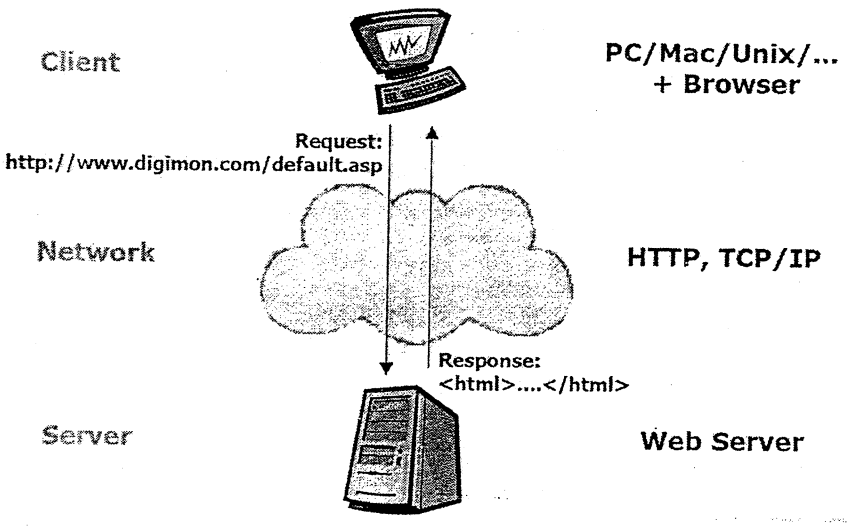
Prasanna wadekar 9970248407

63=00

ASP.NET Web Forms

Background Web Architecture



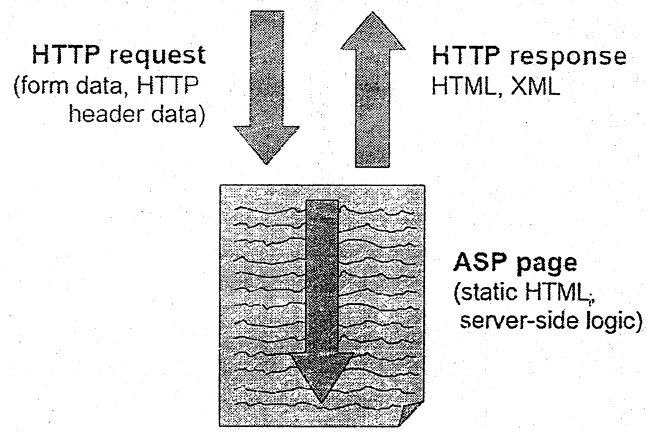
Background Web Technologies

Client-side technologies
HTML, DHTML, JavaScript
Server-side technologies
ASP (Active Server Pages
ASP.NET is the next generation of ASP

Background what is ASP?

Server-side programming technology
Consists of static HTML interspersed with script
ASP intrinsic objects (Request, Response, Server, Application, and Session) provide services
Commonly uses ADO to interact with databases
Application and session variables
Application and session begin/end events
ASP manages threads, database connections...

PRASANNA



Hello World.asp

```
<html>
<head><title>HelloWorld.asp</title></head>
<body>
<form method="post">
<input type="submit" id=button1 name=button1 value="Push Me" />
<%
 if (Request.Form("button1") <> "") then
Response.Write "<p>Hello, the time is " & Now()
end if
%>
</form>
</body>
</html>
```

Background ASP Success

- Simple procedural programming model
- Access to COM components
- ActiveX Data Objects (ADO)
- File System Object
- Custom components
- Script-based: no compiling, just edit, save & run
- VBScript, JScript – leverages existing skills
- Support for multiple scripting languages
- ASP has been very popular

Background ASP Challenges

- Coding overhead (too much code)
  - Everything requires writing code!
- Code readability (too complex; code and UI intermingled)
- Maintaining page state requires more code
- Reuse is difficult
- Supporting many types of browsers is difficult
- Deployment issues (e.g. DLL locking)
- Session state scalability and availability
- Limited support for caching, tracing, debugging, etc.
- Performance and safety limitations of script.

ASP.NET Overview

- ASP.NET provides services to allow the creation, deployment, and execution of Web Applications and Web Services
- Like ASP, 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 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 Key Features

Web Forms
Web Services
Built on .NET Framework
Simple programming model
Maintains page state
Multi-browser 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
Cookie less sessions

ASP.NET Overview Demo: Hello World.aspx

```
<%@ Page language="c#" %>
  <html>
     <head></head>
        <script runat="server">
           public void B_Click (object sender, System EventArgs e) {
           Label1.Text = "Hello Transflower, 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>
    </html>
```

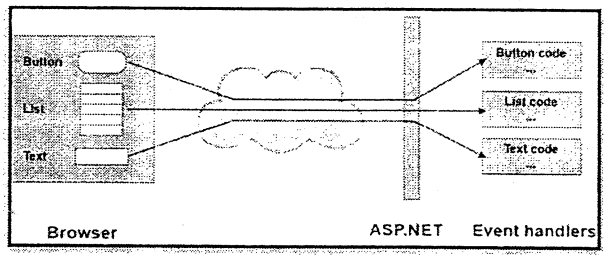
ASP.NET Overview Architecture

ASP.NET is built upon

- .NET Framework
- Internet Information Server (IIS)
- IIS MMC Snap-In (Internet Services Manager)
  - Tool to manage IIS
- Virtual Directories
  - Provides a mapping between URL and file path
    - E.g., on my machine the URL:
      - http://localhost/TFL15
      - maps to the file path:
      - C :\inetpub\wwwroot\TFL15

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



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 Postbacks

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 states

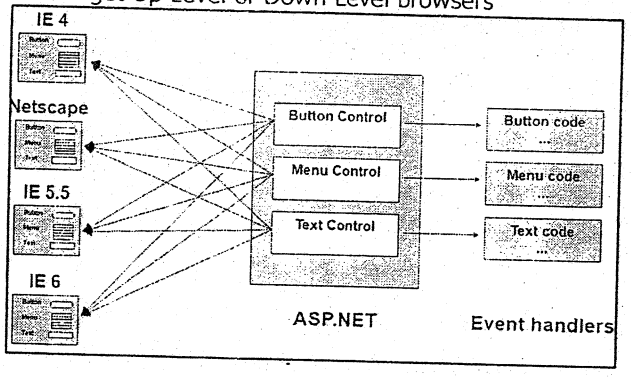
By default, ASP.NET maintains the state of all server-side controls during a postback
Can use method="post" or method="get"
Server-side control objects are automatically populated during postback
No state stored on server
Works with all browsers

Programming Model Server-Side Controls

Multiple sources of controls
Built-in
3rd party
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 Up Level or Down Level browsers



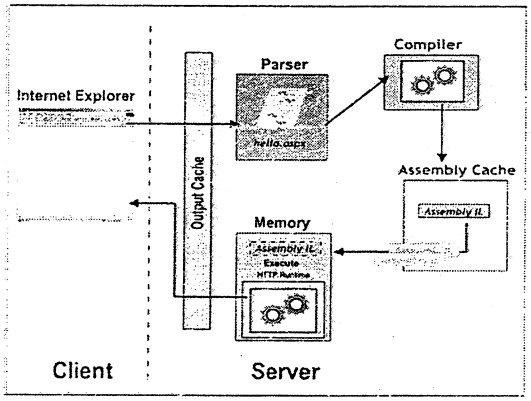
Programming Model Code behind pages

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" %>
```

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 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 Page Directive

Lets you specify page-specific attributes, e.g.

AspCompat: Compatibility with ASP
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

Only one page directive per .aspx file

Programming Basics 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

Programming Basics 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">
```

Tags and attributes are case-insensitive
Control properties can be set programmatically

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

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

Programming Basics Server Code Blocks

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

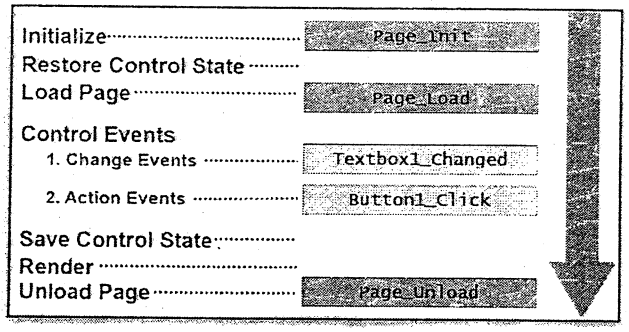
```
<script language="C#" runat=server>
<script language="VB" runat=server>
<script language="JScript" runat=server>
```

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

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



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;
}
```

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 = "Welcome to Transflower!";
 }
 // Rest of procedure executes on every request
}
```

Programming Basics Server Control Event

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!

Programming Basics 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";
 }
```

Programming Basics 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

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 Import Directive

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

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

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 ()

Server Controls
ASP.NET ships with ~50 built-in controls
Organized into logical families
HTML controls
Controls / properties map 1:1 with HTML
Web controls
Richer functionality
More consistent object model

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
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>
```

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

Server Controls HTML 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
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
Web Controls provide extensive properties to control display and format, e.g.

Font,
BackColor, ForeColor
BorderColor, BorderStyle, BorderWidth
Style, CssClass
Height, Width
Visible, Enabled

Four types of Web Controls

- Intrinsic controls
- List controls
- Rich controls
- Validation controls

Server Controls 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>

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 List Controls

Controls that handle repetition
Supported controls

<asp:dropdownlist>
<asp:listbox>
<asp:radiobuttonlist>
<asp:checkboxlist>
<asp:repeater>
<asp:datalist>
<asp:datagrid>

Repeater, DataList and DataGrid 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

Server Controls CheckBoxList and 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>
```

Server Control Rich Control
Custom controls with rich functionality
Supported Controls

<asp:calendar>
<asp:adrotator>

More will be added
3rd party controls are coming

Server Control Validation Control

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

<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
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 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
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!";
 }
}
```

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

Data Binding How to Populate Server Controls?
Specify the data in the control's tags
Not dynamic: can't get data from a database
Write code that uses the control's object model
This is okay if you need to populate a simple value or list, but quickly gets too complicated for populating sophisticated displays
Data binding
Create an object that holds the data
(Dataset, Array, string, int, etc.)
Associate that object with the control

Data Binding

Provides a single simple yet powerful way to populate Web Form controls with data
Enables clean separation of code from UI
Supports binding to any data source Properties, expressions, method calls,
Collections (Array, Hashtable, etc.)
Dataset, DataTable, DataView, DataReader
XML
One way snapshot model
Requires code to reapply to data model
Allows you to specify an expression
When the DataBind method of the control is called, the expression is evaluated and bound DataBind for a single control (and subcontrols)
Page.DataBind binds all controls on a page

Works for scalars, e.g. Label control
Works for lists, e.g. DropDown control, ListBox control, etc.
Enables the use of templates

Data Binding Scalar Expressions
Data binding expression: <%# expression %>
Expression is evaluated when DataBind() is called

```
<asp:Label id=label1
Text=<%# "The result is " + (1 + 2) +   ", the time is " +
DateTime.Now.ToLongTimeString() %>  runat="server" />

public void Page_Load(object s, EventArgs e) {
 if (! Page.IsPostBack)
 Page.DataBind();
}
```

Data Binding Simple List
Data binding a list creates a user interface element for each item in the list
Each item contains text (displayed to user) and an optional value (not displayed)

The simple list controls:

```
<asp:ListBox>
Single or multiple select
<asp:DropDownList>
<asp:RadioButtonList>
<asp:CheckBoxList>
```

Steps to data bind a list control
Declare the list control
Optionally set DataValueField and DataTextField
Set its DataSource
Call DataBind() method

Data Binding Database

Data binding can be used to populate server controls with data from a database
Each UI element corresponds to a row
Bind to a DataReader (preferred)
Bind to a DataView of a DataSet
Specify value and text with DataValueField and DataTextField, respectively
Each of these corresponds to a column.

Data Binding Data Source Example

```
DataView GetSampleData()
{
DataSet ds;
SqlConnection cxn;
SqlDataAdapter adp;
cxn = new SqlConnection("server=localhost; " +
"uid=sa;pwd=;database=Northwind");
adp = new SqlDataAdapter("select CategoryID, CategoryName from Categories",
cxn);
ds = new DataSet();
adp.Fill(ds, "Categories");
return ds.Tables["Categories"].DefaultView;
}
```

Data Binding List Binding Example

```
void Page_Load(object s, EventArgs e)
{
ListBox1.DataSource = GetSampleData();
ListBox1.DataValueField = "CategoryID";
ListBox1.DataTextField = "CategoryName";
ListBox1.DataBind();
}

<asp : ListBox id="ListBox1" runat="server" />
void Page_Load(object s, EventArgs e)
{
   ListBox1.DataBind();
}
```

```
<asp:ListBox id="ListBox1" runat="server" DataSource=<%#GetSampleData()%>
DataValueField="CategoryID"
DataTextField="CategoryName" />
```

Data Binding DataGrid
Full-featured list output
Default look is a grid
Default is to show all columns, though you can specify a subset of columns to display
Columns can be formatted with templates
Optional paging
Updateable

Data Binding binding to all Columns
Binding all columns in the datasource
Declare an <asp:DataGrid>
Set its DataSource
Call DataBind()

```
void Page_Load(object s, EventArgs e)
{
        myDataGrid.DataSource = GetSampleData();
        myDataGrid.DataBind();
}
<asp:datagrid id=myDataGrid runat="server" />
```

Data Binding to Specific Column
By default, DataGrid will display all columns
To control columns to display:
Set AutoGenerateColumns="false"
Specify Columns property
Add column definition
BoundColumn
TemplateColumn
ButtonColumn, EditCommandColumn, HyperlinkColumn

Binding to specific columns in the datasource
Declare an <asp:DataGrid>
Declare its Columns collection
Set its DataSource
Call its DataBind() method

```
<asp:datagrid id=myDataGrid
        autogeneratecolumns=false runat=server>
<Columns>
<asp:BoundColumn HeaderText="Id" DataField="title_id" />
<asp:BoundColumn HeaderText="Title" DataField="title"/>
</Columns>
</asp:datagrid>
```

Data Binding DataGrid Paging

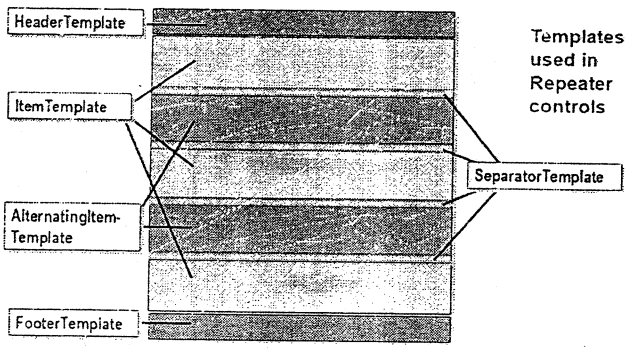
When there is too much data to display in one screen, a DataGrid can provide automatic paging
Set AllowPaging="true"
Set PageSize=5
Handle OnPageIndexChanged event
Set page index
Fetch data
Re-bind data

Data Binding Template

Templates provide a powerful way to customize the display of a server control
Customize structure – not just style
Can use controls or other HTML within a template
3rd party controls can expose new templates
With data binding, templates specify a set of markup (HTML or server controls) for each bound piece of data
Not just specifying formatting and style for a column
However, templates are not limited to data binding
No fixed set of templates
Controls may define their own and expose any number of them

Standard templates for list-bound controls

HeaderTemplate: rendered once before all data bound rows
ItemTemplate: rendered once for each row in the data source
AlternatingItemTemplate: like ItemTemplate, but when present is used for every other row
SeparatorTemplate: rendered between each row
FooterTemplate: rendered once, after all data bound rows



Data Binding in Template

Templates need to access the bound data
Container is an alias for the template's containing control
DataItem is an alias for the current row of the datasource
DataBinder.Eval is a utility function provided to retrieve and format data within a template

```
<%# DataBinder.Eval(Container.DataItem, "price", "$ {0}") %>
```

Data Binding Repeater Control

Provides simple output of a list of items
No inherent visual form
Templates provide the visual form
No paging
Can provide templates for separators
Not updateable

```
<asp:Repeater id="repList" runat="server">
 <template name="HeaderTemplate">
   <table>
    <tr><td>Title</td><td>Type</td></tr>
    </template>
    <template name="ItemTemplate">
     <tr>
       <td><%# DataBinder.Eval(Container.DataItem,"title_id") %></td>
       <td><%# DataBinder.Eval(Container.DataItem,"type") %></td>
     </tr>
    </template>
   <template name="FooterTemplate">
 </table>
 </template>
 </asp:Repeater>
```

Data Binding DataList Control

Provides list output with editing
Default look is a table
Customized via templates
Directional rendering (horizontal or vertical)
Single and multiple selection
Alternate item
Updateable
No paging

```
void Page_Load(object s, EventArgs e) {
myDataGrid.DataSource = GetSampleData();
myDataGrid.DataBind();
}
<asp:datalist id=myDataList runat=server>
<template name="itemtemplate">
<b>Title id:</b>
<%# DataBinder.Eval(Container.DataItem, "title_id") %>
<br> <b>Title:</b>
<%# DataBinder.Eval(Container.DataItem, "title") %>
</template>
</asp:datalist>
```

Building Data-Driven ASP.NET Application

Simplified Data Binding

Data binding expressions are now simpler and support hierarchical (XML) data binding

```
<!—classical  ASP.NET 1.x data binding expression -->
<%# DataBinder.Eval (Container.DataItem, "Price") %>
<!-- Equivalent ASP.NET data binding expression -->
<%# Eval ("Price") %>
<!-- XML data binding -->
<%# XPath ("Price") %>
```

Data Source Controls

Declarative (no-code) data binding

Name	Description
SQLDataSource	Connect data-binding controls to SQL database
AccessDataSource	Connect data-binding controls to Access database
XMLDataSource	Connect data-binding controls to XML
ObjectDataSource	Connect data-binding controls to data components
SiteMapDataSource	Connects site navigation controls to site map data

