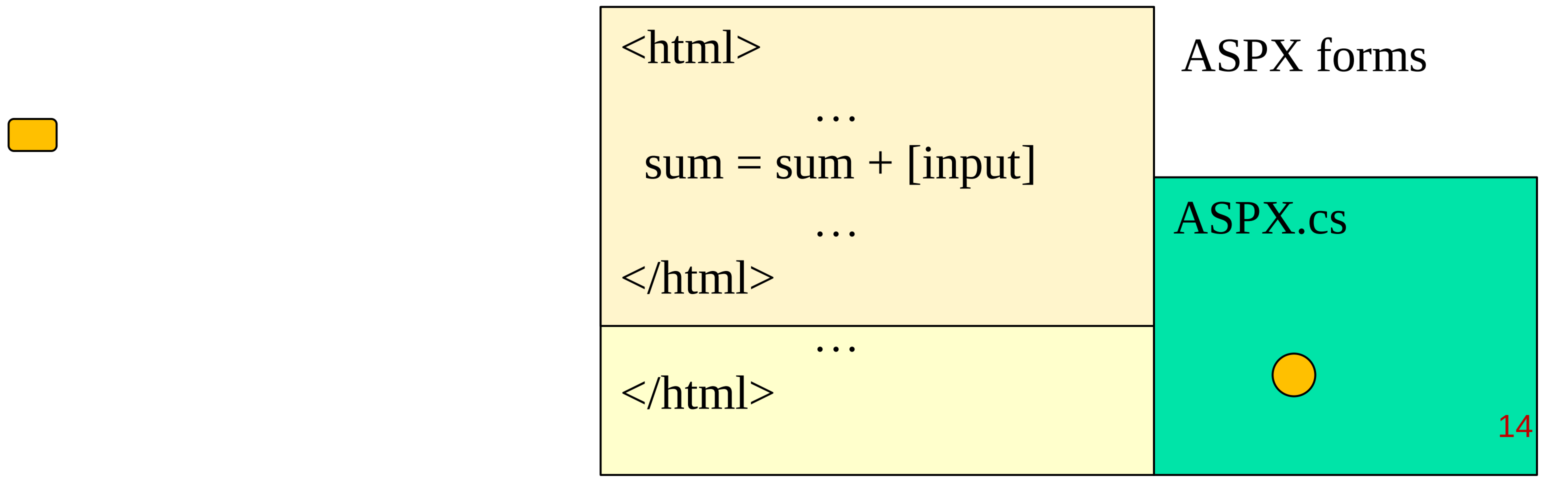

Distributed Software
Operating Systems
Total Amount: 164.85

```
public partial class MyCart : System.Web.UI.Page {
   protected void Page_Load(object sender, EventArgs e) {
      Double totalAmount = 0;
      for (Int16 i = 1; i <= Session.Count; i++) {
         string indexKey = "sBook" + i;
         Book aBook = (Book)Session[indexKey];
         if (aBook._InCart) {
            ListBoxCart.Items.Add(aBook._Title);
            totalAmount = totalAmount + Convert.ToDouble(aBook._Price);
       } }
     lblTotalAmt.Text = "Total Amount: "+Convert.ToString(totalAmount);
   }
   protected void btnToCatalog_Click(object sender, EventArgs e) {
      Response.Redirect("Default.aspx"); // continue shopping
   }
 protected void btnToCheckout_Click(object sender, EventArgs e) {
      Response.Redirect(“Checkout.aspx");
} }
```

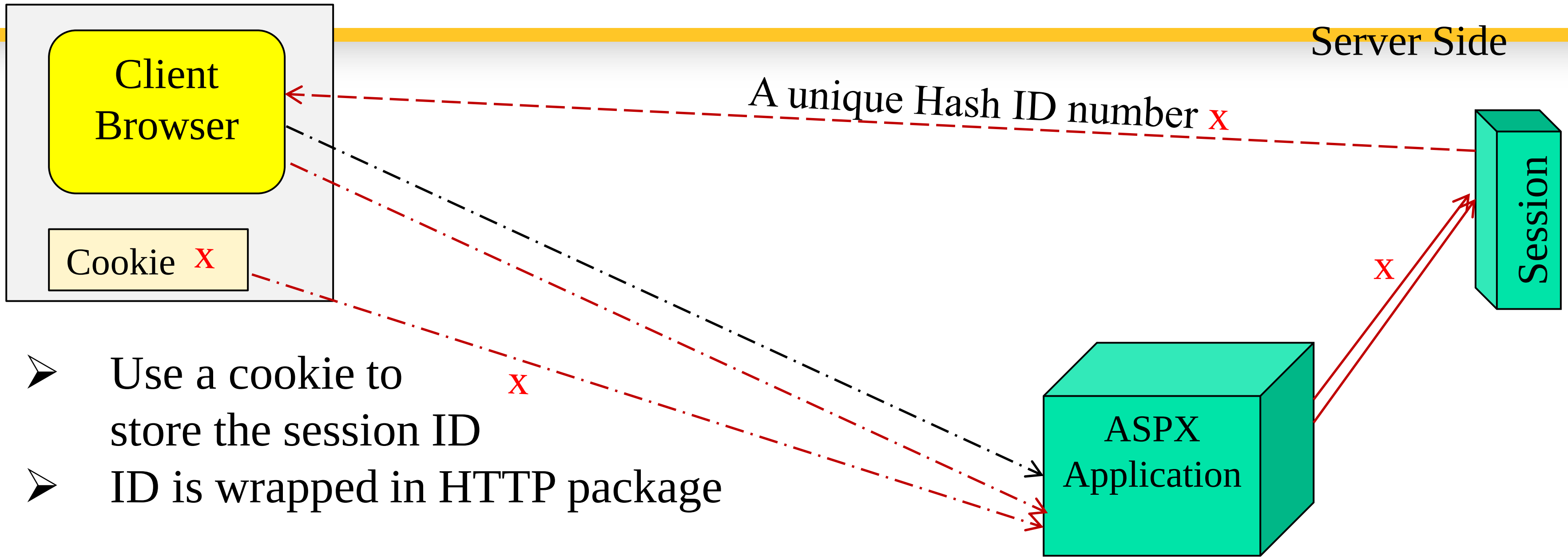
Checkout page not shown in this example

Cookie Support to Session State

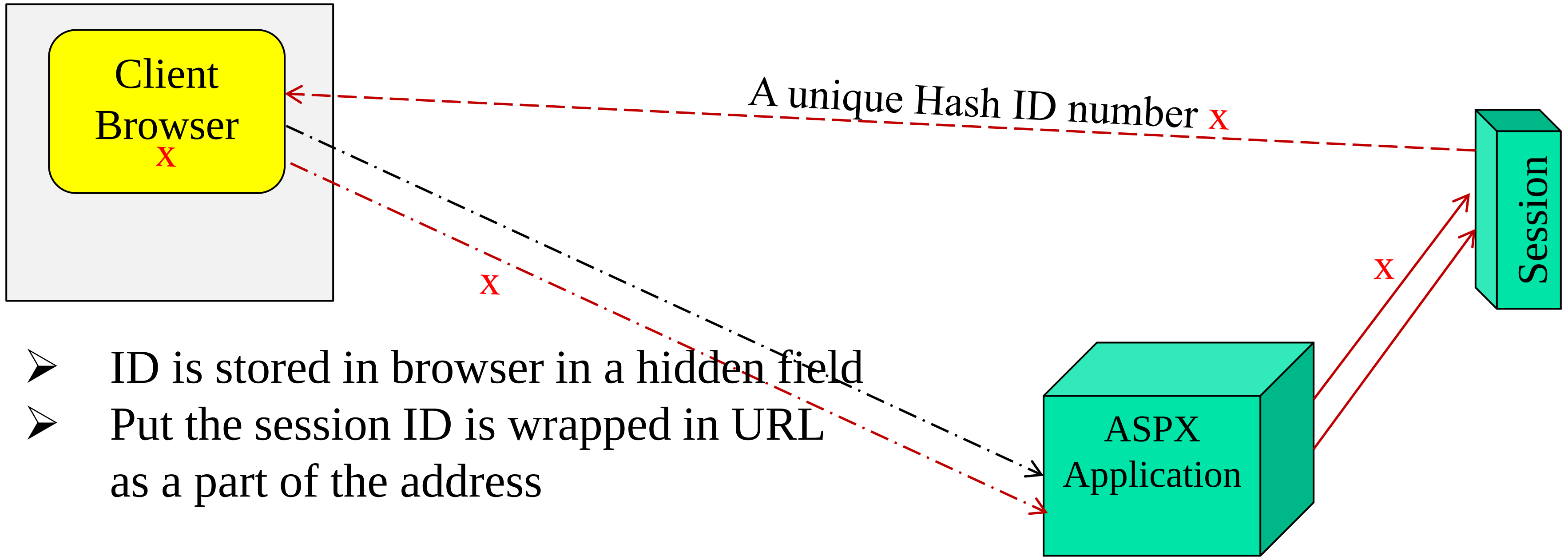
- HTTP is stateless. Each visit is considered to be from a new user;
- The browser needs to explicitly carry the session ID (of 120 bits) when it revisits a session and its session variable;
  - Use a cookie to store the session id
  - Put the session id in URL as a part of the address