SqlData Source

Declarative data binding to SQL databases
- Any database served by a managed provider

Two-way data binding
- SelectCommand defines query semantics
- InsertCommand, UpdateCommand, and DeleteCommand define update semantics

Optional caching of query results
Parameterized operation

Using SQL Data Source

```
<asp:SqlDataSource ID="Titles" RunAt="server"
  ConnectionString="server=localhost;database=pubs;integrated security=true"
  SelectCommand="select title_id, title, price from titles" />
<asp:DataGrid DataSourceID="Titles" RunAt="server" />
```

Name	Description
ConnectionString	Connection string used to connect to data source
SelectCommand	Command used to perform queries
InsertCommand	Command used to perform inserts
UpdateCommand	Command used to perform updates

DeleteCommand	Command used to perform deletes
DataSourceMode	Specifies whether DataSet or DataReader is used(default=DataSet)
ProviderName	Specifies provider (default=SQL Server .NET provider)

SqlData Source and Caching

SqlDataSource supports declarative caching of results through these properties:

Name	Description
EnableCaching	Specifies whether caching is enabled (default=false)
CacheDuration	Length of time in seconds results should be cached
CacheExpirationPolicy	Specifies whether cache duration is sliding or absolute
CacheKeyDependency	Creates dependency on specified cache key
SqlCacheDependency	Creates dependency on specified database entity

Caching Query Results

```
<asp:SqlDataSource ID="Countries" RunAt="server"
      ConnectionString="server=localhost;database=northwind;..."
      SelectCommand="select distinct country from customers order by country"
      EnableCaching="true" CacheDuration="60" />
      <asp:DropDownList ID="MyDropDownList" DataSourceID="Countries"
      DataTextField="country" AutoPostBack="true" RunAt="server" />
```

Parameterized Command

XxxParameters properties permit database commands to be parameterized
Example: Get value for WHERE clause in SelectCommand from query string parameter or item selected in drop-down list
Example: Get value for WHERE clause in DeleteCommand from GridView

XxxParameter types specify source of parameter values

XxxParameters Properties

Name	Description
SelectParameters	Specifies parameter for select command
InsertParameters	Specifies parameter for insert command

UpdateParameters	Specifies parameter for update command
DeleteParameters	Specifies parameter for delete command
FilterParameters	Specifies parameter for filterExpression

XxxParameters

Name	Description
Parameter	Binds a replaceable parameter to a data field
Controlparameter	Binds a replaceable parameter to a control property
Cookieparameter	Binds a replaceable parameter to a cookie value
FormParameter	Binds a replaceable parameter to a form field
ProfileParameter	Binds a replaceable parameter to a profile property
QueryStringParameter	Binds a replaceable parameter to a query string parameter
SessionParameter	Binds a replaceable parameter to a session variable

Using Control Parameter

```
<asp:SqlDataSource ID="Countries" RunAt="server"
ConnectionString="server=localhost;database=northwind;..."
SelectCommand="select distinct country from customers order by country"/>
<asp:SqlDataSource ID="Customers" RunAt="server"
      ConnectionString="server=localhost;database=northwind;..."
      SelectCommand="select * from customers where country=@Country">
      <SelectParameters>
      <asp:ControlParameter Name="Country" ControlID="MyDropDownList"
                            PropertyName="SelectedValue" />
       </SelectParameters>
</asp:SqlDataSource>
<asp:DropDownList ID="MyDropDownList" DataSourceID="Countries"
       DataTextField="country" AutoPostBack="true" RunAt="server" />
<asp:DataGrid DataSourceID="Customers" RunAt="server" />
```

Calling Stored Procedures

```
<asp:SqlDataSource ID="Countries" RunAt="server"
  ConnectionString="server=localhost;database=northwind;..."
  SelectCommand="proc_GetCountries" />
<asp:SqlDataSource ID="Customers" RunAt="server"
  ConnectionString="server=localhost;database=northwind;..."
  SelectCommand="proc_GetCustomers">
  <SelectParameters>
    <asp:ControlParameter Name="Country" ControlID="MyDropDownList"
      PropertyName="SelectedValue" />
  </SelectParameters>
</asp:SqlDataSource>
<asp:DropDownList ID="MyDropDownList" DataSourceID="Countries"
     DataTextField="country" AutoPostBack="true" RunAt="server" />
<asp:DataGrid DataSourceID="Customers" RunAt="server" />
```

```
CREATE PROCEDURE proc_GetCustomers

@Country nvarchar (32) AS
  SELECT * FROM Customers
  WHERE Country = @Country
GO
CREATE PROCEDURE proc_GetCountries AS
  SELECT DISTINCT Country
  FROM Customers
  ORDER BY Country
GO
```

XMLDataSource

- Declarative data binding to XML data
- Supports caching and XSL transformations
- One-way data binding only; no updating

```
<asp:XmlDataSource ID="Rates" DataFile="Rates.xml" RunAt="server" />

<asp:TreeView ID="MyTreeView" DataSourceID="Rates" RunAt="server" />
```

Key XMLDataSource Properties

Name	Description
DataFile	Path to file containing XML data
TransformFile	Path to file containing XSL style sheet
EnableCaching	Specifies whether caching is enabled (default=false)
CacheDuration	Length of time in seconds data should be cached
CacheExpirationPolicy	Specifies whether cache duration is sliding or absolute
CacheKeyDependency	Creates dependency on specified cache key
XPath	Path expression used to filter data

ObjectDataSource

Declarative binding to data components
- Leverage middle-tier data access components
- Keep data access code separate from UI layer

Two-way data binding
- SelectMethod, InsertMethod, UpdateMethod, and DeleteMethod

Optional caching of query results

Parameterized operation

Key ODS Properties

Name	Description
TypeName	Type name of data component
SelectMethod	Method called on data component to perform queries
InsertMethod	Method called on data component to perform insert
UpdateMethod	Method called on data component to perform update
DeleteMethod	Method called on data component to perform delete
EnableCaching	Specifies whether caching is enabled(default=false)

Key ODS Properties Count

Name	Description
CacheDomain	Length of time in seconds data should be cached
SqlCacheDependency	Creates dependency on specified database entity
SelectParameter	Specifies parameter for SelectMethod
InsertParameter	Specifies parameter for InsertMethod
UpdateParameter	Specifies parameter for UpdateMethod
DeleteParameter	Specifies parameter for DeleteMethod

Initialization and Clean-Up

ObjectDataSource.SelectMethod et al can identify static methods or instance methods

If instance methods are used:
- ODS creates new class instance on each call
- Class must have public default constructor

Use ObjectCreated and ObjectDisposing events to perform specialized initialization or clean-up work on data components

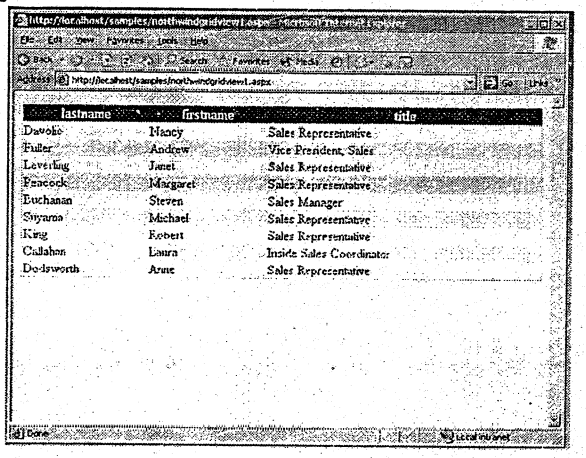
The GridView Control

- Enhanced DataGrid control
  - Renders sets of records as HTML tables
- Built-in sorting, paging, selecting, updating, and deleting support
- Supports rich assortment of field types, including CheckBoxFields
  - Declared in <Columns> element
- Highly customizable UI

GridView Example

```
<asp:SqlDataSource ID="Employees" RunAt="server"
ConnectionString="server=localhost;database=northwind;..."
 SelectCommand="select lastname, firstname, title from employees" />
<asp:GridView DataSourceID="Employees" Width="100%" RunAt="server" />
```

Output



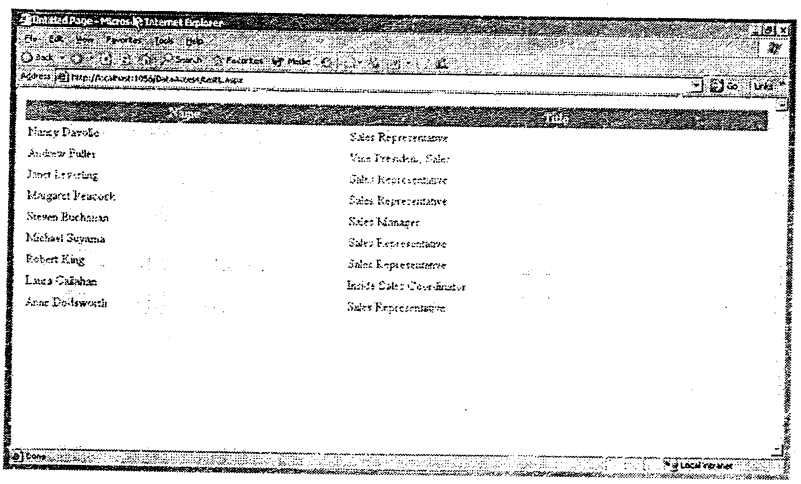
GridView Field Type

Name	Description
BoundField	Renders columns of text fields in data source
ButtonField	Renders columns of button(push button,image,or link
CheckBoxField	Renders Booleans as check boxes
CommandField	Renders columns of selecting and editing GridView data
HyperLinkField	Renders columns of hyperlinks
ImageField	Renders columns of images from URL text
TemplateField	Renders columns using HTML templates

Specifying Field Type

```
      <asp:SqlDataSource ID="Employees" RunAt="server"
      ConnectionString="server=localhost;database=northwind;..."
 SelectCommand="select photo, lastname, firstname, title from  employees" />
      <asp:GridView DataSourceID="Employees" Width="100%" RunAt="server"
      AutoGenerateColumns="false" >
      <Columns>
       <asp:TemplateField HeaderText="Name">
      <ItemTemplate>
        <%# Eval ("firstname") + " " + Eval ("lastname") %>
      </ItemTemplate>
    </asp:TemplateField>
    <asp:BoundField HeaderText="Title" DataField="title" />
  </Columns>
</asp:GridView>
```

Output

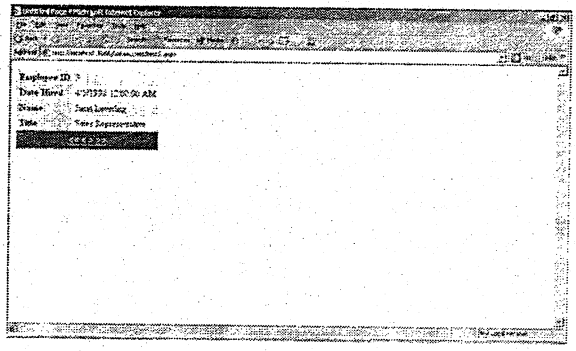


The DetailsView Control

- Renders individual records
  - Pair with GridView for master-detail views
  - Or use without GridView to display individual records
- Built-in paging, inserting, updating, deleting
- Uses same field types as GridView
  - Declared in <Fields> element
- Highly customizable UI

DetailsView Example

```
<asp:SqlDataSource ID="Employees" RunAt="server"
ConnectionString="server=localhost;database=northwind;..."
SelectCommand="select employeeid, photo, ... from employees" />
<asp:DetailsView DataSourceID="Employees" RunAt="server"
AllowPaging="true" AutoGenerateRows="false"
PagerSettings-Mode="NextPreviousFirstLast">
<Fields>
<asp:BoundField HeaderText="Employee ID" DataField="employeeid" />
<asp:BoundField HeaderText="Date Hired" DataField="hiredate" />
<asp:TemplateField HeaderText="Name">
<ItemTemplate>
  <%# Eval ("firstname") + " " + Eval ("lastname") %>
</ItemTemplate>
</asp:TemplateField>
<asp:BoundField HeaderText="Title" DataField="title" />
 </Fields>
</asp:DetailsView>
```



Inserting Updating and Deleting

Data controls supply editing UIs
- AutoGenerateXxxButton properties
- Insert/EditRowStyle properties

Data source controls supply editing logic
- Insert/Update/DeleteCommand properties
- Insert/Update/DeleteParameters properties
- Inserting/ed, Updating/ed, Deleting/ed events
- Visual Studio supplies the glue

Editing with GridView

```
<asp:SqlDataSource ID="Employees" RunAt="server"
      ConnectionString="server=localhost;database=northwind;..."
SelectCommand="select employeeid, lastname, firstname from employees"
UpdateCommand="update employees set lastname=@lastname, firstname=
@firstname where employeeid=@original_employeeid">
<UpdateParameters>
<asp:Parameter Name="EmployeeID" Type="Int32" />
<asp:Parameter Name="lastname" Type="String" />
<asp:Parameter Name="firstname" Type="String" />
</UpdateParameters>
</asp:SqlDataSource>
<asp:GridView DataSourceID="Employees" Width="100%" RunAt="server"
DataKeyNames="EmployeeID" AutoGenerateEditButton="true" />
```

Conflict Detection

First-in wins
- Update fails if data has changed
- Structure UpdateCommand accordingly
- Set ConflictDetection="CompareAllValues"

Last-in wins
- Update succeeds even if data has changed
- Structure UpdateCommand accordingly
- Set ConflictDetection="OverwriteChanges"

First-In-Wins Update

```
<asp:SqlDataSource ID="Employees" RunAt="server"
      ConnectionString="server=localhost;database=northwind;..."
      SelectCommand="select employeeid, lastname, firstname from employees"
      UpdateCommand="update employees set lastname=@lastname, firstname=
    @firstname where employeeid=@original_employeeid and lastname=
    @original_lastname and firstname=@original_firstname"
      ConflictDetection="CompareAllValues">
      <UpdateParameters>
    <asp:Parameter Name="EmployeeID" Type="Int32" />
    <asp:Parameter Name="lastname" Type="String" />
    <asp:Parameter Name="firstname" Type="String" />
      </UpdateParameters>
      </asp:SqlDataSource>
      <asp:GridView DataSourceID="Employees" Width="100%" RunAt="server"
      DataKeyNames="EmployeeID" AutoGenerateEditButton="true" />
```

Error Detection

Controls fire events after database updates
- GridView.RowUpdated
- DetailsView.ItemUpdated
- SqlDataSource.Updated, etc.

Event handlers receive "status" objects
- Reveal whether database exception occurred
- Allow exceptions to be handled or rethrown
- Reveal how many rows were affected

Handling Update Errors

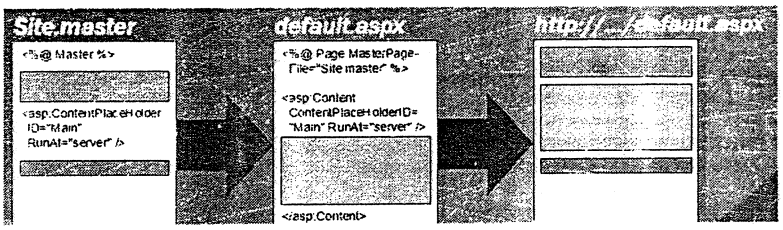
```
<asp:SqlDataSource ID="Employees" RunAt="server" ...
  UpdateCommand="..." OnUpdated="OnUpdateComplete">
    ...
</asp:SqlDataSource>
...
void OnUpdateComplete (Object source, SqlDataSourceStatusEventsArgs e)
{
    if (e.Exception != null) {
        // Exception thrown. Set e.ExceptionHandled to true to prevent
        // the SqlDataSource from throwing an exception, or leave it set
        // to false to allow SqlDataSource to rethrow the exception
    }
    else if (e.AffectedRows == 0) {
        // No exception was thrown, but no records were updated, either.
        // Might want to let the user know that the update failed
    }
}
```

Managing Look, Feel, and Layout with Visual Studio (ASP.NET)

Master Pages

Master Page Basics

Masters define common content and placeholders (<asp:ContentPlaceHolder>)
Content pages reference masters and fill placeholders with content (<asp:Content>)



Defining a Master Page

```
<%@ Master %>
<html>
  <body>
   <!-- Banner shown on all pages that use this master -->
   <table width="100%">
     <tr>
        <td bgcolor="darkblue" align="center">
          <span style="font-size: 36pt; color: white">ACME Inc.</span>
        </td>
      </tr>
    </table>
    <!-- Placeholder for content below banner -->
    <asp:ContentPlaceHolder ID="Main" RunAt="server" />
  </body>
</html>
```

Applying a Master Page

```
<%@ Page MasterPageFile="~/Site.master" %>
<asp:Content ContentPlaceHolderID="Main" RunAt="server">
  This content fills the place holder "Main" defined in the master page
</asp:Content>
```

Applying a Master Page to a Site

```
<configuration>
  <system.web>
    <pages masterPageFile="~/Site.master" />
  </system.web>
</configuration>
```

Applying a Master Page Programmatically

```
void Page_PreInit (Object sender, EventArgs e)
{
   Page.MasterPageFile = "~/Site.master";
}
```

Default Content

ContentPlaceHolder controls can define content of their own ("default content")
Default content is displayed ONLY if not overridden by content page

```
<%@ Master %>
  ...
<asp:ContentPlaceHolder ID="Main" RunAt="server">
  This is default content that will appear in the absence of a
  matching Content control in a content page.
<asp:ContentPlaceHolder>
```

The Page. Master Property

Retrieves reference to master page
Instance of class derived from System.Web.UI.MasterPage
Null if page doesn't have a master

Used to programmatically access content defined in the master page
Use FindControl for weak typing
Use public property in master page for strong typing (preferred)

Accessing a Control in the Master Page (Weak Typing)

In the Master Page
<asp:Label ID="Title" RunAt="server" />

In the Content Page
((Label) Master.FindControl ("Title")).Text = "Orders";

Accessing a Control in the Master Page (Strong Typing)

In the Master Page

```
<asp:Label ID="Title" RunAt="server" />
  <script language="C#" runat="server">
      public string TitleText
      {
            get { return Title.Text; }
            set { Title.Text = value; }
      }
 </script>
```

In the Content page

```
Master.TitleText = "Orders";
```

Nesting Master Pages

Master pages can be nested
Master pages that have masters must contain only Content controls, but Content controls can contain ContentPlaceHolders

```
<!-- Orders.Master -->
<%@ Master MasterPageFile="~/Site.Master" %>
<asp:Content ContentPlaceHolderID="..." RunAt="server">
  <asp:ContentPlaceHolder ID="..." RunAt="server">
    ...
  </asp:ContentPlaceHolder>
<asp:Content>
```

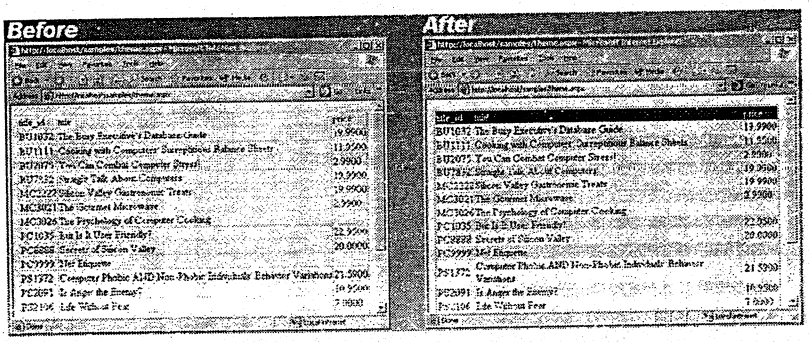
Themes and Skins

Mechanism for theming controls, pages, and sites by group-initializing control properties
Skin = Visual attributes for control(s)
- Physically stored in .skin files
- Default skins and named skins

Theme = Collection of one or more skins
- Physically stored in Themes subfolders
- Global themes and local themes

Applying a Theme to a Page

```
<%@ Page Theme="BasicBlue">
```



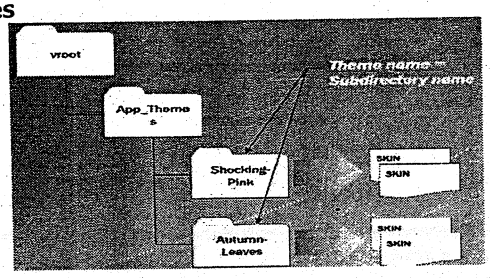
Applying a Theme to a Site

```
<configuration>
  <system.web>
    <pages theme="BasicBlue" />
  </system.web>
</configuration>
```

Applying a Theme Programmatically

```
void Page_PreInit (Object sender, EventArgs e)
{
    Page.Theme = "BasicBlue";
}
```

Themes



Defining Skins

```
<!-- Default look for DropDownList controls -->
<asp:DropDownList runat="server" BackColor="hotpink" ForeColor="white" />
<!-- Default look for DataGrid controls -->
<asp:DataGrid runat="server" BackColor="#CCCCCC" BorderWidth="2pt"
  BorderStyle="Solid" BorderColor="#CCCCCC" GridLines="Vertical"
  HorizontalAlign="Left">
  <HeaderStyle ForeColor="white" BackColor="hotpink" />
  <ItemStyle ForeColor="black" BackColor="white" />
  <AlternatingItemStyle BackColor="pink" ForeColor="black" />
</asp:DataGrid>
  ...
```

Named Skins

Skins without SkinIDs are default skins
Skins with SkinIDs are named skins
SkinIDs must be unique per control type
Can be defined in same SKIN file as default skins or in separate files
Use controls' SkinID properties to apply named skins

Defining Named Skins

```
<!-- Default look for DropDownList controls -->

<asp:DropDownList runat="server" BackColor="blue" ForeColor="white"
  SkinID="Blue" />

<!-- Default look for DataGrid conotrols -->

<asp:DataGrid runat="server" BackColor="#CCCCCC" BorderWidth="2pt"
  BorderStyle="Solid" BorderColor="#CCCCCC" GridLines="Vertical"
  HorizontalAlign="Left" SkinID="Blue">
  <HeaderStyle ForeColor="white" BackColor="blue" />
  <ItemStyle ForeColor="black" BackColor="white" />
  <AlternatingItemStyle BackColor="lightblue" ForeColor="black" />
</asp:DataGrid>
...
```

Using a Named Skin

```
<asp:DropDownList ID="Countries" SkinID="Blue" RunAt="server" />
```

The EnableTheming Property

Supported by all pages and controls
Defaults to true
Set EnableTheming to false to disable theming for individual controls or entire pages

```
<asp:DropDownList ID="Countries" EnableTheming="false" RunAt="server" />
```

Other ASP.NET Controls

Category	Controls
Data controls	GridView and Details View
Data Source Controls	SqlDataSource,ObjectSource,XmlDataSource, etc
Login controls	Login,CreateUserWizard,PasswordRecovery, etc
Navigation Controls	Menu, TreeView ,and SiteMapPath
Web Part Controls	WebPartManager,WebPartZone, etc
UI Controls	File Upload, Bulleted List , MultiView, Wizard , etc

UI Controls

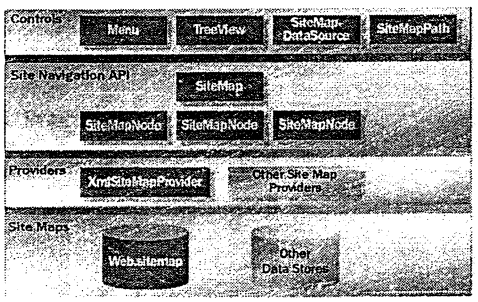
Name	Description
BulletedList	Renders bulleted lists of items
FileUpload	UI for uploading files to Web Servers
HiddenField	Renders hidden fields
ImageMap	Renders HTML image maps
MultiView	Defines multiple views displayed one at a time
View	Defines views in MultiView controls
Substitution	Designates non-cached regions of cached pages
Wizard	Guides users through stepwise procedures

Building Dynamic Navigation Systems

Site Navigation

- Navigation UIs are tedious to implement
  - Especially if they rely on client-side script
- New controls simplify site navigation
  - TreeView and Menu - Navigation UI
  - SiteMapDataSource - XML site maps
  - SiteMapPath - "Bread crumb" controls
- Public API provides foundation for controls
- Provider-based for flexibility

Site Navigation Schema



TreeView Controls

- Render hierarchical data as trees
  - Expandable and collapsible branches
  - Nodes are navigable, selectable, or static and can include check boxes
- Content defined by TreeNode objects
  - TreeNodes can be added declaratively, programmatically, or through data binding
  - TreeNodes can also be demand-loaded
- Highly customizable UI

Declaring a TreeView

```
<asp:TreeView RunAt="server>
<Nodes>
<asp:TreeNode Text="Training" RunAt="server">
<asp:TreeNode Text="Programming .NET" RunAt="server"
  Navigateurl="Classes.aspx?id=1" />
<asp:TreeNode Text="Programming ASP.NET" RunAt="server"
  NavigateUrl="Classes.aspx?id=2" />
<asp:TreeNode Text="Programming Web Services" RunAt="server"
  NavigateUrl="Classes.aspx?id=3" />
</asp:TreeNode>
<asp:TreeNode Text="Consulting" RunAt="server"
  NavigateUrl="Consulting.aspx" />
<asp:TreeNode Text="Debugging" RunAt="server"
  NavigateUrl="Debugging.aspx" />
</Nodes>
</asp:TreeView>
```

TreeView Properties

Name	Description
checkedNodes	Indicates which, if any, check boxes are checked
ExpandDepth	Specifies the TreeView's initial expand depth
Nodes	TreeNodeCollection representing tree node
SelectedNode	TreeNode representing the node that is currently selected
ShowCheckBoxes	Specifies which, if any, node types include check boxes
ShowExpandCollapse	Specifies whether expand/collapse indicators are shown
ShowLines	Specifies whether nodes are connected by lines

Key TreeView Properties Count

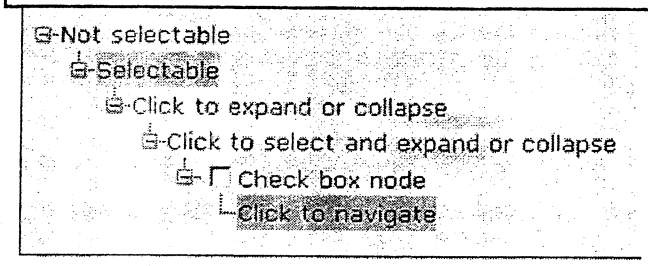
Name	Description
LevelStyle	Specifies appearance of nodes by level
NodeStyle	Specifies default appearance of nodes
RootNodeStyle	Specifies appearance of root nodes
LeafNodeStyle	Specifies appearance of leaf nodes
SelectedNodeStyle	Specifies appearance of selected nodes
HoverNodeStyle	Specifies appearance of nodes when cursor hovers overhead
ImageUrl properties	Identify expand/collapse indicator images and other

Key TreeNode Properties

Name	Description
Checked	Indicates whether the nodes check box is selected
ImageUrl	URL of image to display next to node text
NavigateUrl	URL to navigate to when node is clicked
SelectAction	Action to take when non-navigable node is clicked
Selected	Indicates whether node is currently selected
ShowCheckBox	Specifies whether node includes a check box
Test	Node text

Example

```
<asp:TreeView ShowLines="true" Font-Name="Verdana" Font-Size="10pt" ... >
<SelectedNodeStyle BackColor="Yellow" />
<HoverNodeStyle BackColor="LightBlue" />
<Nodes>
<asp:TreeNode Text="Not selectable" SelectAction="None" RunAt="server">
<asp:TreeNode Text="Selectable" SelectAction="Select" RunAt="server" >
<asp:TreeNode Text="Click to expand or collapse"
  SelectAction="Expand" Runat="server">
<asp:TreeNode Text="Click to select and expand or collapse"
  SelectAction="SelectExpand" Runat="server">
<asp:TreeNode Text="Check box node" ShowCheckBox="true"
  Runat="server">
<asp:TreeNode Text="Click to navigate" NavigateUrl="..."
  Runat="server" /></asp:TreeNode>
</asp:TreeNode> </asp:TreeNode> </asp:TreeNode></asp:TreeNode>
</Nodes>
</asp:TreeView>
```



Tree View Events

Name	Description
checkedChanged	Fired when a check box is checked or unchecked
SelectedNodeChanged	Fired when the selected node changes
TreeNodeCollapsed	Fired when a branch is collapsed
TreeNodeExpanded	Fired when a branch is expanded
TreeNoeDataBound	Fired when a tree node binds to a data source
TreeNodePopulate*	Fired when a PopulateOnDemand node needs content

Using SelectedNode Changed

```
<asp:TreeView ID="Tree" OnSelectedNodeChanged="OnUpdate" RunAt="server">
<Nodes>
   ...
</Nodes>
</asp:TreeView>
  .
  .
  .
<script language="C#" runat="server">
void OnUpdate (Object sender, EventArgs e)
{
    // Get the text of the selected node
    string text = Tree.SelectedNode.Text;
      ...
}
</script>
```

Populating Nodes on Demand

```
<asp:TreeView OnTreeNodePopulate="OnPopulate" EnableClientScript="false"
  RunAt="server">
  <Nodes>
    <asp:TreeNode Text="Populate this node on demand"
      PopulateOnDemand="true" RunAt="server" />
  </Nodes>
</asp:TreeView>
  .
  .
  .
<script language="C#" runat="server">
void OnPopulate (Object sender, TreeNodeEventArgs e)
{
    // Called first time the populate-on-demand node is expanded
    TreeNode node = new TreeNode ("This node added dynamically");
    e.Node.ChildNodes.Add (node);
}
</script>
```

Menu Controls

Drop-down/fly-out menus for Web pages
Items are navigable or selectable
Can be oriented horizontally or vertically
Content defined by MenuItem objects
MenuItems can be added declaratively, programmatically, or through data binding
Highly customizable UI

Declaring a Menu

```
<asp:Menu Orientation="Horizontal" RunAt="server">
<Items>
<asp:MenuItem Text="Training" RunAt="server">
<asp:MenuItem Text="Programming .NET" RunAt="server"
  Navigateurl="Classes.aspx?id=1" />
<asp:MenuItem Text="Programming ASP.NET" RunAt="server"
  NavigateUrl="Classes.aspx?id=2" />
<asp:MenuItem Text="Programming Web Services" RunAt="server"
  NavigateUrl="Classes.aspx?id=3" />
</asp:MenuItem>
<asp:MenuItem Text="Consulting" RunAt="server"
  NavigateUrl="Consulting.aspx" />
<asp:MenuItem Text="Debugging" RunAt="server"
  NavigateUrl="Debugging.aspx" />
</Items>
</asp:Menu>
```

Key Menu Properties

Name	Description
Items	MenuItemCollection representing menu items
ItemWrap	Specifies whether menu item should wrap
Orientation	Specifies whether to orient menu horizontally or vertically
SelectedItem	MenuItem representing the item that is currently selected
StaticStyle properties	Specify appearance of static menus
DynamicStyle properties	Specify appearance of dynamic menus

Key MenuItem Properties

Name	Description
ImageUrl	URL of image to display next to menu item text
NavigateUrl	URL to navigate to when menu item is clicked
Selected	Indicates whether the item is currently selected
Text	Menu item text (visible to user)
ToolTip	Tooltip text displayed when cursor pauses over menu item
Value	Menu item value (not visible to user)

Menu Events

Name	Description
MenuItemClick	Fired when a menu item is clicked
MenuItemDataBound	Fired when a menu item binds to a data source

Using MenuItemClick

```
<asp:Menu ... OnMenuItemClick="OnClick" RunAt="server">
<Items>
...
</Items>
</asp:Menu>
 .
 .
 .
<script language="C#" runat="server">
 void OnClick (Object sender, MenuEventArgs e)
 {
  // Get the text of the selected menu item
  string text = e.Item.Text;
  ...
 }
</script>
```

SiteMapDataSource

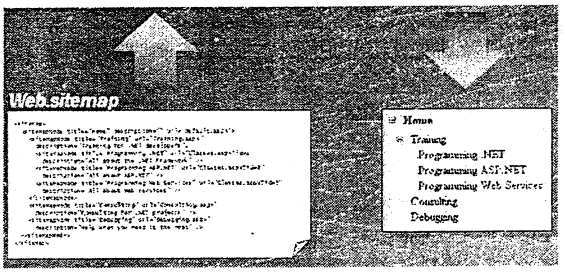
Data source control representing site maps
Site map = List of pages and URLs
Nodes can include descriptive text
Permits TreeViews and Menus to be populated with links through data binding
Supports "security trimming"
Specified nodes visible only to specified roles
Provider-based for flexible data storage

XML Site Map

```
<siteMap>
  <siteMapNode title="Home" description="" url="default.aspx">
  <siteMapNode title="Training" url="Training.aspx"
   description="Training for .NET developers">
  <siteMapNode title="Programming .NET" url="Classes.aspx?id=1"
   description="All about the .NET Framework" />
  <siteMapNode title="Programming ASP.NET" url="Classes.aspx?id=2"
   description="All about ASP.NET" />
  <siteMapNode title="Programming Web Services" url="Classes.aspx?id=3"
   escription="All about Web services" />
  </siteMapNode>
  <siteMapNode title="Consulting" url="Consulting.aspx"
description="Consulting for .NET projects" />
  <siteMapNode title="Debugging" url="Debugging.aspx"
description="Help when you need it the most" />
  </siteMapNode>
</siteMap>
```

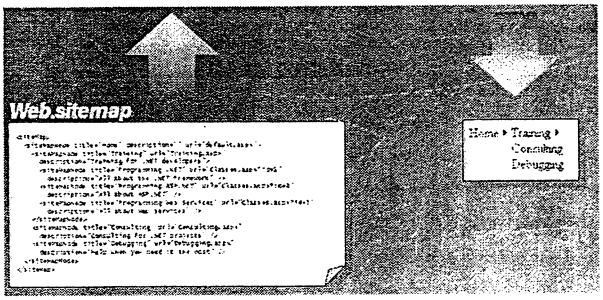
Tree View and Site Maps

```
<asp:SiteMapDataSource ID="SiteMap" RunAt="server" />
<asp:TreeView DataSourceID="SiteMap" RunAt="server" />
```



Menus and Site Maps

```
<asp:SiteMapDataSource ID="SiteMap" RunAt="server" />
<asp:Menu DataSourceID="SiteMap" RunAt="server" />
```



Changing the file Name

```
<configuration>
  <system.web>
   <siteMap>
     <providers>
       <remove name="AspNetXmlSiteMapProvider" />
<add name="AspNetXmlSiteMapProvider"
     type="System.Web.XmlSiteMapProvider, System.Web, ..."
     siteMapFile="Acme.sitemap" />
     </providers>
   </siteMap>
 </system.web>
</configuration>
```

<siteMapNode> Attributes

Name	Description
description	Description of node
role	Role or roles for which this node is visible
title	Title of this node
url	URL of this node

Security trimming

Visible to everyone

```
<siteMap>
  <siteMapNode title="Home" description="" url="default.aspx">
    <siteMapNode title="Announcements" url="Announcements.aspx"
      description="Information for all employees" />
    <siteMapNode title="Salaries" url="Salaries.aspx"
      description="Salary data" roles="Managers,CEOs" />
  <siteMapNode>
</siteMap>
```