Storing and Sending Session ID

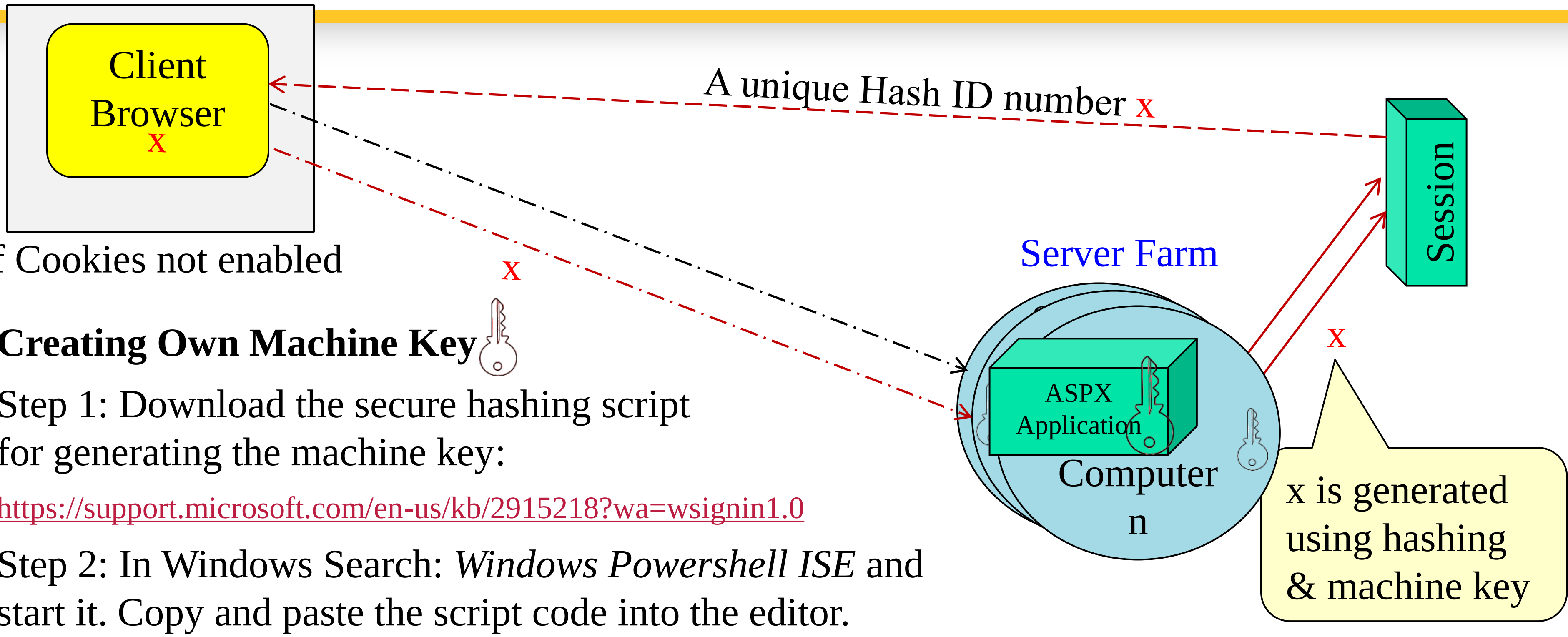


- Use a cookie to store the session ID
- ID is wrapped in HTTP package



- ID is stored in browser in a hidden field
- Put the session ID is wrapped in URL as a part of the address

View State Security and Creating Machine Key



Client Browser
x
A unique Hash ID number x
Session
Server Farm
x
ASPX Application
Computer n
x
x is generated using hashing & machine key

If Cookies not enabled

Creating Own Machine Key

Step 1: Download the secure hashing script for generating the machine key:

https://support.microsoft.com/en-us/kb/2915218?wa=wsignin1.0

Step 2: In Windows Search: *Windows Powershell ISE* and start it. Copy and paste the script code into the editor.

Step 3: Run the script by clicking the **green triangle button** ▶ at the upper part of the window. Then, type the following command at the prompt:

```
PS C:\scripts> Generate-Machinekey <enter>
```

Step 4: Add it into the project's Web.config file in the element:

```
<system.web>
    <machineKey decryption="AES" decryptionKey="xxxxxxxxxxxxxxxxxxxxxxx"
    validation="HMACSHA256" validationKey="yyyyyyyyyyyyyyyyyyyyyyyy" />
</system.web>
```

was required for using ASU-hosted WebStrar Server

When You Shop and Add Items in Cart

- If you do not enable Cookies, you can go back and forth between different pages, you can still see your items in the shopping cart;
- If you close your session (browser), your items in the cart will disappear.
- If you enabled cookies, your items in cart will stay even if you have closed the browser;
- If you use a different browser, the items will not be visible.
- *If you save the cart items into a disk file associated with your account, the items will be visible in different browsers.*

HttpSessionState Class: Public Properties

- Contents Gets a reference to the current session-state object.
- Count Gets the number of items in the session-state collection.
- IsCookieless Gets a value indicating whether the session ID is embedded in the URL or stored in an HTTP cookie.
- Mode Gets the current session-state mode.
- IsNewSession Gets a value indicating whether the session was created with the current request.
- IsReadOnly Gets a value indicating whether the session is read-only.
- IsSynchronized Gets a value indicating whether access to the collection of session-state values is synchronized (thread safe).
- Keys Gets a collection of the keys of all values stored in the session.
- SessionID Gets the unique session ID used to identify the session.
- StaticObjects Gets a collection of objects declared by <object Runat="Server" Scope="Session"/> tags within the ASP.NET application file global.asax.
- SyncRoot Gets an object that can be used to synchronize access to the collection of session-state values.
- Timeout Gets and sets the time-out period (in minutes) allowed between requests before the session-state provider terminates the session. You can set the minutes in web.config file

Session State Setting Using Web.config

1. UseCookies
2. UseUri
3. UseDeviceProfile
4. AutoDetect

```
<system.web>
     <sessionState
          cookieless = “HttpCookieMode values”
          timeout = “int, number of minutes”
          … >
      </sessionState>
</system.web>
```

HttpCookieMode Values

1. UseCookies
2. UseUri
3. UseDeviceProfile
4. AutoDetect

- **UseCookies**: Always assume that cookies are supported by browser and are enabled.
  - Session will not work if cookies are not enabled
- **UseUri**: There are potential problems:
  - If an absolute path is used in the program, storing session id in browser and use URL will cause an page error.
  - Session variables discarded after the session is terminated.
- **UseDeviceProfile**: It checks if the browser supports cookies. If it does, set mode to UseCookies; Otherwise, set mode to UseUri.
- **AutoDetect**: It checks if the browser supports cookies and tests if the cookie is enabled, by creating a cookie, saving it, and retrieving it. It is slow as it needs to go back and forth several times between the server and client. Otherwise, set mode to UseUri (mostly in mobile devices).