Only visible to CEO and managers

Enabling Security trimming

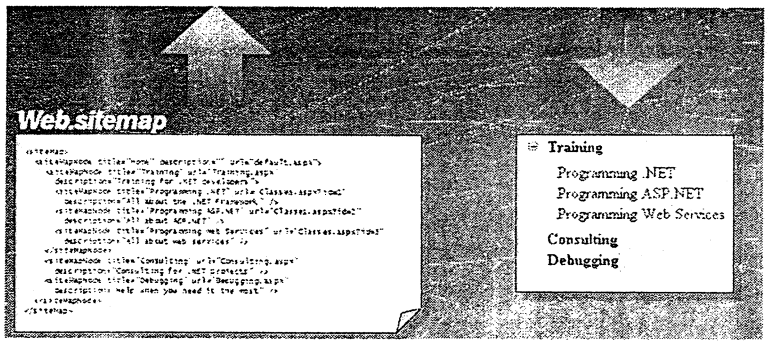
```
<configuration>
  <system.web>
    <siteMap>
      <providers>
        <remove name="AspNetXmlSiteMapProvider" />
        <add name="AspNetXmlSiteMapProvider"
          type="System.Web.XmlSiteMapProvider, System.Web, ..."
          securityTrimmingEnabled="true"
          siteMapFile="web.sitemap" />
      </providers>
    </siteMap>
  </system.web>
</configuration>
```

SiteMapDataSource Properties

Name	Description
Provider	Provider used to obtain site map data
SiteMapProvider	Name of provider used to obtain site map data
ShowStartingNode	Specifies whether to show the root node
StarFormCurrentNode	Specifies whether starting node should be the root node (false) or the current node (true). Default=false
StartingNodeOffset	Starting node identified by level (default=0)
StartingNodeURL	Starting node identified by url

Handling the Root Site Map Node

```
<asp:SiteMapDataSource ID="SiteMap" ShowStartingNode="false"
RunAt="server" />
<asp:TreeView DataSourceID="SiteMap" RunAt="server" />
```



SiteMapPath Control

"Bread crumbs" showing path to page
- By default, renders current node as static text
- By default, renders parent nodes as hyperlinks
- Highly customizable UI
  - Nodes can be stylized and templatized
  - Separators can be stylized and templatized
- Integrates with site map providers to acquire and path info

Using SiteMapPath

```
<asp:SiteMapPath RunAt="server" />
```

Home > Training > Programming ASP.NET

Key SiteMapPath Properties

Name	Description
CurrentNodeStyle	Style used to render the current node
CurrentNodeTemplate	HTML template used to render the current node
NodeStyle	Style used to render non-current nodes
NodeStyleTemplate	HTML template used to render non-current nodes
PathSeparator	Text used for node separators (default=">")
PathSeparatorStyle	Style used to render node separators
PathSeparatorTemplate	HTML template used to render node separators

Stylizing SiteMapPath

```
<asp:SiteMapPath Font-Name="Verdana" Font-Size="10pt" RunAt="server">
  <CurrentNodeStyle Height="24px" BackColor="Yellow" Font-Bold="true" />
  <NodeStyle Height="24px" />
  <PathSeparatorTemplate>
    <ItemTemplate>
      <asp:Image ImageUrl="~/images/arrow.gif" RunAt="server" />
    </ItemTemplate>
  </PathSeparatorTemplate>
</asp:SiteMapPath>
```

SiteMap Providers

- Site maps are provider-based
  - Provider interprets site map data and provides it to SiteMapDataSource controls
  - Provider also tracks current position and provides it to SiteMapPath controls
- ASP.NET 2.0 ships with one provider
  - XmlSiteMapProvider
- Use custom providers for other data stores

SiteMap API

- System.Web.SiteMap represents site maps
  - RootNode property identifies root node
  - CurrentNode property identifies current node
- SiteMapNode represents nodes
  - Interrogate properties of node
  - Walk up, down, and sideways in the hierarchy
- The magic underlying SiteMapPath controls
- Foundation for building controls of your own

Using the SiteMap API

```
// Write the title of the current node to a Label control

Label1.Text = SiteMap.CurrentNode.Title;
// Write the path to the current node to a Label control
SiteMapNode node = SiteMap.CurrentNode;
StringBuilder builder = new StringBuilder (node.Title);
while (node.ParentNode != null) {
    node = node.ParentNode;
    builder.Insert (0, " > ");
    builder.Insert (0, node.Title);
}

Label1.Text = builder.ToString ();
```

The ImageMap Control

```
<asp:ImageMap ImageUrl="Shapes.jpg" OnClick="OnUpdate" RunAt="server">
  <asp:CircleHotSpot X="50" Y="50" Radius="50" PostBackValue="Circle"
    AlternateText="Circle" HotSpotMode="Postback" RunAt="server" />
  <asp:RectangleHotSpot Left="0" Top="100" Right="100" Bottom="200"
    PostBackValue="Rectangle" AlternateText="Rectangle"
    HotSpotMode="Postback" RunAt="server" />
  <asp:PolygonHotSpot Coordinates="50, 200, 0, 300, 100, 300"
    PostBackValue="Triangle" AlternateText="Triangle"
    HotSpotMode="Postback" RunAt="server" />
</asp:ImageMap>
  ...
<script language="C#" runat="server">
void UpdateLabel (Object sender, ImageMapEventArgs e)
{
  ...
}
</script>
```

The FileUpload Control

```
<asp:FileUpload ID="UploadControl" RunAt="server" />
<asp:Button Text="Upload" OnClick="OnUpload" RunAt="server" />
  .
  .
<script language="C#" runat="server">
void OnUpload (Object sender, EventArgs e)
{
    if (FileUploadControl.HasFile) {
        string name = UploadControl.PostedFile.FileName;     // Path name
        Stream bits = UploadControl.PostedFile.InputStream; // Contents
          ...
        // Use the SaveAs method to persist to a local file
        FileInfo file = new FileInfo (UploadControl.PostedFile.FileName);
        UploadControl.SaveAs (Server.MapPath ("~/Uploads/" + file.Name);
    }
}
</script>
```

The MultiView Control

```
<asp:MultiView ID="Main" ActiveViewIndex="0" RunAt="server">
  <asp:View RunAt="server">
    ...
  </asp:View>
  <asp:View RunAt="server">
    ...
  </asp:View>
  <asp:View RunAt="server">
    ...
  </asp:View>
</asp:MultiView>
  .
  .
void OnSwitchView (Object sender, EventArgs e)
{
    Main.ActiveViewIndex = 1; // Switch views
}
```

Declarative View Switching

```
<asp:MultiView ID="Main" ActiveViewIndex="0" RunAt="server">
  <asp:View RunAt="server">
    ...
    <asp:Button CommandName="SwitchViewByIndex" CommandArgument="1"
      Text="Switch to view 2" RunAt="server" />
  </asp:View>
  <asp:View RunAt="server">
    ...
    <asp:Button CommandName="SwitchViewByIndex" CommandArgument="0"
      Text="Switch to view 1" RunAt="server" />
  </asp:View>
</asp:MultiView>
```

IIS

Internet Information Server

- Microsoft's web server
- Foundation for ASP.NET
- Runs in inetinfo.exe process
- Also FTP, NNTP, SMTP
- Shared resources
  - Default location c:\inetpub\wwwroot
  - Internet Services Manager
- A Microsoft Management Console (MMC) snap-in

IIS Virtual Directories

Provides a level of indirection from URL to actual file locations on the server

For example, the file for the url: **http://myServer/TransflowerOnline/tfl.asp**
could be mapped to the physical location: **d:\myFolder\ TransflowerOnline\tfl.asp**

Web Application

What Is a Web Application?
All resources (files, pages, handlers, modules, executable code, etc.) within a virtual directory and its subdirectories

- Configuration files
- Shared data (application variables)
- global.asax
- Scopes user sessions

A session is a series of web page hits by a single user within a block of time
Shared data (session variables)

WebApplication global.asax

Located at application root
Can define and initialize application variables and session variables
Specify object to create with class, COM ProgID or COM ClassID
Be careful: use of shared objects can cause concurrency errors or blocking!

```
<object id="items" runat="server"
  scope="application"
  class="System.Collections.ArrayList" />
```

Can contain user-created code to handle application and session events (just like ASP)
Application_OnStart, Application_OnEnd
Session_OnStart, Session_OnEnd

```
void Application_OnStart () {
  Application ["startTime"]=DateTime.Now.ToString();
}
void Session_OnStart() {
  Session["startTime"]=DateTime.Now.ToString();
}
```

Can use code-behind
Can contain application-level directives

```
<%@ Application Description="This is my app…" %>
<%@ Application Inherits="MyBaseClass" %>
<%@ Application Src="Global.cs" %>
            Visual Studio.NET uses this
<%@ Import Namespace="System.Collections" %>
<%@ Assembly Name="MyAssembly.dll" %>
```

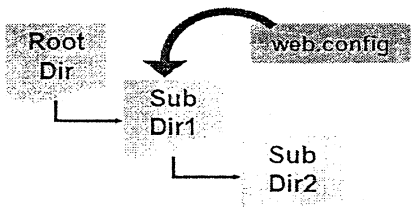
Configuration

Goal
Provide extensible configuration for admins & developers to hierarchically apply settings for an application

Solution
Store configuration data in XML text files
Format is readable and writable by people and machines
Settings specified in configuration sections, e.g.
Security, SessionState, Compilation, CustomErrors, ProcessModel, HTTPHandlers, Globalization, AppSettings, WebServices, WebControls, etc.
Configuration information stored in web.config
It is just a file, no DLL registration, no Registry settings, no Metabase settings
<!-- web.config can have comments -->

Configuration Hierarchy

Configuration files can be stored in application folders
Configuration system automatically detects changes
Hierarchical configuration architecture
Applies to the actual directory and all subdirectories



Configuration web.config Sample

```
<configuration>
 <configsections>
      <add names="httpmodules"
           type="System.Web.Config.HttpModulesConfigHandler"/>
      <add names="sessionState"
           type="..."/>
 </configsections>

 <httpModules>
 <!--- http module subelements go here -->
 </httpModules>
<sessionState>
    <!--- sessionstate subelements go here -->
  </sessionState>
</configuration>
```

Configuration Hierarchy

Standard machine-wide configuration file
Provides standard set of configuration section handlers
Is inherited by all Web Applications on the machine

C:\WINDOWS\Microsoft.NET\...\...\config\machine.config

Configuration User Defined Settings

Create web.config in appropriate folder

```
<configuration>
<appSettings>
 <add key="CxnString"
      value="localhost;uid=sa;pwd=;Database=tfl"/>
</appSettings>
</configuration>
```

Retrieve settings at run-time

```
string cxnStr = ConfigurationSettings.AppSettings["CxnString"];
```

Custom Configuration Handlers

Extend the set of section handlers with your own
Implement the interface:
`System.Web.Configuration.IConfigurationSectionHandler`
Add to web.config or machine.config

Tracing

ASP.NET supports tracing
- Easy way to include "debug" statements
  - No more messy Response.Write() calls!
  - Debug statements can be left in, but turned off
- Great way to collect request details
  - Server control tree
  - Server variables, headers, cookies
  - Form/Query string parameters
  - Tracing provides a wealth of information about the page
- Can be enabled at page- or application- level

Tracing Methods and Properties

Methods
- Trace.Write: Writes category and text to trace
- Trace.Warn: Writes category and text to trace in red

Properties
- Trace.IsEnabled: True if tracing is turned on for the application or just that page
- Trace.Mode: SortByTime, SortByCategory

Implemented in System.Web.TraceContext class

Page Level Tracing

To enable tracing for a single page:
- Add trace directive at top of page
  - <%@ Page Trace="True" %>
- Add trace calls throughout page
  - Trace.Write("MyApp", "Button Clicked");
  - Trace.Write("MyApp", "Value: " + value);
- Access page from browser

Application Level Tracing

To enable tracing across multiple pages:
Create web.config file in application root

```
<configuration>
<trace enabled="true" requestlimit="10"/>
</configuration>
```

Hit one or more pages in the application
Access tracing URL for the application

```
http://localhost/MyApp/Trace.axd
```

Tracing into a Component

To add tracing to a component:

Import the Web namespace: using System.Web;

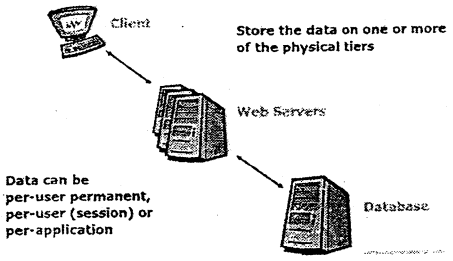
Enable tracing in your class constructor (optional): HttpContext.Current.Trace.IsEnabled = true;

Write to trace: HttpContext.Current.Trace.Write("category","msg");

State Management The Problem

How/where to store data?
How can you pass data from page to page?
How do we get around HTTP statelessness?

State Management Three-Tier Architecture



State Management Client

Client-side state management means this:
Client requests an initial page
The server generates a HTTP/HTML response that is sent back to the client
This response includes data (state)
User looks at the response and makes a selection, causing another request to the server
This second request contains the data that was sent in the first response
The server receives and processes the data
Could be same server or different server

URL in a hyperlink (<a>)
Query string
Very visible to users
This can be good or bad
Hidden form elements
Like __VIEWSTATE
Cookies
Limited to 4K
May be blocked by users

State Management Web Server Middle-Tier

Application variables
Shared by all sessions (users)
Session variables
Still need to pass session id through the client
ASP.NET State Service or database
Caching
Similar to application variables, but can be updated periodically or based upon dependencies

State Management Database

Application-level
Part of application database design
Session-level
Custom session state management in database
ASP.NET database session state support

State Management in ASP.NET
ASP.NET supports both Application-level and Session-level state management
Allows you to store state (data) in middle tier

State Management Application Variables
Application state is stored in an instance of HttpApplicationState
Accessed from Page.Application property
Can lock Application object for concurrent use
Needed only when changing application variable
Again, use this wisely
Use in "read-mostly" style
Initialize in global.asax
Avoid serializing your pages

State Management Sessions

What is a session?
Context in which a user communicates with a server over multiple HTTP requests
Within the scope of an ASP.NET Application
HTTP is a stateless, sessionless protocol
ASP.NET adds the concept of "session"
Session identifier: 120 bit ASCII string
Session events: Session_OnStart, Session_OnEnd
Session variables: store data across multiple requests
ASP.NET improves upon ASP sessions

State Management Session Identifier
By default, session id stored in a cookie
Can optionally track session id in URL
New in ASP.NET
Requires no code changes to app
All relative links continue to work

```
<configuration>
        <sessionState cookieless="true"/>
</configuration>
```

State Management Session Variables
ASP stores session state in IIS process
State is lost if IIS crashes
Can't use session state across machines
ASP.NET stores session state:

Prasanna wadekar

In process
In another process: ASP State NT service
In SQL Server database

```
<sessionstate inproc="false"
      server="AnotherServer" port="42424" />

<sessionstate inproc="false"
      server="AnotherServer" port="42424"
           usesqlserver="true" />
```

"Live" objects are are not stored in session state
Instead, ASP.NET serializes objects out between requests
ASP.NET approach provides:
Ability to recover from application crashes
Ability to recover from IIS crash/restart
Can partition an application across multiple machines (called a Web Farm)
Can partition an application across multiple processes (called a Web Garden)

Transferring Controls between Pages
Link to a page
Postback
Response.Redirect
Causes HTTP redirect
Tells browser to go to another URL
Server.Transfer
Like a redirect but entirely on one server
Server.Execute
Execute one page from another then return control
Both pages processed on same server

Caching

Many sites spend considerable effort generating the same web pages over and over
For example, a product catalog changes overnight, but is accessed tens of thousands of times a day
Server-side caching can vastly improve performance and scalability
ASP.NET provides support for
Page output caching
Data caching

Page out Caching

Entire web page output (HTML) is cached
Must specify life of cached page (in seconds)
Can cache multiple versions of a page, by:
GET/POST parameters; use VaryByParam
HTTP header; use VaryByHeader
E.g. Accept-Language
Browser type or custom string; use VaryByCustom

Partial Page out Caching

Can cache a portion of a page by placing it in a User Control
Can cache multiple versions of a User Control on a page, by property; use VaryByControl

Caching in the Browser

Don't confuse server-side page output caching with how the browser and proxy servers cache the page
Use Response.Cache to specify HTTP cache headers
Contains a HttpCachePolicy object

Data Caching

Data cache is similar to application variables
Can cache objects, HTML fragments, etc.
Usage pattern:
Try to retrieve data
If null then create data and insert into cache

```
DataView Source = (DataView)Cache["MyData"];
if (Source == null) {
  Source = new DataView(ds.Tables["Authors"]);
  Cache["MyData"] = Source;        // Save in cache
}
```

Cache object is stored on the Page and is an instance of System.Web.Caching.Cache
Cache may be scavenged: when memory runs low it will be automatically reclaimed
Can specify data expiration: absolute time (e.g. midnight), relative (in 1 hour)
Cached data can be dependent upon a file or other cache item

```
Cache.Insert("MyData", Source, null, // Expire in 1 hour
  DateTime.Now.AddHours(1), TimeSpan.Zero);
Cache.Insert("MyData", Source,       // Dependent on file
  new CacheDependency(Server.MapPath("authors.xml")));
```

Populating a data cache has an inherent race condition: if hit almost concurrently, multiple pages may try to populate the same cache

This probably doesn't matter at all; it's only significant if the cost to create the data is prohibitive or if there are side effects

If it does matter, there are two solutions:

- Populate the cache in Application_OnStart
- Synchronize access to the cache

```
private static String cacheSynchronize = "tflKey";;
DataView Source = (DataView)Cache["tflDataSet"];
if (Source == null) {
  lock (cacheSynchronize) {
    Source = (DataView)Cache["tflDataSet"];
    if (Source == null) {        // Have to test again
      // Open database ...
      Source = new DataView(ds.Tables["Trainers"]);
      Cache["MyDataSet"] = Source;    // Save in cache
    }
  }
}
```

ASP.NET page state maintenance is great, but __VIEWSTATE can get quite large

Why store constant data in __VIEWSTATE?

E.g. dropdowns listing countries, states, days of the week, months, product categories, SKUs, etc.

Instead, set EnableViewState=false, cache that data on the server, and populate the control from the cache in Page_Load

- Can cache data or even HTML
  - Use Control.Render() to obtain a control's HTML

Error Handling

.NET Common Language Runtime provides a unified exception architecture

- Runtime errors done using exceptions
- VB now supports try/catch/finally

ASP.NET also provides declarative application custom error handling

- Automatically redirect users to error page when unhandled exceptions occur
- Prevents ugly error messages from being sent to users

Error Handling Custom Error pages

- Can specify error pages for specific HTTP status codes in web.config

```
<configuration>
  <customerrors mode="remoteonly"
                defaultredirect="error.htm">
    <error statuscode="404"
           redirect="adminmessage.htm"/>
    <error statuscode="403"
           redirect="noaccessallowed.htm"/>
  </customerrors>
</configuration>
```

Error Handling Error Events

Can override Page.HandleError to deal with any unhandled exception on that page
Global application event raised if unhandled exception occurs
- Provides access to current Request
- Provides access to Exception object
- See HttpApplication.Error event

What do you actually do when an error occurs?
- Use new EventLog class to write custom events to log when errors occur
- Use new SmtpMail class to send email to administrators

Error Handling writing to Event Log

```
<%@ Import Namespace="System.Diagnostics" %>
<%@ Assembly name="System.Diagnostics" %>
<script language="C#" runat=server>
public void Application_Error(object Sender, EventArgs E) {
  string LogName = "MyCustomAppLog";
  string Message = "Url " + Request.Path + " Error: " +
                   this.Error.ToString()
  // Create event log if it doesn't exist
  if (! EventLog.SourceExists(LogName)) {
    EventLog.CreateEventSource(LogName, LogName);
  }
  // Fire off to event log
  EventLog Log = new EventLog();
  Log.Source = LogName;
  Log.WriteEntry(Message, EventLogEntryType.Error);
}
</script>
```

Error Handling Sending SMTP Mail

```
<%@ Import Namespace="System.Web.Util" %>
<%@ Assembly name="System.Diagnostics" %>
<script language="C#" runat=server>
public void Application_Error(object Sender, EventArgs E) {
  MailMessage MyMessage = new MailMessage();
  MyMessage.To = "ravi.tambade@transflower.in";
  MyMessage.From = "TFLAppServer";
  MyMessage.Subject = "Unhandled Error!!!";
  MyMessage.BodyFormat = MailFormat.Html;
  MyMessage.Body = "<html><body><h1>" + Request.Path +
    "</h1>" + Me.Error.ToString() + "</body></html>";
  SmtpMail.Send(MyMessage);
}
</script>
```

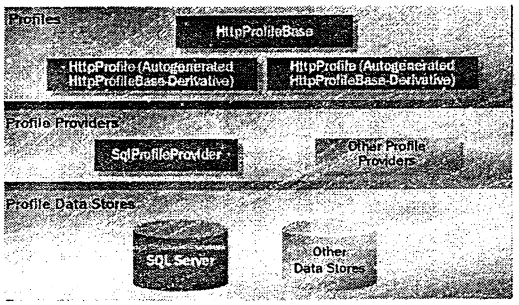
Personalization and Localization is ASP.NET

Profile Service

Stores per-user data persistently
- Strongly typed access (unlike session state)
- On-demand lookup (unlike session state)
- Long-lived (unlike session state)
- Supports authenticated and anonymous users

Accessed through dynamically compiled HttpProfileBase derivatives (HttpProfile)
Provider-based for flexible data storage

Profile Schema



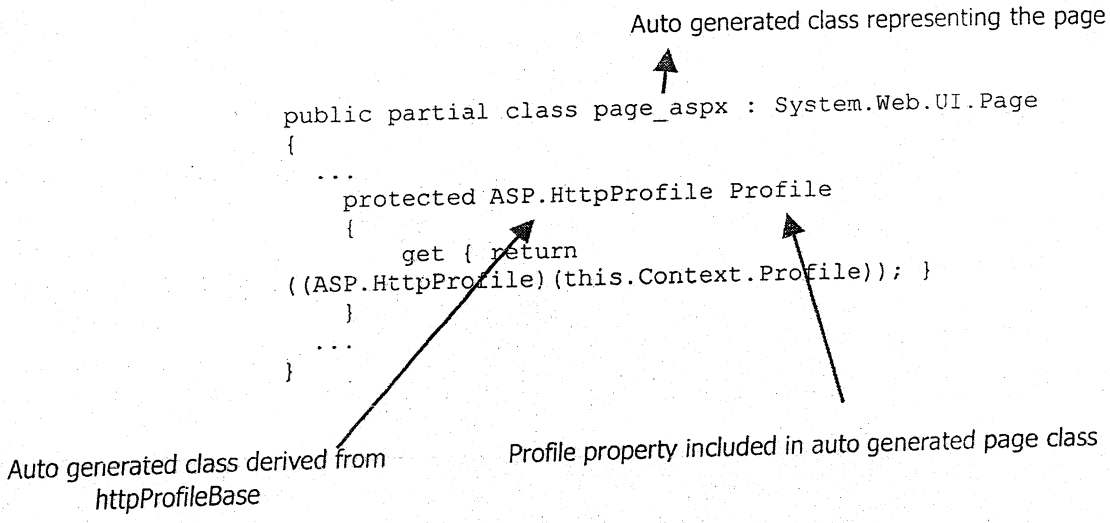
Defining a Profile

```
<configuration>
  <system.web>
    <profile>
      <properties>
        <add name="ScreenName" />
        <add name="Posts" type="System.Int32" defaultValue="0" />
        <add name="LastPost" type="System.DateTime" />
      </properties>
    </profile>
  </system.web>
</configuration>
```

Using a Profile

```
// Increment the current user's post count
Profile.Posts = Profile.Posts + 1;
// Update the current user's last post date
Profile.LastPost = DateTime.Now;
```

How Profile Works



Accessing another User's Profile

Profile.propertyname refers to current user

Use Profile.GetProfile (username) to access profiles for other users

```
// Get a reference to Ravi's profile
HttpProfile profile = Profile.GetProfile ("Ravi");
// Increment Fred's post count
profile.Posts = profile.Posts + 1;
// Update Ravi's last post date
profile.LastPost = DateTime.Now;
```

Accessing a profile from External Component

"Profile" property is only valid in classes generated by ASP.NET (ASPX, ASAX, etc.)

Use HttpContext.Profile property to access profiles from external components

```
// Read the current user's ScreenName property in an ASPX file
string name = Profile.ScreenName;
// Read the current user's ScreenName property in an external component
string name = (string) HttpContext.Current.Profile["ScreenName"];
```

Profile Groups

Properties can be grouped

<group> element defines groups

```
<profile>
  <properties>
    <add ... />
    <group name="...">
        <add ... />
    </group>
  </properties>
</profile>
```

Defining a Profile Group

```
<configuration>
  <system.web>
    <profile>
      <properties>
        <add name="ScreenName" />
        <group name="Forums">
          <add name="Posts" type="System.Int32" defaultValue="0" />
          <add name="LastPost" type="System.DateTime" />
        </group>
      </properties>
    </profile>
  </system.web>
</configuration>
```

Accessing a Profile Group

```
// Increment the current user's post count
Profile.Forums.Posts = Profile.Forums.Posts + 1;

// Update the current user's last post date
Profile.Forums.LastPost = DateTime.Now;
```

Custom DataType

Profiles support base types
- String, Int32, Int64, DateTime, Decimal, etc.

Profiles also support custom types
- Use type attribute to specify type
- Use serializeAs attribute to specify serialization mode: Binary, Xml (default), or String

serializeAs="Binary" types must be serializable
serializeAs="String" types need type converters

Using a Custom DataType

```
<configuration>
  <system.web>
    <profile>
      <properties>
        <add name="Cart" type="ShoppingCart" serializeAs="Binary" />
      </properties>
    </profile>
  </system.web>
</configuration>
```

Anonymous User Profiles

By default, profiles aren't available for anonymous (unauthenticated) users
- Data keyed by authenticated user IDs

Anonymous profiles can be enabled
- Step 1: Enable anonymous identification
- Step 2: Specify which profile properties are available to anonymous users

Data keyed by user anonymous IDs

Profiles for Anonymous Users

```
<configuration>
  <system.web>
    <anonymousIdentification enabled="true" />
    <profile>
      <properties>
        <add name="ScreenName" allowAnonymous="true" />
        <add name="Posts" type="System.Int32" defaultValue="0 />
        <add name="LastPost" type="System.DateTime" />
      </properties>
    </profile>
  </system.web>
</configuration>
```

Anonymous Identification

Anonymous identification can be cookied or cookieless (URL munging)
Cookies can be configured

```
<anonymousIdentification
      enabled="[true|false]"
      cookieName=".ASPXANONYMOUS"
       cookieTimeout="100000"
      cookiePath="/"
      cookieRequireSSL="[true|false]"
       cookieSlidingExpiration="[true|false]"
       cookieProtection="[None|Validation|Encryption|All]"
      cookieless="[UseCookies|UseUri|AutoDetect|UseDeviceProfile]"
/>
```

Profile Events

Profile service and anonymous identification service fire global events

Global.asax Handler Name	Description
Anonymousidentification_Create	Called when anonymous ID is issued
Anonymousidentification_Remove	Called when authentication request contains anonymous ID and ID is therefore deleted
Profile_MigrateAnonymous	Called other anonymousidentification_Remove to migrate settings for anonymous users

Profile Providers

Profile service is provider-based
VS 2005 ships with SqlProfileProvider (SQL Server)
Use custom providers to add support for other data stores

Localization

@ Page (UI)Culture "auto" keyword
Declarative mapping of Accept-Language headers to relevant thread properties
Simplified resource handling
Declarative mapping of control properties to resources using <%$ ... %> expressions
Strongly typed programmatic resource loading
<localize runat="server"> and more

Culture Handling (ASP.NET 1.x)

Code required to map Accept-Language headers to Current (UI) Culture properties of current thread

Global.asax

```
<script language="C#" runat="server">
void Application_BeginRequest (Object sender, EventArgs e)
{
    Thread.CurrentThread.CurrentCulture =
        CultureInfo.CreateSpecificCulture (Request.UserLanguages[0]);
    Thread.CurrentThread.CurrentUICulture =
        Thread.CurrentThread.CurrentCulture;
}
</script>
```

Culture Handling (ASP.NET)

"auto" keyword maps Accept-Language headers to Current(UI)Culture properties of current thread
Single Page (ASPX)

```
<%@ Page Culture="auto" UICulture="auto" %>
```

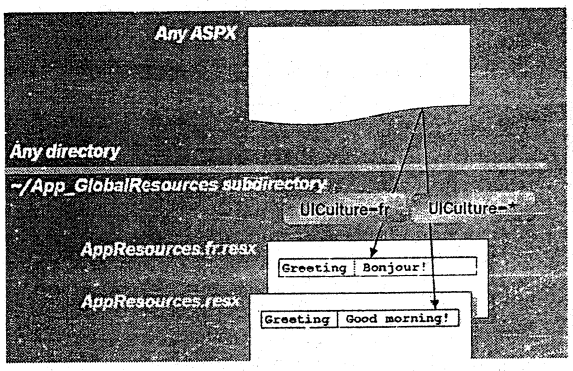
All pages (Web.config)

```
<globalization culture="auto" uiCulture="auto" />
```

Localization Resources

Application resources
Available to all pages in application
RESX files in ~/App_GlobalResources subdirectory
Local resources
Available to a specific page
RESX files in App_LocalResources subdirectories
Use filename.culture.resx naming schema for localization based on CurrentUICulture

Application Resources



Loading Application Resources

Declarative

```
<asp:Label ID="Output" RunAt="server"
        Text='<%$ Resources:AppResources, Greeting %>' />
```

(RESX Name: AppResources) (Resource Name: Greeting)

Programmatic

```
// Strong typing
Output.Text = Resources.AppResources.Greeting;
// Weak typing
Output.Text = (string) GetAppResourceObject ("AppResources", "Greeting");
```

Loading Local Resources

Declarative

```
<asp:Label ID="Output" RunAt="server" Text='<%$ Resources:Greeting%>' />
```

Resource Name

Programmatic

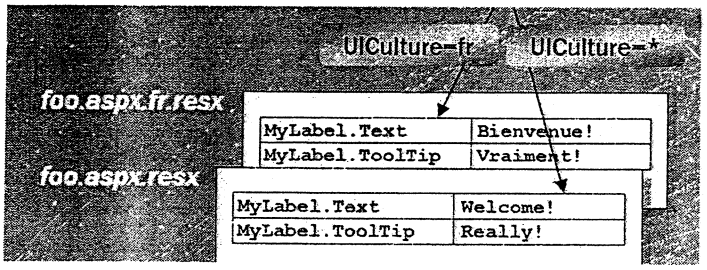
```
// Weak typing only
Output.Text = (string) GetPageResourceObject ("Greeting");
```

Implicit Expressions

Batch-initialize control properties
Applicable to local resources only

foo.aspx

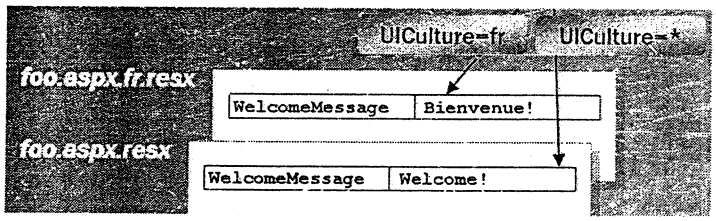
```
<asp:Label ID="Output" RunAt="server" meta:ResourceKey="MyLabel" />
```



The Localize Control

Loads static content from resources based on Thread.CurrentThread.CurrentUICulture
foo.aspx

```
<asp:Localize Runat="server" Text="<%$ Resources:WelcomeMessage %>" />
```



Panel. Direction

Direction="RightToLeft" right-justifies content generated by child controls
Use it to right-justify right-to-left text
Supported at design time by VS IDE

```
<asp:Panel Direction="RightToLeft" RunAt="server">
     Imagine this were in Hebrew<br />
   <asp:Login RunAt="server" />
</asp:Panel>
```

Deployment

- XCOPY deployment
  - Components are placed in .\bin folder
- No DLL deployment, registration
  - Unless you're using COM or other DLLs
- No locked DLLs
  - DLLs are "shadow copied" into a hidden folder
  - .aspx files are automatically compiled
  - Not true for codebehind
- Update code (.aspx and assemblies) while server is running
- No need to stop/bounce the server

- Applications are isolated
  - Each can have their own version of components
- Uninstall = delete /s *.*

Security

- Reasons for Security
  - Prevent access to areas of your Web server
  - Record and store secure relevant user data
- Security Configuration
  - <security> tag in web.config file
- Authentication, Authorization, Impersonation
- Code Access Security
  - Are you the code you told me you are?
  - Protect your server from bad code

Security Authentication

Who are you?
Server must authenticate client
Client should authenticate server
Kerberos does
Need a directory to store user accounts
NT: Security Accounts Manager
Good for intranet usage up to about 40,000 accounts
Windows Server: Active Directory
Good for intranet and Internet usage

Security IIS Authentication

Anonymous
A single Windows Server account is used for all visitors
Basic authentication
Standard, commonly supported
Password sent in clear text
Digest authentication
Standard, but not yet common
Integrated Windows Authentication
NTLM
Kerberos (Windows 2000 only)
Client certificates
Mapped to W2K/NT account

Security ASP.NET Authentication

Passport module provided
Exposes passport profile API
Custom, forms-based authentication
Easy to use, with cookie token tracking
Enables custom login screen (no popup dialogs)
Supports custom credential checks against database, exchange, etc.

Security Authorization

Now that I know who you are, here's what you are allowed to do
W2K/NT DACLs (Discretionary Access-Control List)
Grant and deny read/write/execute/etc. permission to users or groups of users
IIS also provides coarse-grained control
Read, write, run script, run executable, directory browsing, script access for virtual directories, directories and files

Security ASP.NET Authorization

ASP.NET supports authorization using either users or roles
Roles map users into logical groups
Example: "User", "Manager", "VP", etc.
Provides nice developer/admin separation
Developers can perform runtime role checks in code
if (User.IsInRole("Admin") { }

Security Impersonation

IIS authenticates the "user"
A token is passed to the ASP.NET application
ASP.NET impersonates the given token
Access is permitted according to NTFS settings

Developing Web Forms Debugging

Basic Debugger ships with the SDK
Multi-language
Single stack trace
Breaks, watches, etc.
Visual Studio® Debugger
Adds remote debugging support
Supports managed/unmanaged debugging

1. Create web.config file in application root

```
<configuration>
<compilation debugmode="true"/>
</configuration>
```

2. Attach using debugger
3. Set breakpoints
4. Hit page or service

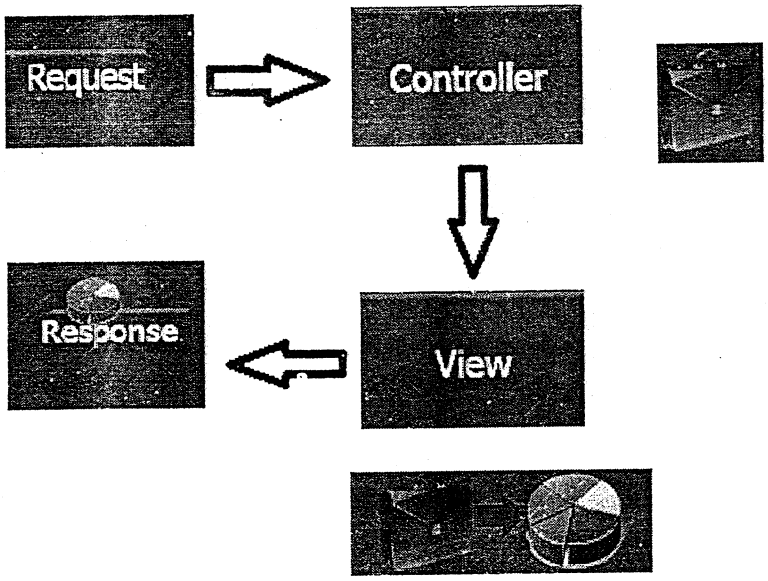
ASP.NET MVC

ASP.NET
: One Web application framework to rule them all...