HttpSessionState Class: Public Methods

- Abandon Cancels the current session.
- Add Adds a new item to session state, or use Session[“key”] = x;
- Clear Clears all values from session state.
- CopyTo Copies the collection of session-state values to a one-dimensional array, starting at the specified index in the array.
- Equals (inherited from Object) Overloaded. Determines whether two Object instances are equal.
- GetEnumerator Gets an enumerator of all session state-values in the current session.
- GetType (inherited from Object) Gets the Type of the current instance.
- Remove Deletes an item from the session-state collection.
- RemoveAll Clears all session-state values.
- RemoveAt Deletes an item at a specified index from the session-state collection.
- ToString (inherited from Object) Returns a String that represents the current Object.

Using Add and Remove Methods

```
string itemName = Server.HtmlEncode(TextBox1.Text);
string itemValue = Server.HtmlEncode(TextBox2.Text);
Session.Add(itemName, itemValue);
// Same as Session[itemName] = itemValue;


RedundantItem  itemToRemove = e.Item;
string sessionItemToRemove =
    ((Label)itemToRemove.FindControl("Label1")).Text;
Session.Remove(sessionItemToRemove);
```

From Session State to Application State

- Session[“index”] allows you to store an object into server memory, and all pages in the session can access the session variable. But data will disappear after closing the browser
- Application[“index”] allows you to store an object into server memory. All sessions and all pages in each session can access the application variable;
    - You can define, for example, Application[“SuperCounter”], similar to the Global.asax file and access the variable in each session;
    - The challenges remain here: write-write conflict and lock performance, if the application is frequently accessed.

ASU® Ira A. Fulton Schools of Engineering

Arizona State University

M12 L5
Web State Management: File System

Ira A. Fulton Schools of Engineering
Arizona State University

Lecture Outline

- File system operations: read and write
- Save web data into XML file on server
- Accessing your files
- From XML files to XML database

Save Data Permanently into Server Disk

All techniques discussed so far are not permanent. Data can disappear for different reasons.

- Session state data disappears after the session is closed.
- Application state data (e.g., the global counter) will also disappear if the application is closed. When?

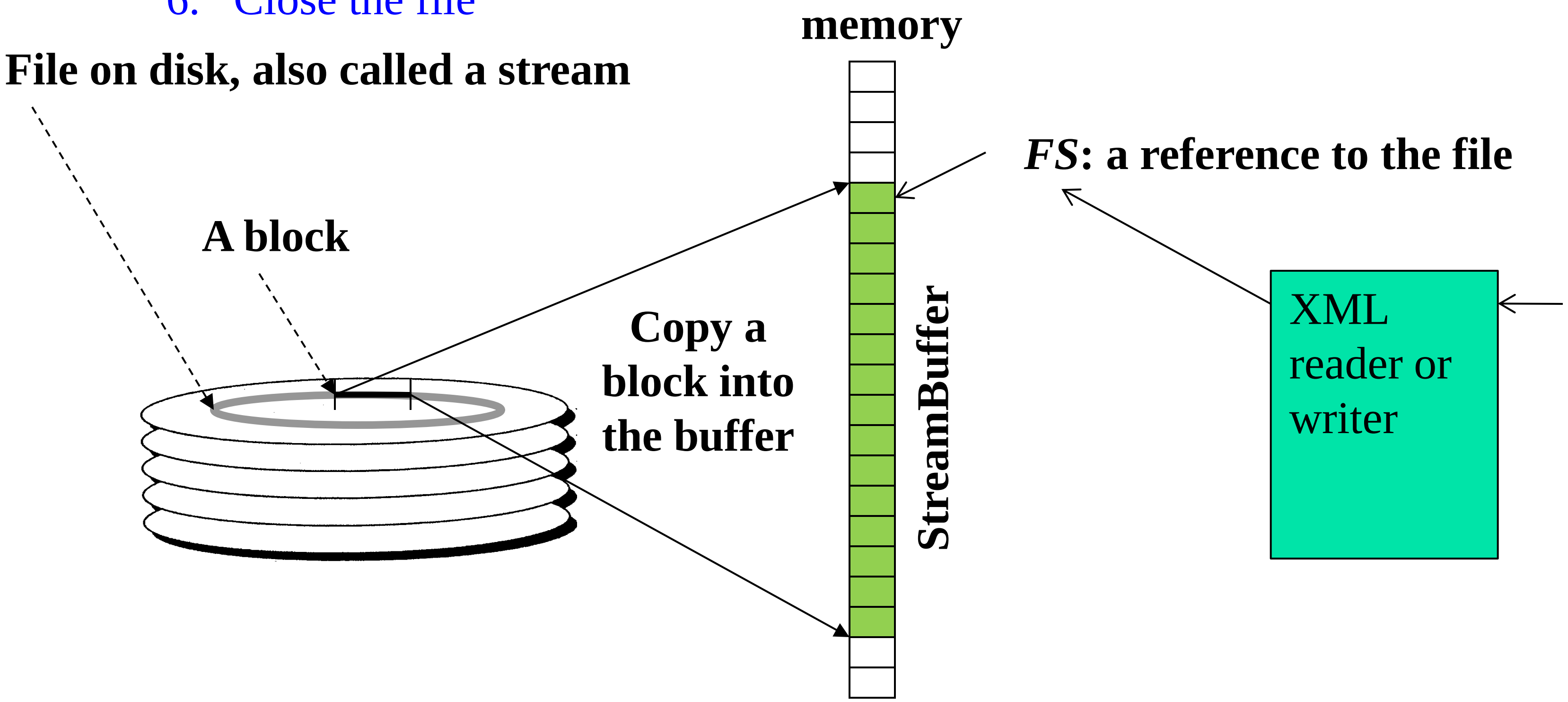
▪ Save into a text file;	▪ Stream read a string;
▪ Save into a binary file;	▪ Read structured variable;
▪ Save into an XML file using the XML Writer;	▪ XMLDocument class;
▪ Save into database;	▪ Read from database;

Text chapter 10

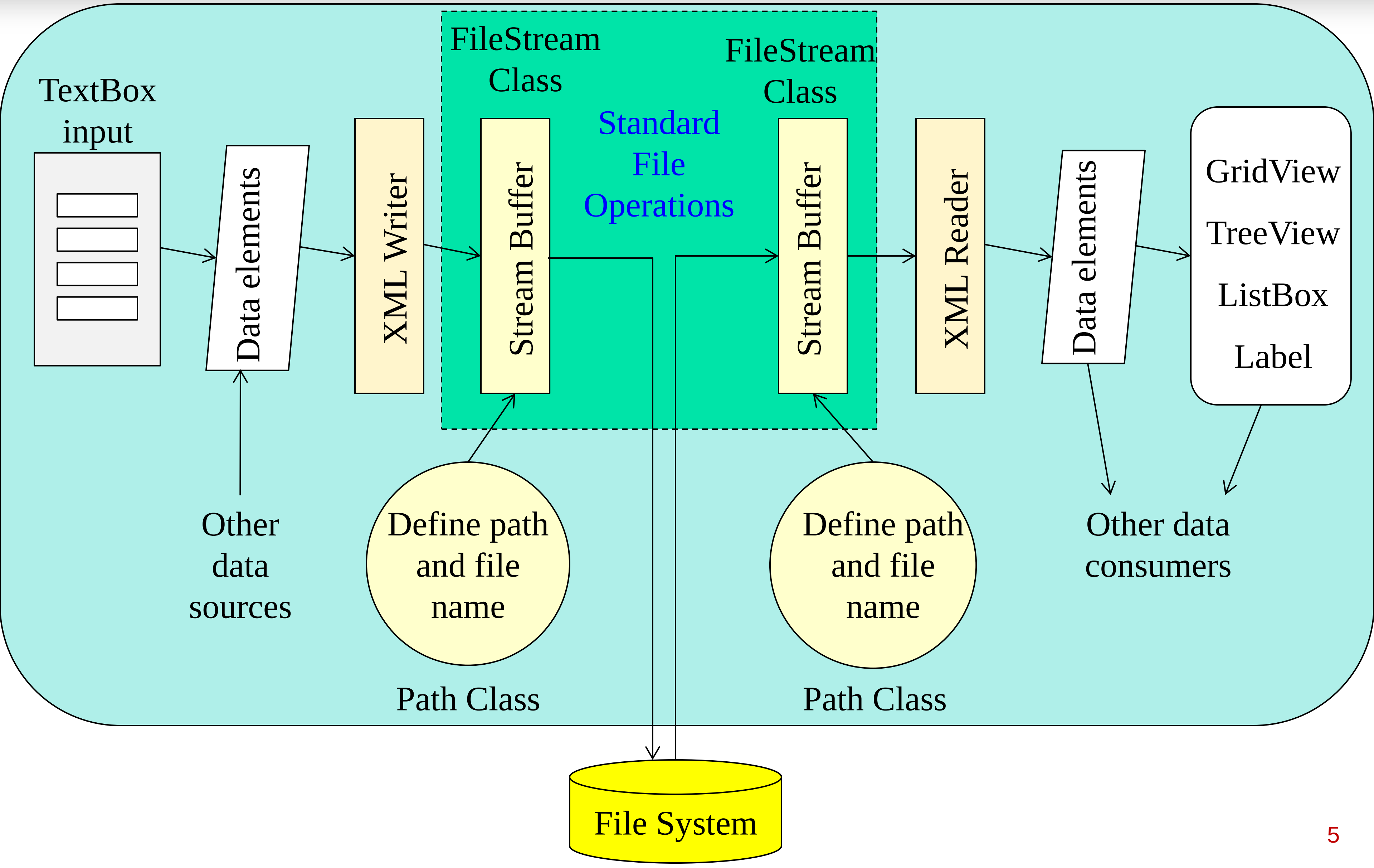
File System Read (and Write) XML Files

OS operations, not a part of ASP .Net

1. Declare a reference FS of a FileStream type;
2. Open a file for read or write: It creates a buffer that can hold a block of bytes;
3. Copy the first block of a file into the buffer;
4. Create an XML reader or writer that uses the reference to read/write data in the buffer;
5. When the reference moves down to the end of the buffer, the next block is fetched.
6. Close the file


memory
File on disk, also called a stream
FS: a reference to the file
A block
Copy a block into the buffer
StreamBuffer
XML reader or writer

A Scenario of XML Data Exchange between ASP application and File System

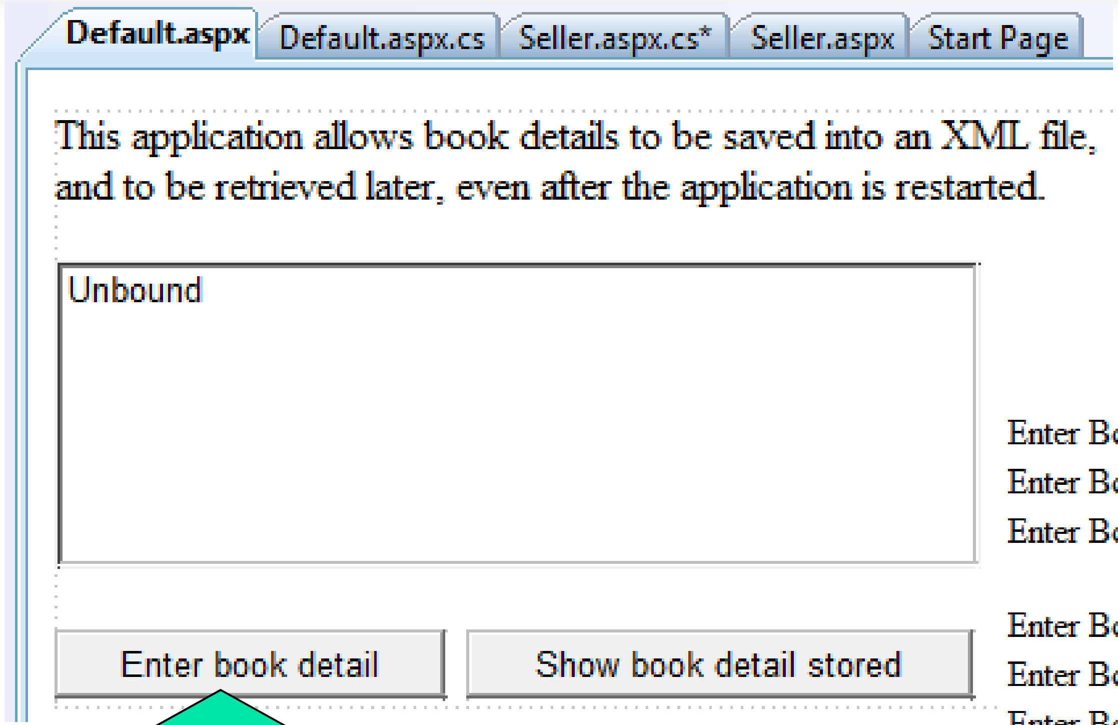


.Net System.IO Namespace and its Classes

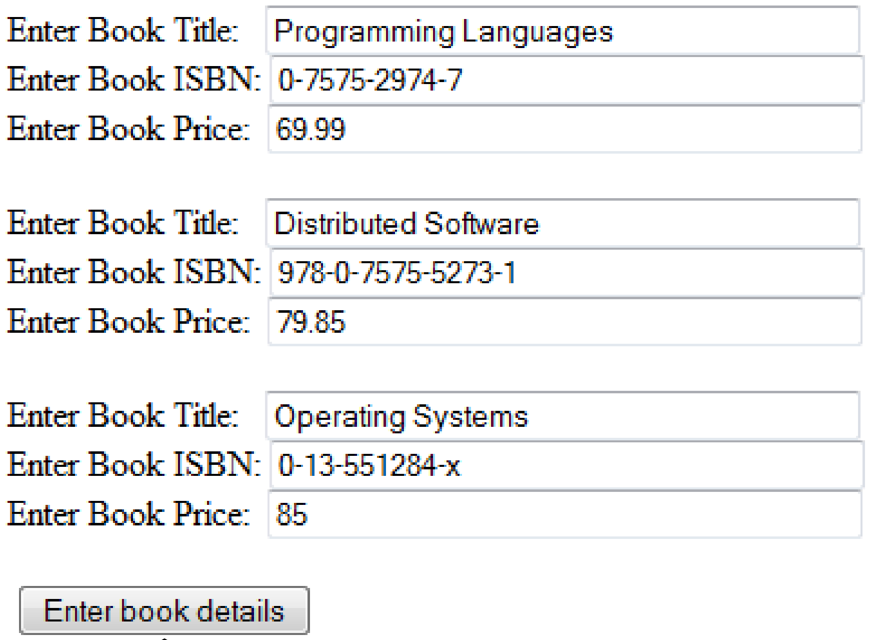
- Namespace System.IO has a number of classes.
- Path class specifies the path and file name to be accessed;
- FileStream class creates a buffer and connection to the file system;

```
string p= @“c:cse445\fileAccess\App_Data\Book.xml”
string fName = Path.GetFileName(p);  ➡  Book.xml

string p1 = @“c:cse445\”
string p2 = @“fileAccess\App_Data\Book.xml”
string location= Path.Combine(p1, p2);
```