Caching	Modules	Globalization
Pages	Controls	Master Pages
Profile	Roles	Membership
Handlers	Etc.	

ASP.NET MVC
Separation of concerns
- **Model**
- **View**
- **Controller**

How does MVC Look?



There are two major considerations for choosing between Webforms and ASP.NET MVC:

- **Test Driven Development** – life would be MUCH easier using MVC while following TDD.
- **Data Driven Application** – life would be MUCH easier using WebForms if the application is data heavy.

There are no rights or wrongs, and every application can be written in both frameworks. In fact, you can even have a hybrid approach, where you can write some part in WebForms and some in MVC.

SWOT Analysis Webforms and ASP.NET MVC

Asp.Net WebForms

Strengths

- Provides very good RAD development capabilities.
- Great designer support in Visual Studio.
- Ease of development for data-heavy LOB applications.
- Very rich control libraries and third party vendor support.
- A familiar event-driven model when compared to Windows Forms development, and so easy for developers to pick up.

Weaknesses

- UI logic coupled with the code, and thus is hard to separate.
- Harder to unit test, so difficult to employ TDD.
- Heavy page sizes due to view state management.

Opportunities

- Great at creating quick prototypes for business applications. This comes in very handy when trying to show quick Proof of Concepts to clients.

Threats

- Harder to adopt to various UI views despite the various frameworks available (master pages, Themes, etc.).

WebForms is great, but options are good.....

Classic ASP.NET (Web Forms)

High level abstraction over HTML / HTTP

Simplified state management

ViewState and the post-back model

Control model

Data binding

Simple event-driven mechanism

Simple Page Controller pattern

What makes it lazy ..... web forms

Sub-optimal URLs

blog.aspx?date=21032008

Form runat="server"

ViewState

Hard to test

All sorts of code in the page

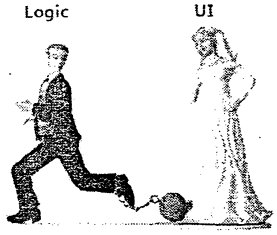
Requirement to test with an HttpContext

No real role responsibility...

Pages, Master pages, UI logic, Business logic, Data Access everything together with the help of powerful Webserver controls.

Control abstractions can be negative...

It isn't easy enough to test...



Asp.Net MVC

Strengths

- Provides fine control over rendered HTML.
- Cleaner generated HTML.
- Superior separation between UI and code.
- Easier to unit test.
- Can support multiple view engines.
- By default uses RESTful interfaces for URLs – so better SEO.
- No ViewState (this may also be a weakness).
- Typical size of page is small.
- Easy integration with frameworks like JQuery.

Weaknesses

- Not event driven, so maybe difficult for people who know only Asp.Net Webforms to wrap their minds around it.
- Third party control library support is not that strong.
- No ViewState(this is also a strength).

Opportunities

- Allows for Test Driven Development (TDD) – it is built with TDD in mind, so its much easier to write unit test cases, mock objects, and to intercept the program flow.
- Allows for reuse of the same models to present different UIs and Interfaces.

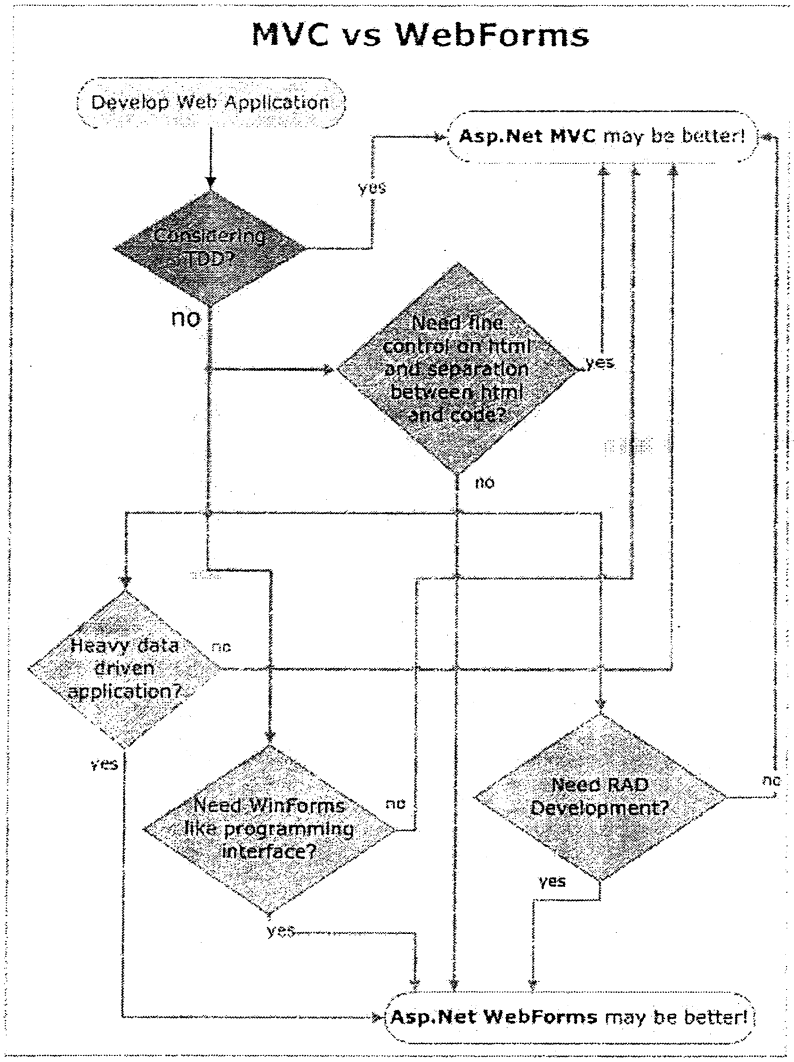
Threats

- Bigger ramp-up and training time required for developers with no or little experience in web application development.

ASP.NET MVC Tenets

Alternative
Testable
Extensible
Routable

Choosing between ASP.NET Webforms and ASP.NET MVC



ASP.NET MVC doesn't have...

Postbacks
View state
Control state
Server-side form
Page/Control lifecycle

ASP.NET MVC still has...

Web designer
Master pages & User controls
Membership/Roles/Profile
Globalization & Caching
HTTP intrinsics:
- HttpContext
- HttpRequest
- HttpResponse
- Etc.

Routing

Routing provides "clean" URLs
URL is mapped to a route handler
Extra level of indirection
Handlers can be changed without impacting URL
URL can be changed without impacting handler
Enables support for multilingual URLs
URL Example: http://www.mysite.com/Home/ProductList

Developers adds Routes to a global RouteTable
Mapping creates a RouteData - a bag of key/values

```
public static void RegisterRoutes(RouteCollection routes)
{
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
routes.MapRoute( "Default",
                 "{controller}/{action}/{id}",
            new {  controller = "Home", // Parameter defaults
            action = "Index",
                  action = "Index",  id = ""
}
};
}
protected void Application_Start()
{
      RegisterRoutes(RouteTable.Routes);
}
```

MVC Controllers

Scenarios, Goals and Design
URLs route to controller "actions",
not pages – mark actions in Controller.
Controller executes logic, chooses view.
All public methods are accessible

```
public void ShowPost(int id) {
   Post p = PostRepository.GetPostById(id);
   if (p != null) {
       RenderView("showpost", p);
   } else {
       RenderView("nosuchpost", id);
   }
}
```

Controller Conventions

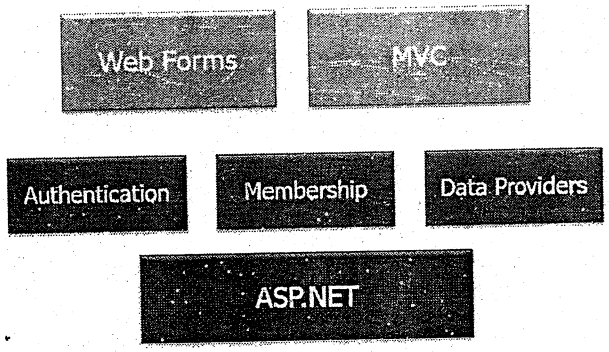
Controller
Must...
Be suffixed with "Controller"
Implement IController (or inherit from Controller)
Action
Must...
Be Public
Return ActionResult or void
Can't...
Be generic
Have a NonActionAttribute
Have out/ref parameters

Views

- Are for rendering/output.
- Pre-defined and extensible rendering helpers
- Can use .ASPX, .ASCX, .MASTER, etc.
- Can replace with other view technologies
- Template engines (NVelocity, Brail, ...).
- Output formats (images, RSS, JSON, ...).
- Mock out for testing.
- Controller sets data on the View
- Loosely typed or strongly typed data

Clean URLs
Don't settle for...
/Products.aspx?CategoryID=123
When you can easily have...
/Product/Puppies

Visual Studio



ASP.NET MVC Features

- Razor View Engine
- Multiple View Engine Support
- Validation Improvements
- Dynamic ViewBag
- Global Filters
- Action Results
- Taskbased Helpers
- Dependency Injection
- Porting MVC Libraries to jQuery
- Granular Validation Input
- Caching Support
- JSON Binding Support

```
protected void Application_Start()
{
    ViewEngines.Engines.Add(new SparkViewFactory());
    ...
}
```

Razor

The Razor View Engine
Razor syntax is clean and concise, requiring a minimum number of keystrokes.
Razor is easy to learn, in part because it's based on existing languages like C# and Visual Basic.
Visual Studio includes IntelliSense and code colorization for Razor syntax.
Razor views can be unit tested without requiring that you run the application or launch a web server.

Syntax Comparison
Using traditional asp.net code

```
<h1>Code Nugget Example with .ASPX</h1>
```

```
<h3>Hello <%=name %>, the year is <%= DateTime.Now.Year %></h3>
<p>Checkout <a href="/Products/Details/<%=productId %">this product</a>
</p>
```

Using Razor

```
<h1>Razor Example</h1>
<h3>
     Hello @name, the year is @DateTime.Now.Year
</h3>
<p>
     Checkout <a href="="/Products/Details/@productId">this product</a>
</p>
```

Filters

Filters are custom classes that provide both a declarative and programmatic means to add pre-action and post-action behavior to controller action methods

- Authorization filters
- Action filters
- Result filters
- Exception filters

Dynamic View Bag

MVC 2 controllers support a ViewData property that enables you to pass data to a view template using a late-bound dictionary API.

In MVC 3, you can also use somewhat simpler syntax with the ViewBag property to accomplish the same purpose.

For example, instead of writing ViewData["Message"]="text", you can write ViewBag.Message="text".

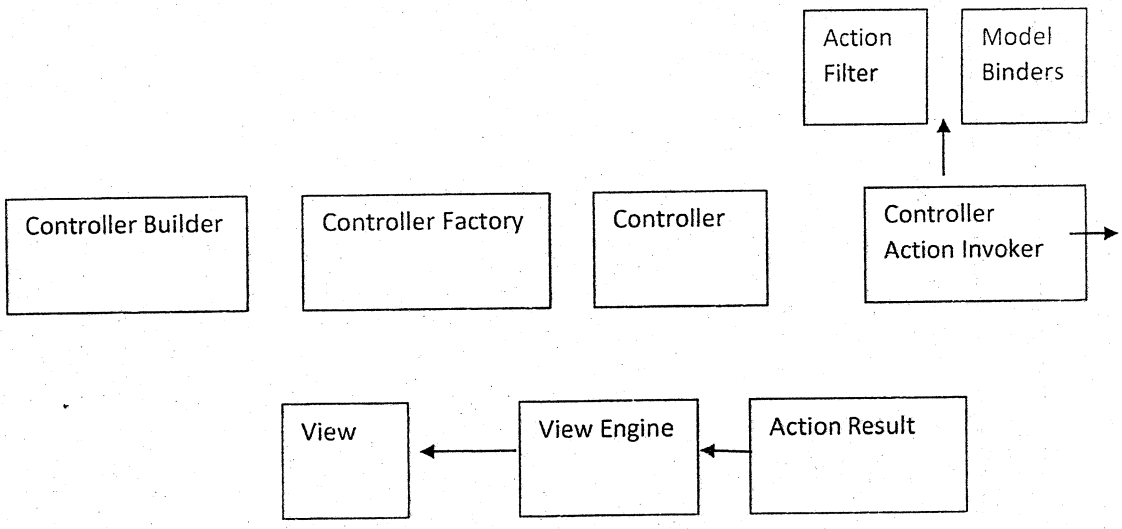
You do not need to define any strongly-typed classes to use the ViewBagproperty.

Dynamic property, you can instead just get or set properties and it will resolve them dynamically at run time.

```
<head>
<title>@ViewBag.Title<title>
</head>
```

Extensibility

Any of these can be replaces.



Dependency Injection Improvements

ASP.NET MVC provides better support for applying Dependency Injection (DI) and for integrating with Dependency Injection or Inversion of Control (IOC) containers. Support for DI has been added in the following areas:

Controllers (registering and injecting controller factories, injecting controllers).
Views (registering and injecting view engines, injecting dependencies into view pages).
Action filters (locating and injecting filters).
Model binders (registering and injecting).
Model validation providers (registering and injecting).
Model metadata providers (registering and injecting).
Value providers (registering and injecting).

Model Binding

Model binding is the process of creating .NET objects using the data sent by the browser in an HTTP request.
Default model binder search in following location and order for named parameters data.

Name	Description
Request.Form	Content HTML form element data
RouteData.Values	Application routes values
Request.QueryString	Data in the query string of the request URL
Request.Files	Files that have been uploaded as part of the request

Bind attribute: To include or exclude model properties from the binding process.
To include first and last name in person object. Person FirstName and LastName property value will be consider other will ignored.

```
public ActionResult Register([Bind(Include = "FirstName, LastName")]
                                    Person  person)
```

Example: To exclude sex property in person object.
Person sex property value will be ignored.

```
public ActionResult Register([Bind(Exclude = "Sex")] Person person)
```

MVC Interview Questions/Answers

What are the 3 main components of an ASP.NET MVC application?
1. M - Model
2. V - View
3. C - Controller

In which assembly is the MVC framework defined?
System.Web.Mvc

Is it possible to combine ASP.NET webforms and ASP.MVC and develop a single web application?
Yes, it is possible to combine ASP.NET webforms and ASP.MVC and develop a single web application.

What does Model, View and Controller represent in an MVC application?
Model: Model represents the application data domain. In short the applications business logic is contained with in the model.

View: Views represent the user interface, with which the end users interact. In short the all the user interface logic is contained with in the UI.

Controller: Controller is the component that responds to user actions. Based on the user actions, the respective controller, work with the model, and selects a view to render that displays the user interface. The user input logic is contained with in the controller.

What is the greatest advantage of using asp.net mvc over asp.net webforms?
It is difficult to unit test UI with webforms, where views in mvc can be very easily unit tested.

Which approach provides better support for test driven development - ASP.NET MVC or ASP.NET Webforms?
ASP.NET MVC

What are the advantages of ASP.NET MVC?
1. Extensive support for TDD. With asp.net MVC, views can also be very easily unit tested.
2. Complex applications can be easily managed
3. Seperation of concerns. Different aspects of the application can be divided into Model, View and Controller.
4. ASP.NET MVC views are light weight, as they donot use viewstate.

Is it possible to unit test an MVC application without running the controllers in an ASP.NET process?
Yes, all the features in an asp.net MVC application are interface based and hence mocking is much easier. So, we don't have to run the controllers in an ASP.NET process for unit testing.

Is it possible to share a view across multiple controllers?
Yes, put the view into the shared folder. This will automatically make the view available across multiple controllers.

What is the role of a controller in an MVC application?
The controller responds to user interactions, with the application, by selecting the action method to execute and also selecting the view to render.

Where are the routing rules defined in an asp.net MVC application?
In Application_Start event in Global.asax

Name a few different return types of a controller action method?
The following are just a few return types of a controller action method. In general an action method can return an instance of a any class that derives from ActionResult class.
1. ViewResult
2. JavaScriptResult
3. RedirectResult
4. ContentResult
5. JsonResult

What is the significance of NonActionAttribute?
In general, all public methods of a controller class are treated as action methods. If you want prevent this default behaviour, just decorate the public method with NonActionAttribute.

What is the significance of ASP.NET routing?
ASP.NET MVC uses ASP.NET routing, to map incoming browser requests to controller action methods. ASP.NET Routing makes use of route table. Route table is created when your web application first starts. The route table is present in the Global.asax file.

What are the 3 segments of the default route, that is present in an ASP.NET MVC application?
1st Segment - Controller Name
2nd Segment - Action Method Name
3rd Segment - Parameter that is passed to the action method

Example: http://pragimtech.com/Customer/Details/5
Controller Name = Customer
Action Method Name = Details
Parameter Id = 5

ASP.NET MVC application, makes use of settings at 2 places for routing to work correctly. What are these 2 places?
1. Web.Config File : ASP.NET routing has to be enabled here.
2. Global.asax File : The Route table is created in the application Start event handler, of the Global.asax file.

What is the adavantage of using ASP.NET routing?
In an ASP.NET web application that does not make use of routing, an incoming browser request should map to a physical file. If the file does not exist, we get page not found error.

An ASP.NET web application that does make use of routing, makes use of URLs that do not have to map to specific files in a Web site. Because the URL does not have to map to a file, you can use URLs that are descriptive of the user's action and therefore are more easily understood by users.