Save Data into an XML-File on Server

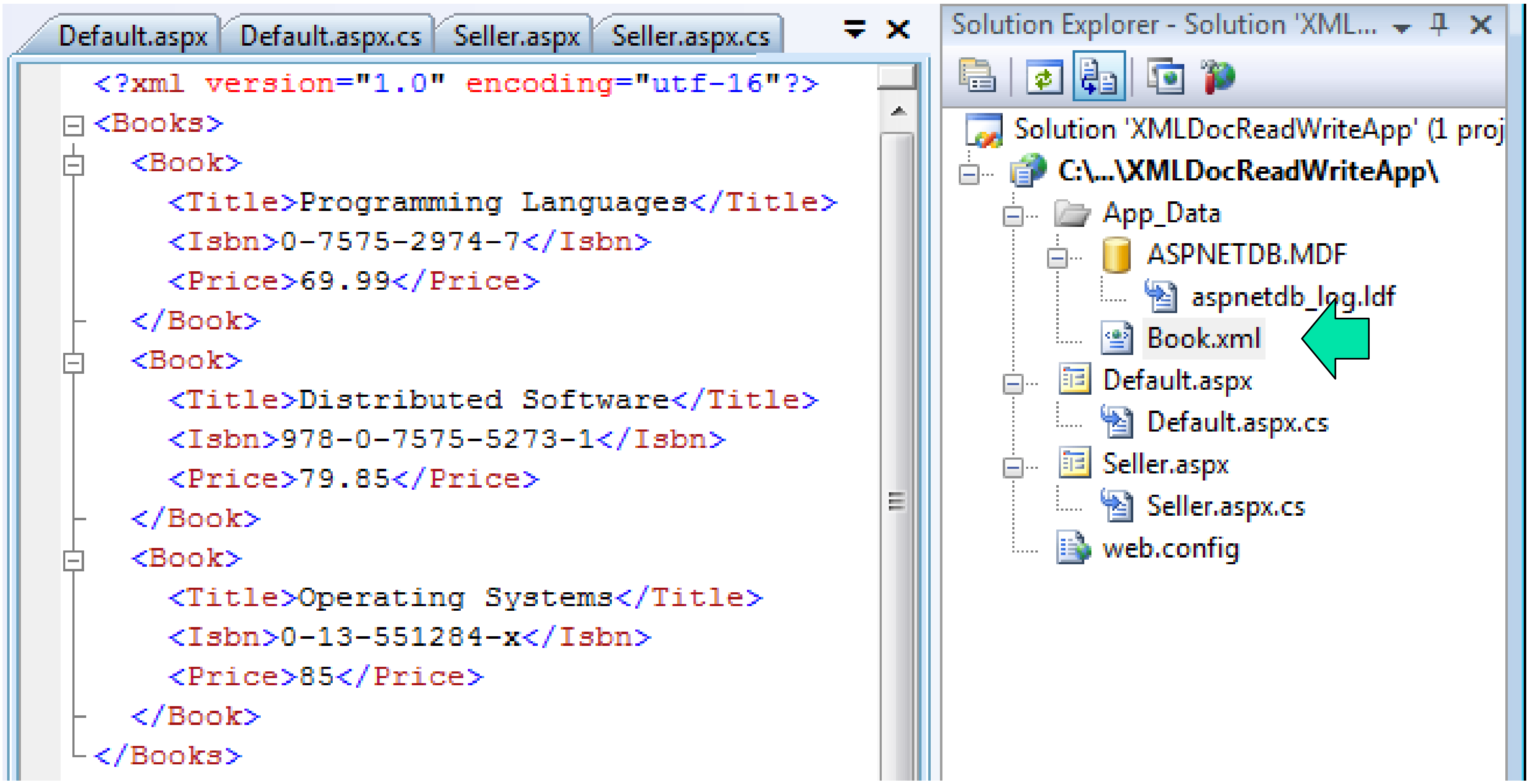


This application allows book details to be saved into an XML file, and to be retrieved later, even after the application is restarted.



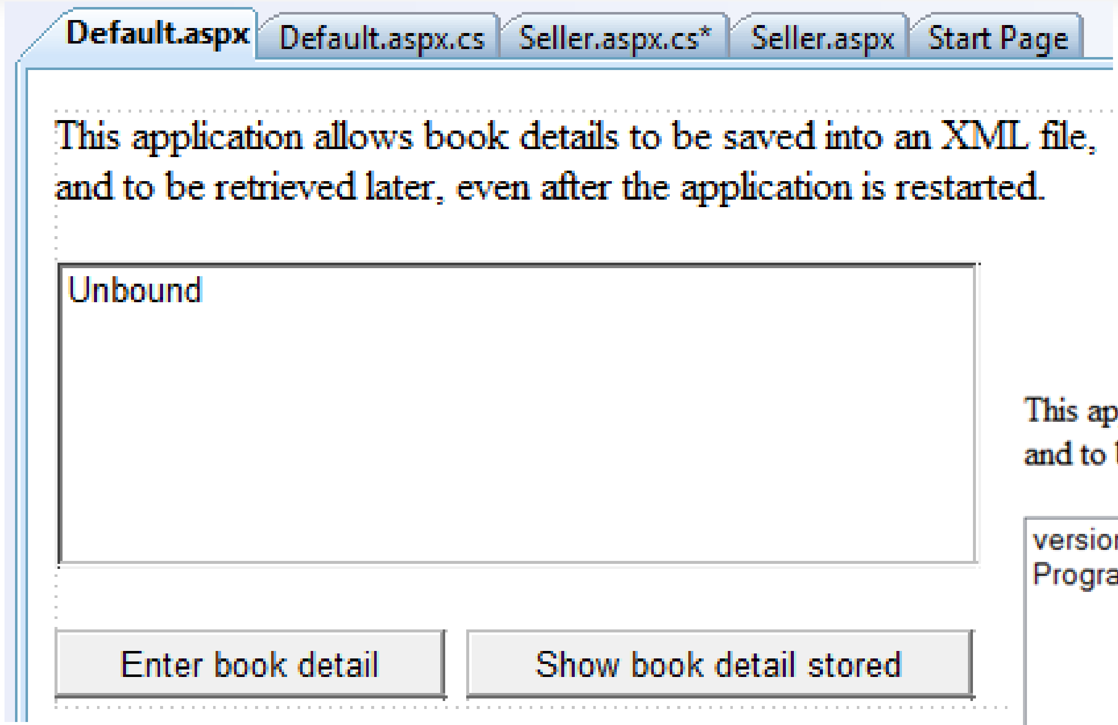
Enter Book Title:	Programming Languages
Enter Book ISBN:	0-7575-2974-7
Enter Book Price:	69.99
Enter Book Title:	Distributed Software
Enter Book ISBN:	978-0-7575-5273-1
Enter Book Price:	79.85
Enter Book Title:	Operating Systems
Enter Book ISBN:	0-13-551284-x
Enter Book Price:	85

Book.xml Generated through XMLWriter



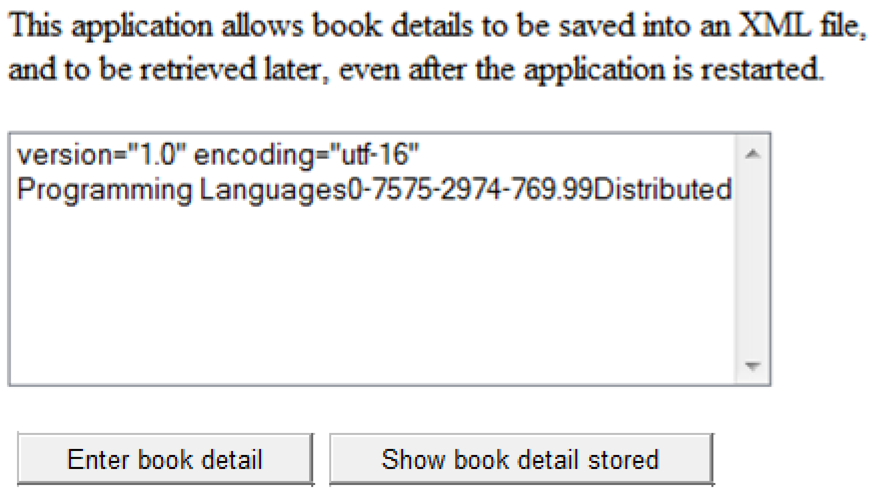
```xml
<?xml version="1.0" encoding="utf-16"?>
<Books>
  <Book>
    <Title>Programming Languages</Title>
    <Isbn>0-7575-2974-7</Isbn>
    <Price>69.99</Price>
  </Book>
  <Book>
    <Title>Distributed Software</Title>
    <Isbn>978-0-7575-5273-1</Isbn>
    <Price>79.85</Price>
  </Book>
  <Book>
    <Title>Operating Systems</Title>
    <Isbn>0-13-551284-x</Isbn>
    <Price>85</Price>
  </Book>
</Books>
```

Save Data into XML-File on Server



This application allows book details to be saved into an XML file, and to be retrieved later, even after the application is restarted.

Unbound

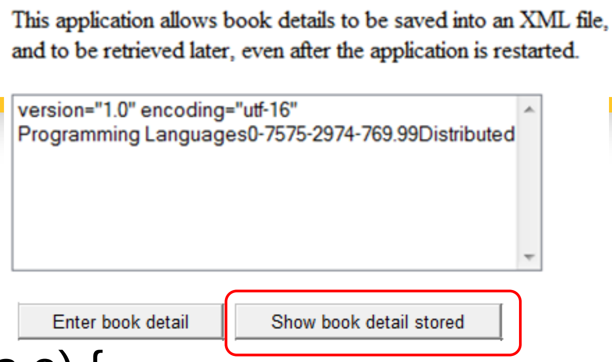


This application allows book details to be saved into an XML file, and to be retrieved later, even after the application is restarted.

```
version="1.0" encoding="utf-16"
Programming Languages0-7575-2974-769.99Distributed
```

You can see a better formatted display at this page:
http://venus.sod.asu.edu/WSRepository/XMLDocReadWriteApp/Default.aspx
http://webstrar1.fulton.asu.edu/page1/Default.aspx

Code Behind the Default Page



This application allows book details to be saved into an XML file, and to be retrieved later, even after the application is restarted.

```
version="1.0" encoding="utf-16"
Programming Languages0-7575-2974-769.99Distributed
```

```
public partial class _Default : System.Web.UI.Page  {
     protected void btnSeller_Click(object sender, EventArgs e) {
             Response.Redirect("seller.aspx");
     }
     protected void btnShowBook_Click(object sender, EventArgs e) {
     FileStream fS = null;
     string fLocation = Path.Combine(Request.PhysicalApplicationPath,
     @"App_Data\Book.xml"); // or: HttpRuntime.AppDomainAppPath
      try {
        if (File.Exists(fLocation)) {
           FileStream fS= new FileStream(fLocation, FileMode.Open, FileAccess.Read);
           XmlDocument xd = new XmlDocument();
           xd.Load(fS);
          fS.Close();
```

Find path to the current location

Check if the file exists

Open for read only

Load the XML file into memory

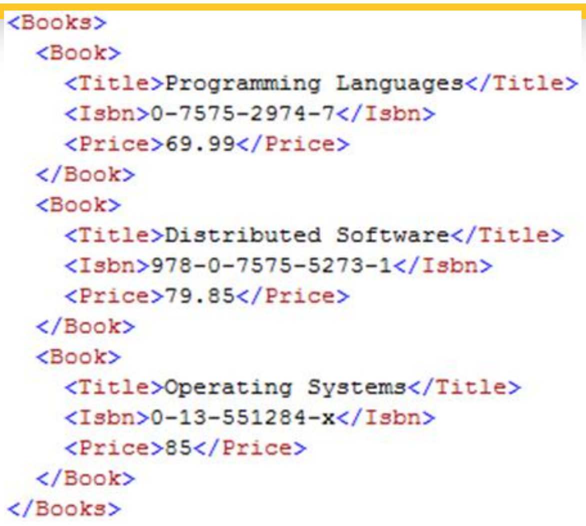
Close the file immediately after loading the entire tree.

Code Behind the Default (Reader) Page

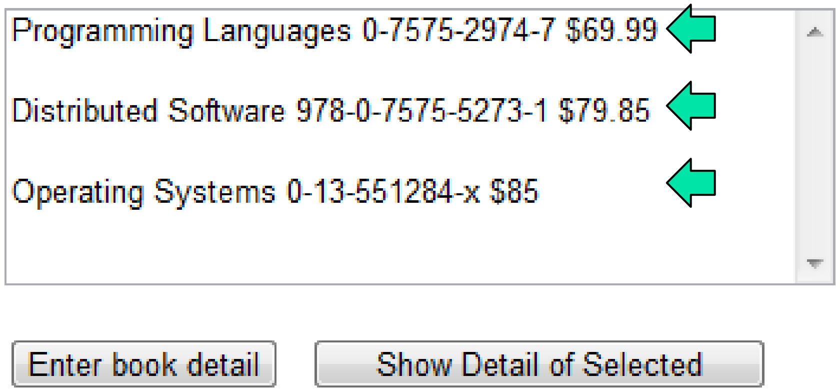
```
        XmlNode node = xd;
        XmlNodeList children = node.ChildNodes;
        foreach (XmlNode child in children)
        {
            ListBox1.Items.Add(child.InnerText);
        }
      }
     finally  {
        fS.Close();
      }
   }
}
```

In case the session crashes

This part of the code needs to be refined, using what you have done in Project 4 XML processing.



```
<Books>
  <Book>
    <Title>Programming Languages</Title>
    <Isbn>0-7575-2974-7</Isbn>
    <Price>69.99</Price>
  </Book>
  <Book>
    <Title>Distributed Software</Title>
    <Isbn>978-0-7575-5273-1</Isbn>
    <Price>79.85</Price>
  </Book>
  <Book>
    <Title>Operating Systems</Title>
    <Isbn>0-13-551284-x</Isbn>
    <Price>85</Price>
  </Book>
</Books>
```

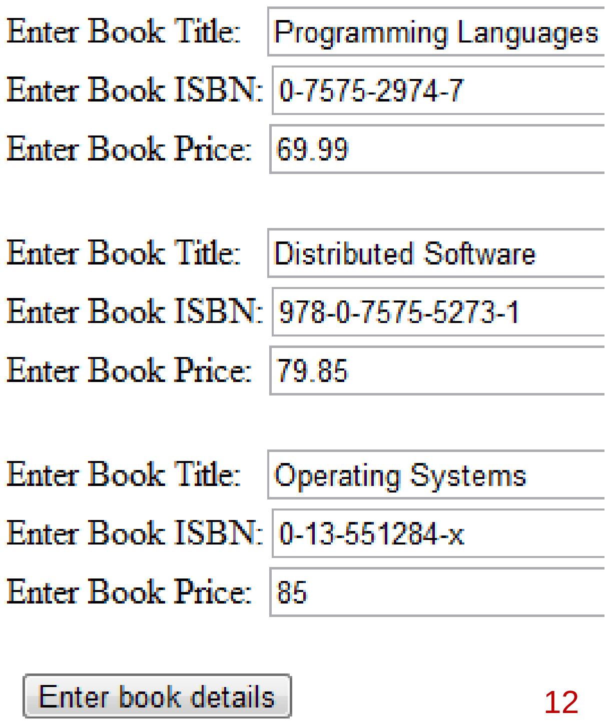


```
Programming Languages 0-7575-2974-7 $69.99
Distributed Software 978-0-7575-5273-1 $79.85
Operating Systems 0-13-551284-x $85
```

Code Behind the Data Enter (Writer) Page

```
public partial class Seller : System.Web.UI.Page  {
     protected void Page_Load(object sender, EventArgs e) {
     }
   protected void btnEnterBook_Click(object sender, EventArgs e)  {
            string title1 = txtTitle1.Text;
            string isbn1 = txtIsbn1.Text;
            string sPrice1 = txtPrice1.Text;

            string title2 = txtTitle2.Text;
             string isbn2 = txtIsbn2.Text;
             string sPrice2 = txtPrice2.Text;

            string title3 = txtTitle3.Text;
            string isbn3 = txtIsbn3.Text;
            string sPrice3 = txtPrice3.Text;
```