What are the 3 things that are needed to specify a route?
1. URL Pattern - You can include placeholders in a URL pattern so that variable data can be passed to the request handler without requiring a query string.
2. Handler - The handler can be a physical file such as an .aspx file or a controller class.
3. Name for the Route - Name is optional.

Is the following route definition a valid route definition?
{controller}{action}/{id}

No, the above definition is not a valid route definition, because there is no literal value or delimiter between the placeholders. Therefore, routing cannot determine where to separate the value for the controller placeholder from the value for the action placeholder.

What is the use of the following default route?
{resource}.axd/{*pathInfo}
This route definition, prevent requests for the Web resource files such as WebResource.axd or ScriptResource.axd from being passed to a controller.

What is the difference between adding routes, to a webforms application and to an mvc application?
To add routes to a webforms application, we use MapPageRoute() method of the RouteCollection class, where as to add routes to an MVC application we use MapRoute() method.

How do you handle variable number of segments in a route definition?
Use a route with a catch-all parameter. An example is shown below. * is referred to as catch-all parameter.
controller/{action}/{*parametervalues}

What are the 2 ways of adding constraints to a route?
1. Use regular expressions
2. Use an object that implements IRouteConstraint interface

Give 2 examples for scenarios when routing is not applied?
1. A Physical File is Found that Matches the URL Pattern - This default behaviour can be overriden by setting the RouteExistingFiles property of the RouteCollection object to true.
2. Routing Is Explicitly Disabled for a URL Pattern - Use the RouteCollection.Ignore() method to prevent routing from handling certain requests.

What is the use of action filters in an MVC application?
Action Filters allow us to add pre-action and post-action behavior to controller action methods.

If I have multiple filters impleted, what is the order in which these filters get executed?
1. Authorization filters
2. Action filters
3. Response filters
4. Exception filters

What are the different types of filters, in an asp.net mvc application?
1. Authorization filters
2. Action filters
3. Result filters
4. Exception filters

Give an example for Authorization filters in an asp.net mvc application?
1. RequireHttpsAttribute
2. AuthorizeAttribute

Which filter executes first in an asp.net mvc application?
Authorization filter

What are the levels at which filters can be applied in an asp.net mvc application?
1. Action Method
2. Controller
3. Application

Is it possible to create a custom filter?
Yes

What filters are executed in the end?
Exception Filters

Is it possible to cancel filter execution?
Yes

What type of filter does OutputCacheAttribute class represents?
Result Filter

What are the 2 popular asp.net mvc view engines?
1. Razor
2. .aspx

What symbol would you use to denote, the start of a code block in razor views?
@

What symbol would you use to denote, the start of a code block in aspx views?
<%= %>

In razor syntax, what is the escape sequence character for @ symbol?
The escape sequence character for @ symbol, is another @ symbol

When using razor views, do you have to take any special steps to proctect your asp.net mvc application from cross site scripting (XSS) attacks?
No, by default content emitted using a @ block is automatically HTML encoded to protect from cross site scripting (XSS) attacks.

When using aspx view engine, to have a consistent look and feel, across all pages of the application, we can make use of asp.net master pages. What is asp.net master pages equivalent, when using razor views?
To have a consistent look and feel when using razor views, we can make use of layout pages. Layout pages, reside in the shared folder, and are named as _Layout.cshtml

What are sections?
Layout pages, can define sections, which can then be overriden by specific views making use of the layout. Defining and overriding sections is optional.

What are the file extensions for razor views?
1. .cshtml - If the programming lanugaue is C#
2. .vbhtml - If the programming lanugaue is VB

How do you specify comments using razor syntax?
Razor syntax makes use of @* to indicate the begining of a comment and *@ to indicate the end. An example is shown

below.

@* This is a Comment *@

LINQ Language Integrated Query

The C# 3.0 language enhancements build on C# 2.0 to increase developer productivity: they make written code more concise and make working with data as easy as working with objects. These features provide the foundation for the LINQ project, a general purpose declarative query facility that can be applied to in-memory collections and data stored in external sources such as XML files and relational databases.

The C# 3.0 language enhancements consist of:

- **Auto-implemented properties,** which automate the process of creating properties with trivial implementations.
- **Implicitly typed local variables,** which permit the type of local variables to be inferred from the expressions used to initialize them.
- **Implicitly typed arrays**, a form of array creation and initialization that infers the element type of the array from an array initializer.
- **Extension methods,** which make it possible to extend existing types and constructed types with additional methods.
- **Lambda expressions,** an evolution of anonymous methods that concisely improves type inference and conversions to both delegate types and expression trees.
- **Expression trees**, which permit lambda expressions to be represented as data (expression trees) instead of as code (delegates).
- **Object and collection initializers, which you can use** to conveniently specify values for one or more fields or properties for a newly created object, combining creation and initialization into a single step.
- **Query expressions**, which provide a language-integrated syntax for queries that is similar to relational and hierarchical query languages such as SQL and XQuery.
- **Anonymous types**, which are tuple types automatically inferred and created from object initializers.

Use of Automatically Implemented Properties
Easy Initialization with Object and Collection Initializers

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace NewLanguageFeatures
{
    public class Customer
    {
        public int CustomerID { get; private set; }
        public string Name { get; set; }
        public string City { get; set; }

        public Customer(int ID)
        {
            CustomerID = ID;
        }

        public override string ToString()
        {
            return Name + "\t" + City + "\t" + CustomerID;
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Customer c = new Customer(1);
            c.Name = "Maria Anders";
            c.City = "Berlin";

            Console.WriteLine(c);
        }

    }
}
```

Implicitly Typed Local Variables and Implicitly Typed Arrays

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace NewLanguageFeatures
{
    public class Customer
    {
        public int CustomerID { get; private set; }
        public string Name { get; set; }
        public string City { get; set; }

        public Customer(int ID)
        {
            CustomerID = ID;
        }

        public override string ToString()
        {
            return Name + "\t" + City + "\t" + CustomerID;
        }
    }

    class Program
    {

        static void Main(string[] args)
        {
            List<Customer> customers = CreateCustomers();

            Console.WriteLine("Customers:\n");
            foreach (Customer c in customers)
                Console.WriteLine(c);
        }

        static List<Customer> CreateCustomers()
        {
           return new List<Customer>
           {
        new Customer(1) { Name = "Maria Anders",      City = "Berlin"    },
        new Customer(2) { Name = "Laurence Lebihan", City = "Marseille" },
        new Customer(3) { Name = "Elizabeth Brown",  City = "London"    },
        new Customer(4) { Name = "Ann Devon",        City = "London"    },
        new Customer(5) { Name = "Paolo Accorti",    City = "Torino"    },
        new Customer(6) { Name = "Fran Wilson",      City = "Portland"  },
        new Customer(7) { Name = "Simon Crowther",   City = "London"    },
        new Customer(8) { Name = "Liz Nixon",        City = "Portland"  }
           };
        }
    }
}
```

Extending Types with Extension Methods

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace NewLanguageFeatures
{
    public class Customer
    {
        public int CustomerID { get; private set; }
        public string Name { get; set; }
        public string City { get; set; }

        public Customer(int ID)
        {
            CustomerID = ID;
        }

        public override string ToString()
        {
            return Name + "\t" + City + "\t" + CustomerID;
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            List<Customer> customers = CreateCustomers();

            Console.WriteLine("Customers:\n");
            foreach (Customer c in customers)
                Console.WriteLine(c);
        }

        static List<Customer> CreateCustomers()
        {
            return new List<Customer>
            {
            new Customer(1) { Name = "Maria Anders",      City = "Berlin"    },
            new Customer(2) { Name = "Laurence Lebihan", City = "Marseille" },
            new Customer(3) { Name = "Elizabeth Brown",  City = "London"    },
            new Customer(4) { Name = "Ann Devon",        City = "London"    },
            new Customer(5) { Name = "Paolo Accorti",    City = "Torino"    },
            new Customer(6) { Name = "Fran Wilson",      City = "Portland"  },
            new Customer(7) { Name = "Simon Crowther",   City = "London"    },
            new Customer(8) { Name = "Liz Nixon",        City = "Portland"  }
            };
        }
    }
}
```

Working with Lambda Expressions

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace NewLanguageFeatures
{
    public static class Extensions
    {
        public static List<T> Append<T>(this List<T> a, List<T> b)
        {
            var newList = new List<T>(a);
            newList.AddRange(b);
            return newList;
        }

        public static bool Compare(this Customer customer1, Customer customer2)
        {
            if (customer1.CustomerID == customer2.CustomerID &&
                customer1.Name == customer2.Name &&
                customer1.City == customer2.City)
            {
                return true;
            }
            return false;
        }
    }

    public class Customer
    {
        public int CustomerID { get; private set; }
        public string Name { get; set; }
        public string City { get; set; }

        public Customer(int ID)
        {
            CustomerID = ID;
        }

        public override string ToString()
        {
            return Name + "\t" + City + "\t" + CustomerID;
        }
    }
```

```
class Program
{
    static void Main(string[] args)
    {
        var customers = CreateCustomers();

        var addedCustomers = new List<Customer>
        {
            new Customer(9)   { Name = "Paolo Accorti", City = "Torino" },
            new Customer(10)  { Name = "Diego Roel", City = "Madrid" }
        };

        var updatedCustomers = customers.Append(addedCustomers);

        var newCustomer = new Customer(10)
        {
            Name = "Diego Roel",
            City = "Madrid"
        };

        foreach (var c in updatedCustomers)
        {
            if (newCustomer.Compare(c))
            {
             Console.WriteLine("The new customer was already in the list");
                return;
            }
        }
        Console.WriteLine("The new customer was not in the list");
    }

    static List<Customer> CreateCustomers()
    {
        return new List<Customer>
        {
        new Customer(1) { Name = "Maria Anders",      City = "Berlin"    },
        new Customer(2) { Name = "Laurence Lebihan", City = "Marseille" },
        new Customer(3) { Name = "Elizabeth Brown",  City = "London"    },
        new Customer(4) { Name = "Ann Devon",        City = "London"    },
        new Customer(5) { Name = "Paolo Accorti",    City = "Torino"    },
        new Customer(6) { Name = "Fran Wilson",      City = "Portland"  },
        new Customer(7) { Name = "Simon Crowther",   City = "London"    },
        new Customer(8) { Name = "Liz Nixon",        City = "Portland"  }
        };
    }
}
```

Using Lambda Expressions to Create Expression Trees

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace NewLanguageFeatures
{
    public static class Extensions
    {
        public static List<T> Append<T>(this List<T> a, List<T> b)
        {
            var newList = new List<T>(a);
            newList.AddRange(b);
            return newList;
        }

        public static bool Compare(this Customer customer1, Customer customer2)
        {
            if (customer1.CustomerID == customer2.CustomerID &&
                customer1.Name == customer2.Name &&
                customer1.City == customer2.City)
            {
                return true;
            }
            return false;
        }
    }

    public class Customer
    {
        public int CustomerID { get; private set; }
        public string Name { get; set; }
        public string City { get; set; }

        public Customer(int ID)
        {
            CustomerID = ID;
        }

        public override string ToString()
        {
            return Name + "\t" + City + "\t" + CustomerID;
        }
    }
```

```
class Program
    {
        static void Main(string[] args)
        {
            var customers = CreateCustomers();

            foreach (var c in FindCustomersByCity(customers, "London"))
                Console.WriteLine(c);
        }

        public static List<Customer> FindCustomersByCity(
            List<Customer> customers,
            string city)
        {
            return customers.FindAll(c => c.City == city);
        }

        static List<Customer> CreateCustomers()
        {
           return new List<Customer>
           {
            new Customer(1) { Name = "Maria Anders",      City = "Berlin"    },
            new Customer(2) { Name = "Laurence Lebihan", City = "Marseille" },
            new Customer(3) { Name = "Elizabeth Brown",   City = "London"    },
            new Customer(4) { Name = "Ann Devon",         City = "London"    },
            new Customer(5) { Name = "Paolo Accorti",     City = "Torino"    },
            new Customer(6) { Name = "Fran Wilson",       City = "Portland"  },
            new Customer(7) { Name = "Simon Crowther",    City = "London"    },
            new Customer(8) { Name = "Liz Nixon",         City = "Portland"  }
           };
        }
    }
}
```

Understanding Queries and Query Expressions

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Linq.Expressions;

namespace NewLanguageFeatures
{
    public static class Extensions
    {
        public static List<T> Append<T>(this List<T> a, List<T> b)
        {
            var newList = new List<T>(a);
            newList.AddRange(b);
            return newList;
        }

        public static bool Compare(this Customer customer1, Customer customer2)
        {
            if (customer1.CustomerID == customer2.CustomerID &&
                customer1.Name == customer2.Name &&
                customer1.City == customer2.City)
            {
                return true;
            }
            return false;
        }
    }

    public class Customer
    {
        public int CustomerID { get; private set; }
        public string Name { get; set; }
        public string City { get; set; }

        public Customer(int ID)
        {
            CustomerID = ID;
        }

        public override string ToString()
        {
            return Name + "\t" + City + "\t" + CustomerID;
        }
    }
```

LINQ Language Integrated Query

```
class Program
    {
        static void Main(string[] args)
        {
            Func<int, int> addOne = n => n + 1;
            Console.WriteLine("Result: {0}", addOne(5));

            Expression<Func<int, int>> addOneExpression = n => n + 1;

            var addOneFunc = addOneExpression.Compile();
            Console.WriteLine("Result: {0}", addOneFunc(5));
        }

        public static List<Customer> FindCustomersByCity(
            List<Customer> customers,
            string city)
        {
            return customers.FindAll(c => c.City == city);
        }

        static List<Customer> CreateCustomers()
        {
           return new List<Customer>
           {
            new Customer(1) { Name = "Maria Anders", City = "Berlin"},
            new Customer(2) { Name = "Laurence Lebihan", City = "Marseille" },
            new Customer(3) { Name = "Elizabeth Brown", City = "London"    },
            new Customer(4) { Name = "Ann Devon",       City = "London"    },
            new Customer(5) { Name = "Paolo Accorti",   City = "Torino"    },
            new Customer(6) { Name = "Fran Wilson",     City = "Portland"  },
            new Customer(7) { Name = "Simon Crowther",  City = "London"    },
            new Customer(8) { Name = "Liz Nixon",       City = "Portland"  }
           };
        }
    }
}
```

Anonymous Types and Advanced Query Creation

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Linq.Expressions;

namespace NewLanguageFeatures
{
    public static class Extensions
    {
        public static List<T> Append<T>(this List<T> a, List<T> b)
        {
            var newList = new List<T>(a);
            newList.AddRange(b);
            return newList;
        }

        public static bool Compare(this Customer customer1, Customer customer2)
        {
            if (customer1.CustomerID == customer2.CustomerID &&
                customer1.Name == customer2.Name &&
                customer1.City == customer2.City)
            {
                return true;
            }
            return false;
        }
    }

    public class Store
    {
        public string Name { get; set; }
        public string City { get; set; }

        public override string ToString()
        {
            return Name + "\t" + City;
        }
    }
    public class Customer
    {
        public int CustomerID { get; private set; }
        public string Name { get; set; }
        public string City { get; set; }

        public Customer(int ID)
        {
            CustomerID = ID;
        }

        public override string ToString()
        {
            return Name + "\t" + City + "\t" + CustomerID;
        }
    }
```