Taking data from text boxes



Enter Book Title:	Programming Languages
Enter Book ISBN:	0-7575-2974-7
Enter Book Price:	69.99
Enter Book Title:	Distributed Software
Enter Book ISBN:	978-0-7575-5273-1
Enter Book Price:	79.85
Enter Book Title:	Operating Systems
Enter Book ISBN:	0-13-551284-x
Enter Book Price:	85

Code Behind the Data Enter (Writer) Page

```
        string fLocation = Path.Combine(Request.PhysicalApplicationPath,
            @"App_Data\Book.xml");  // or: HttpRuntime.AppDomainAppPath
FileStream fS = null;
try {
        fS = new FileStream(fLocation, FileMode.Truncate);
        XmlTextWriter writer = new XmlTextWriter(fS,
            System.Text.Encoding.Unicode);
        writer.Formatting = Formatting.Indented;
        writer.WriteStartDocument();
        writer.WriteStartElement("Books");
        writer.WriteStartElement("Book");
        writer.WriteElementString("Title", title1);
        writer.WriteElementString("Isbn", isbn1);
        writer.WriteElementString("Price", sPrice1);
        writer.WriteEndElement();
```

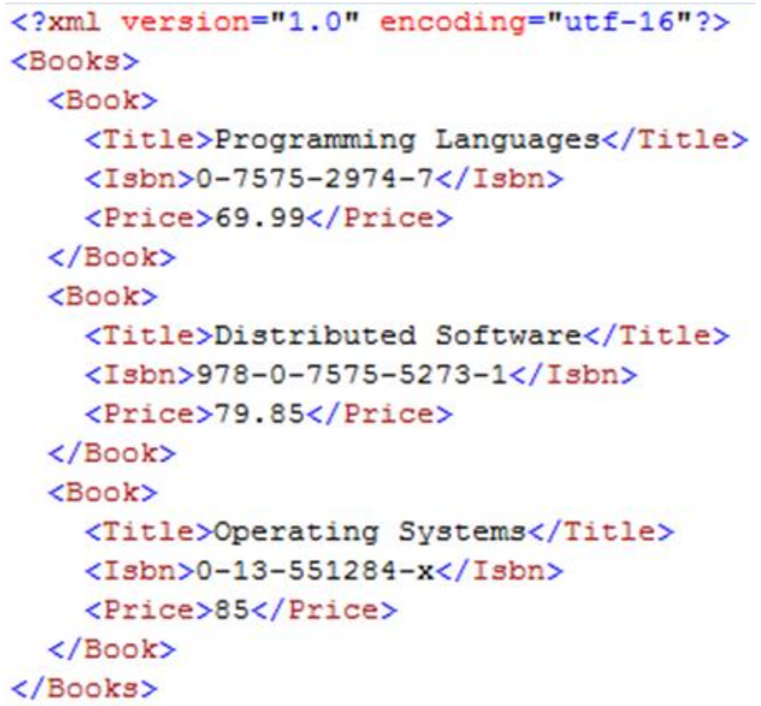
Delete the existing content.
Other modes include OpenOrCreate, Append, …

See chapter 4 slides on XMLWriter

Code Behind the Data Enter (Writer) Page

```
    writer.WriteStartElement("Book");
    writer.WriteElementString("Title", title2);
    writer.WriteElementString("Isbn", isbn2);
    writer.WriteElementString("Price", sPrice2);
    writer.WriteEndElement();
    writer.WriteStartElement("Book");
    writer.WriteElementString("Title", title3);
    writer.WriteElementString("Isbn", isbn3);
    writer.WriteElementString("Price", sPrice3);
    writer.WriteEndElement();
    writer.WriteEndElement();
    writer.WriteEndDocument();
    writer.Close();
    fS.Close();
}
```

XMLWriter continues to wrote



```
<?xml version="1.0" encoding="utf-16"?>
<Books>
  <Book>
    <Title>Programming Languages</Title>
    <Isbn>0-7575-2974-7</Isbn>
    <Price>69.99</Price>
  </Book>
  <Book>
    <Title>Distributed Software</Title>
    <Isbn>978-0-7575-5273-1</Isbn>
    <Price>79.85</Price>
  </Book>
  <Book>
    <Title>Operating Systems</Title>
    <Isbn>0-13-551284-x</Isbn>
    <Price>85</Price>
  </Book>
</Books>
```

It is necessary to close the XMLWriter **and** to close the file stream connection. You cannot open the file if any one is open.

Code Behind the Data Enter (Writer) Page

```
        finally  {
              fS.Close();
        }

        Response.Redirect("Default.aspx");
    }
        protected void txtIsbn_TextChanged(object sender, EventArgs e)
        {
                // can write an event handler to do something as user types
        }
    }
```

In case the session crashes

Discussions: Book.xml (Catalog) File

- Book.xml is the catalog file and is accessible by the public;
- It can be read by many shoppers simultaneously

  FileStream fS= new FileStream(fLocation, FileMode.Open, FileAccess.Read);

  Multiple sessions can open and read at the same time.

  Other modes are FileAccess.Write and FileAccess.ReadWrite

- Many sellers can try to write it simultaneously

  FileStream fS= new FileStream(fLocation, FileMode.Open, FileAccess.Write);

  Write operations must be locked from other reads or writes.

- Is a deadlock possible on Book.xml?

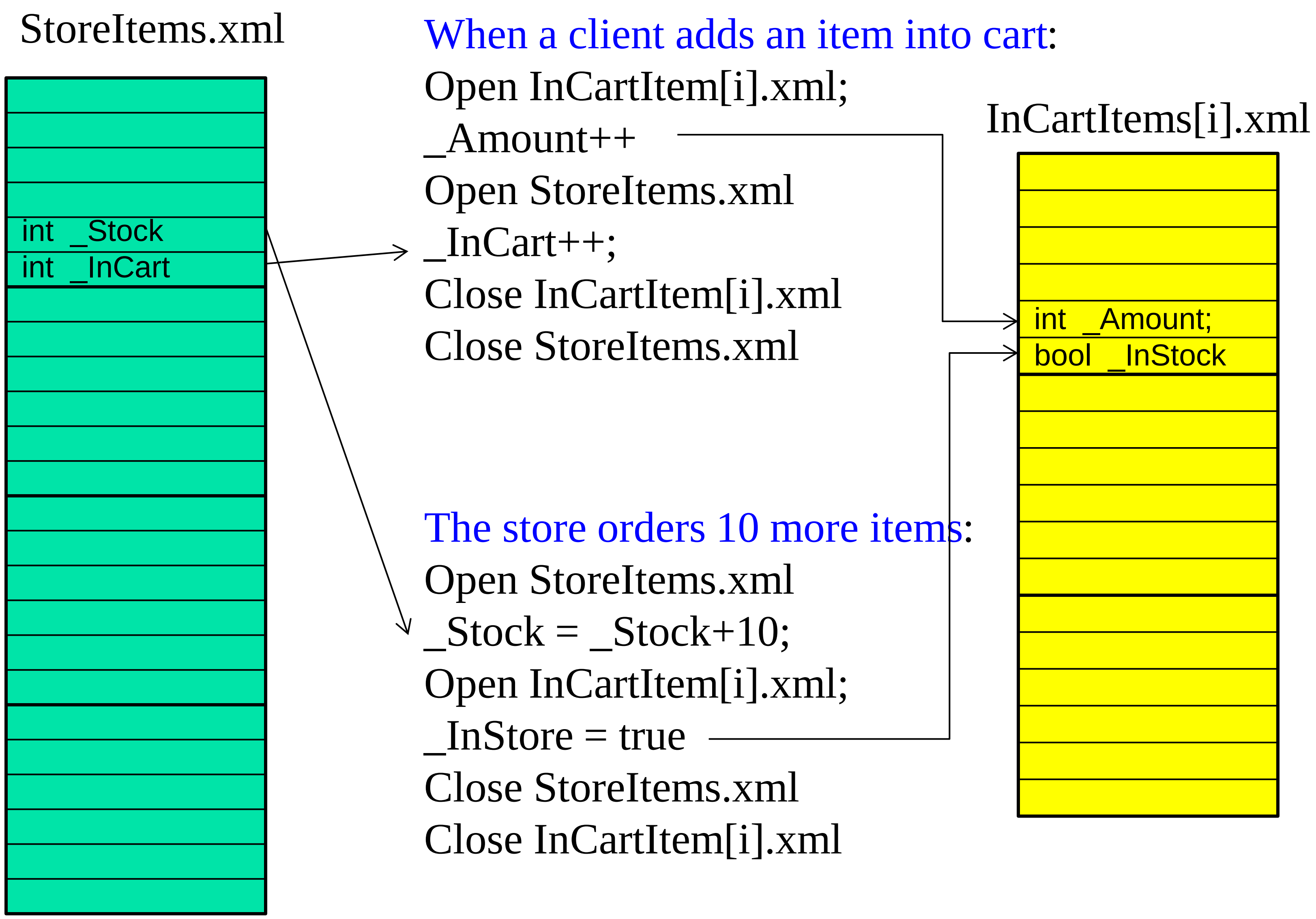
Consider a Business Model with …

```
public class StoreItem {  // for catalog
     public string  _ItemName;
     public string  _ItemNo;
     public double  _UnitPrice;
     public int  _Stock;   // Number of items available in store
     public int  _InCart;  // Number of items in customer carts
}

public class CartItem{
     public string  _ItemName;
     public string  _ItemNo;
     public double  _UnitPrice;
     public int  _Amount;   // Number of items in shopping cart
     public bool  _InStock
}
```

Use these two numbers to predict how many items should be ordered.

- Order too many: product cost and storage cost
- Order too few: lose business opportunity

Managing Your Data Files



When a client adds an item into cart:

```
Open InCartItem[i].xml;
_Amount++
Open StoreItems.xml
_InCart++;
Close InCartItem[i].xml
Close StoreItems.xml
```

The store orders 10 more items:

```
Open StoreItems.xml
_Stock = _Stock+10;
Open InCartItem[i].xml;
_InStore = true
Close StoreItems.xml
Close InCartItem[i].xml
```

Dead Lock

When a client adds an item in cart:

```
Open InCartItem[i].xml;
_Amount++
Open StoreItems.xml
_InCart++;
Close InCartItem[i].xml
Close StoreItems.xml
```

The store orders 10 items:

```
Open StoreItems.xml
_Stock = _Stock+10;
Open InCartItem[i].xml;
_InStore = true
Close StoreItems.xml
Close InCartItem[i].xml
```

Dead Lock Prevention

When a client adds an item in cart:

```
Open InCartItem[i].xml;
_Amount++
Close InCartItem[i].xml
Open StoreItems.xml
_InCart++;
Close StoreItems.xml
```

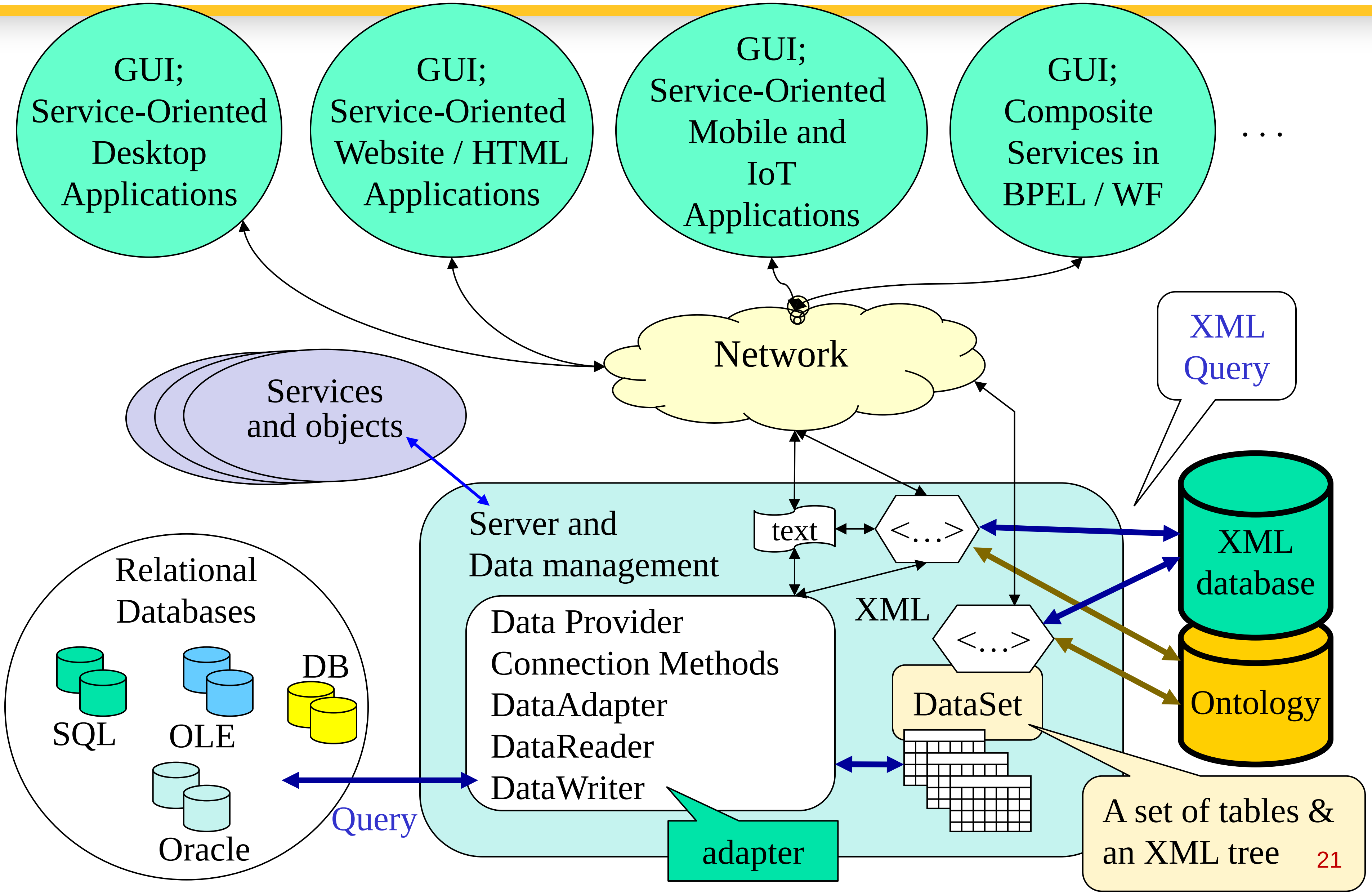
The store orders 10 items:

```
Open StoreItems.xml
_Stock = _Stock+10;
Close StoreItems.xml
Open InCartItem[i].xml;
_InStore = true
Close InCartItem[i].xml
```

XML File vs. XML Database

- XML file access models
  - DOM model: read the entire XML file into memory
  - SAX model: Read one node at a time into memory
- If XML file is very big, none of the models work:
  - It takes too much memory;
  - It takes too much time to sequentially traversing a large file.
- XML database is the solution (Text Chapter 10):
  - It creates index for fast search
  - It runs the search code in database machine, instead of the client machine.
  - It similar to a relational database in query processing

Databases, Web Services, and Web Applications


GUI; Service-Oriented Desktop Applications
GUI; Service-Oriented Website / HTML Applications
GUI; Service-Oriented Mobile and IoT Applications
GUI; Composite Services in BPEL / WF
. . .
Network
XML Query
Services and objects
Server and Data management
text
<…>
XML
<…>
XML database
Ontology
Data Provider
Connection Methods
DataAdapter
DataReader
DataWriter
DataSet
A set of tables & an XML tree
adapter
Relational Databases
SQL
OLE
DB
Oracle
Query

ASU®
Ira A. Fulton Schools of
Engineering
Arizona State University