```
class Program
    {
        static void Query()
        {
            var stores = CreateStores();
            var numLondon = stores.Count(s => s.City == "London");
            Console.WriteLine("There are {0} stores in London. ", numLondon);
        }

        static void Main(string[] args)
        {
            Query();
        }

        public static List<Customer> FindCustomersByCity(
            List<Customer> customers,
            string city)
        {
            return customers.FindAll(c => c.City == city);
        }

        static List<Store> CreateStores()
        {
           return new List<Store>
           {
              new Store { Name = "Jim's Hardware",    City = "Berlin"     },
              new Store { Name = "John's Books",      City = "London"     },
              new Store { Name = "Lisa's Flowers",    City = "Torino"     },
              new Store { Name = "Dana's Hardware",   City = "London"     },
              new Store { Name = "Tim's Pets",        City = "Portland"   },
              new Store { Name = "Scott's Books",     City = "London"     },
              new Store { Name = "Paula's Cafe",      City = "Marseille"  }
           };
        }
        static List<Customer> CreateCustomers()
        {
           return new List<Customer>
           {
              new Customer(1) { Name = "Maria Anders", City = "Berlin"},
              new Customer(2) { Name = "Laurence Lebihan",City = "Marseille"},
              new Customer(3) { Name = "Elizabeth Brown", City = "London"},
              new Customer(4) { Name = "Ann Devon",   City = "London"},
              new Customer(5) { Name = "Paolo Accorti", City = "Torino"},
              new Customer(6) { Name = "Fran Wilson", City = "Portland"},
              new Customer(7) { Name = "Simon Crowther", City = "London"},
              new Customer(8) { Name = "Liz Nixon",City = "Portland"  }
           };
        }
    }
}
```

Service Oriented Architecture (SOA)

Understanding SOA

An architectural concept or style:
- uses a set of "services" to achieve the desired functionality.

SOA implements
- a set of loosely coupled services that collectively achieve the desired results.

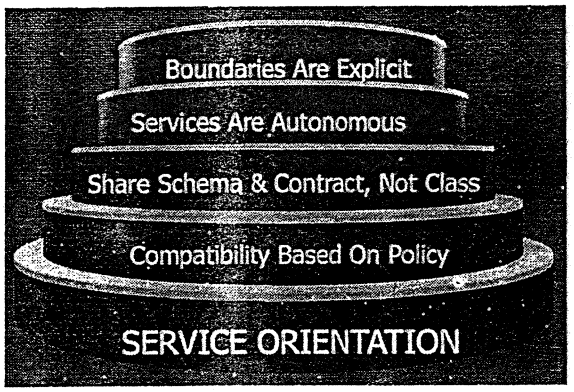
A service is an autonomous (business) system
- that accepts one or more requests and returns one or more responses
  - via a set of published and well defined interfaces.

Design Considerations in SOA

What services do you need?
What services are available for you to consume?
What services will operate together?
What substitute services are available?
What dependencies exist between services and other versions of services?

Service Orientation

A service is based on four fundamental tenets.



Tenet 1: Boundaries Are Explicit

Know your boundaries:
- A well-defined and published public interface is the main entry point into the service.

Services should be easy to consume:
- It should be easy for other developers to consume the service. (without breaking existing consumers)

Keep the service surface area small:
- Provide fewer public interfaces that accept & responds with a well-defined message.

Don't expose implementation details:
- These should be kept internal (loosely coupled)

Tenet 2: Services Are Autonomous

Service versioning and deployment are independent of the system in which they are deployed.
Contracts, once published, should not be changed.

Tenet 3: Services Share the Schema and Contract, Not the Class

Service contracts constitute
data, WSDL, and the policy
Service contract do not change and remain stable.
Contracts should be as explicit as possible

Tenet 4: Service Compatibility Is Based on Policy

Policy separate:
"what is communicated" &
"how/whom a message is communicated"
Policies should be used
to be able to express all the requirements of service interaction via description mechanism (WSDL) alone.

The Connectivity Imperative

Interoperability & Integration
Secure, Reliable, Transacted Messaging
Decoupled, Dynamic Applications

One Question to all Software Developers

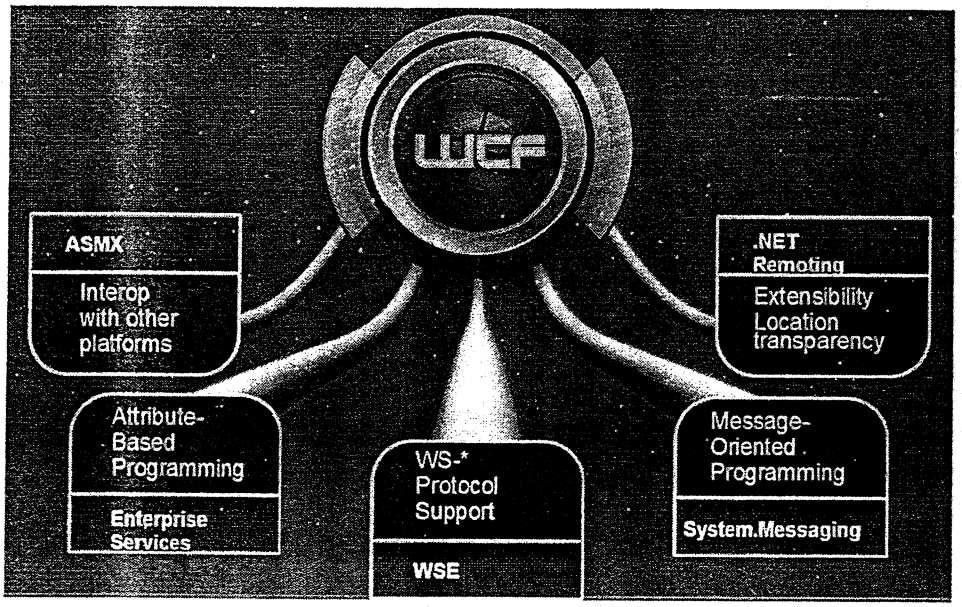
"What would it take for my software to be an asset?"
-Build it to communicate
"I should not be afraid of changing it."
-Provide a thoroughly explicit interface
"I should be able to keep on using it."
-Seamlessly connect into any topology
"I want to be able to see that its working."
-Have rich administration facilities

WCF Services

Windows Communication Foundation

Productivity	Unifies today's distributed technologies Attribute-based development Visual Studio integration
Interoperability	Broad support for WS-* specifications Compatible with existing MS distributed application technologies
Service-Oriented Development	Enables development of loosely-coupled services Config-based communication

Unified Programming Model

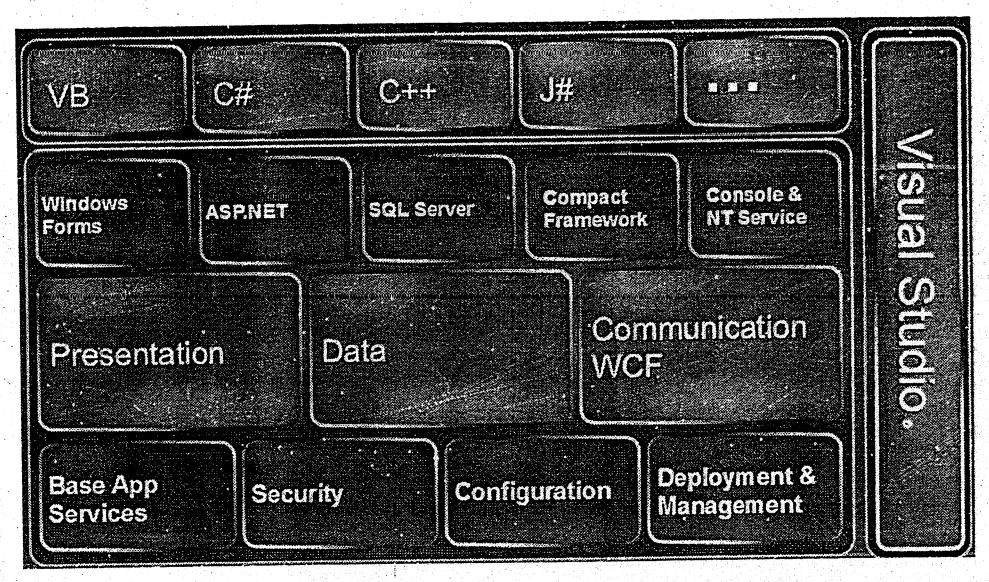


Unified Model Benefits

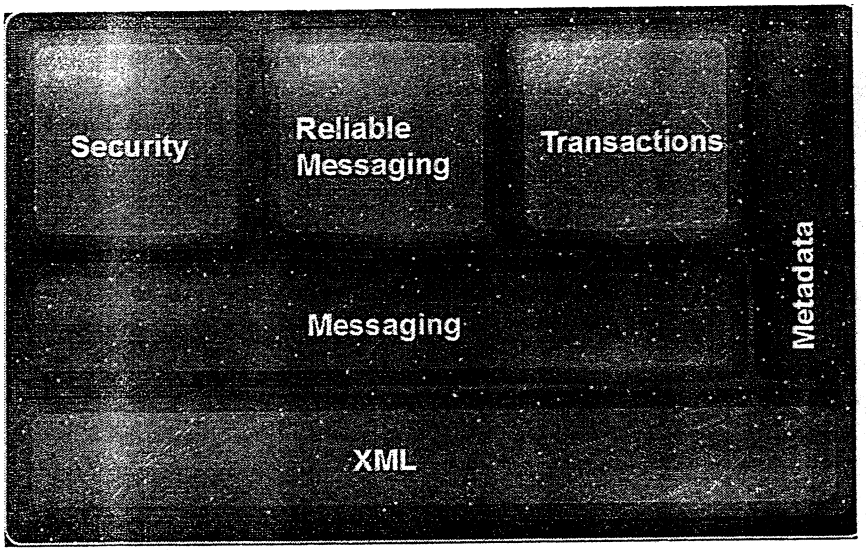
- Programming model
  - Learning curve
  - Consistency
  - Write less code when using multiple technologies
- Flexibility
  - Environments
    - Productivity in development environment
    - Simplify Automated Integration Testing
    - Deployment options in production
      - Design for distribution, run local

Unified Programming Model

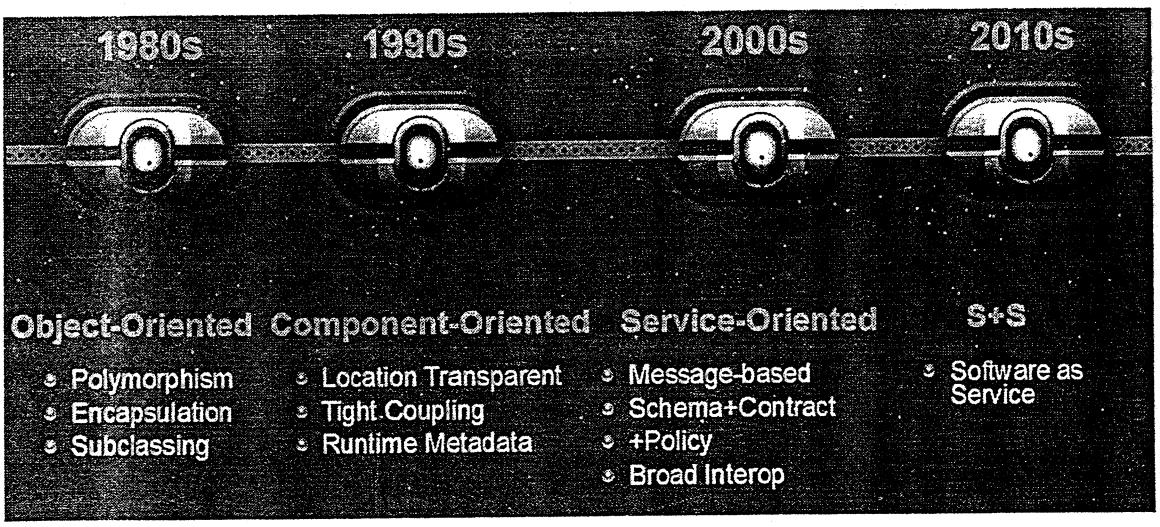
- WCF extends the .NET Framework
- Services are built in Visual Studio using any .NET programming language



WS-* Protocol Support

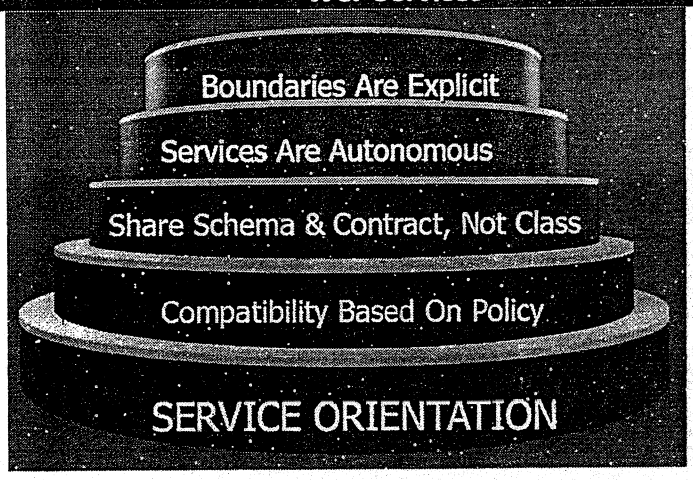


From Objects to Services



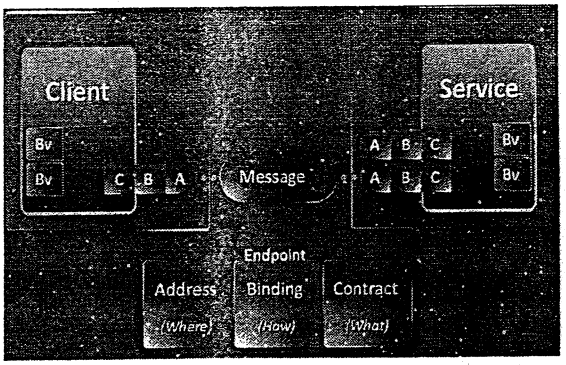
Four Tenets of Service Orientation

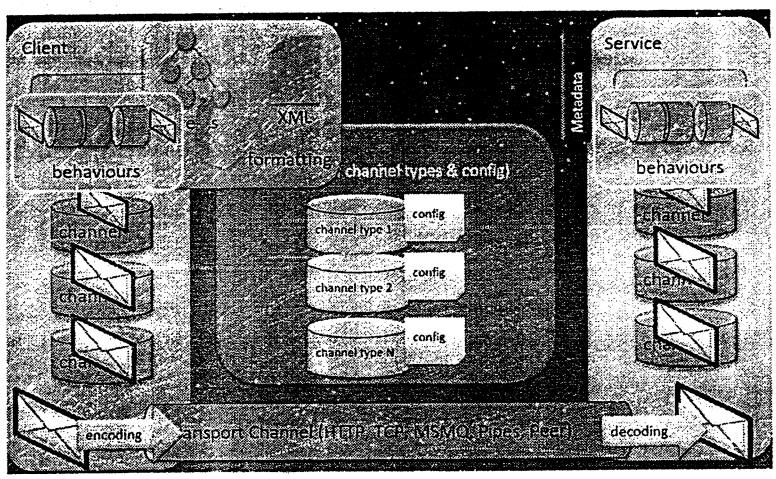
WCF Services



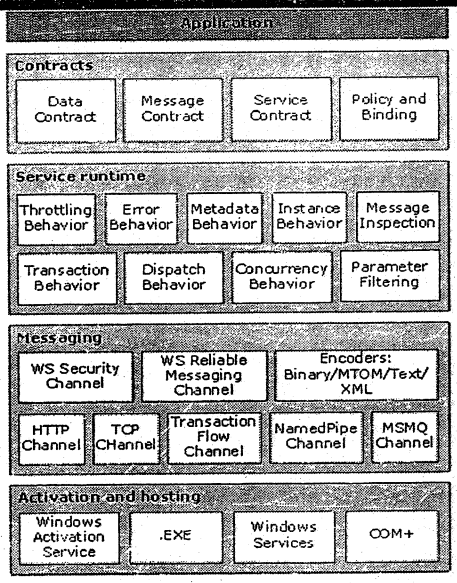
WCF Services

WCF Architecture





WCF Services



Service Endpoints

Address
Where the endpoint can be found?

Binding
How the client communicates with the endpoint?

Contract
What functionality is exposed to the client?

Behaviour
Implementation details

Simple WCF Service

```
[ServiceContract]
public interface IMyInterface
{
[OperationContract]
void MyMethod();
}
```

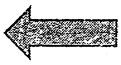
Contract Definition

```
[ServiceBehavior(InstanceContextMode=Single]
public class MyService: IMyInterface
{

[OperationBehavior(Impersonation =
        ImpersorationOption.Required)]
public void MyMethod() {/* do something */}
    }
```

Service Implementation

```
<service name="MyService">
<endpoint
address="net.tcp://localhost:1234/MySvc"
binding="netTcpBinding"
contract="IMyInterface" />
```



Endpoint Definition
Address+Binding+
Contract

WCF **Address**

Service Location

net.pipe://myserver/stockquoteservice
net.tcp://myserver:8080/stockquoteservice
http://myserver/stockquoteservice/
net.msmq://myserver/private/stockquoteresponse

WCF Binding

Binding represent wire-level agreements between a client and a server
Binding is a preconfigured channel stack
Binding specifies
Transport
Encoding
Security

Service Programmer: provide Host

```
                //Self Hosting
ServiceHost serviceHost = new ServiceHost(typeof(StockService),
                          new Uri("http://localhost:8000/Stock"));
serviceHost.AddServiceEndpoint(
                                typeof(IStockService),
                                new BasicHttpBinding(),
                                "");
ServiceMetadataBehavior behavior = new ServiceMetadataBehavior();
behavior.HttpGetEnabled = true;
serviceHost.Description.Behaviors.Add(behavior);
serviceHost.AddServiceEndpoint(
                typeof(IMetadataExchange),
                MetadataExchangeBindings.CreateMexHttpBinding(),
                "mex");
serviceHost.Open();
Console.WriteLine("The services is ready. Press <ENTER> to terminate.\n\n");
Console.ReadLine();
// Close the ServiceHostBase to shutdown the service.
serviceHost.Close();
```

Service Programmer: provide Host

```
//Self Hosting
ServiceHost serviceHost = new ServiceHost(typeof(StockService));
serviceHost.Open();
// The service can now be accessed.
Console.WriteLine("The services is ready. Press <ENTER> to terminate.\n\n");
Console.ReadLine();
// Close the ServiceHostBase to shutdown the service.
serviceHost.Close();
```

Service Programmer: Provide Host

```
//IIS Hosting
//Within IIS 5.1 or 6 (HTTP only) or 7 (any transport)
//DealAnalyzer.svc
<%@Service Class="WoodgroveBank.DealAnalyzer" %>
<%@Assembly Name="DealAnalyzerAssembly" %>
```

Service Administrator: Configure

```
<configuration>
  <system.serviceModel>
    <services>
      <service name="Stock.StockService"
               behaviorConfiguration="myServiceBehavior">
        <endpoint address=" http://localhost:8000/Stock "
               binding="basicHttpBinding"
               contract="EssentialWCF.IStockService" />
        <endpoint address="mex"
               binding="mexHttpBinding"
               contract="IMetadataExchange" />
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior name="myServiceBehavior">
          <serviceMetadata httpGetEnabled="True"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>
</configuration>
```

Client Programmer: Invoke service

```
[ServiceContract]
public interface IStockService
{
```

```
        [OperationContract]
        double GetPrice(string ticker);
    }
ChannelFactory<IStockService> myChannelFactory =
            new ChannelFactory<IStockService>(
            new BasicHttpBinding(),
            new EndpointAddress("http://localhost:8000/Stock"));
IStockService wcfClient = myChannelFactory.CreateChannel();
double p = wcfClient.GetPrice("msft");

Console.WriteLine("Price:{0}",p);
```

Client Programmer: Invoke service

Download Metadata
Generate Typed Proxy
Generate Address
Generate Binding

SvcUtil.exe "http://localhost:8080/Deals/AnalysisService"

```
        /out:DealAnalysisProxy.cs /config: app.config
Invoke Typed Proxy Methods
  StockServiceClient proxy = new StockServiceClient();
        double p = proxy.GetPrice("msft");
        Console.WriteLine("Price:{0}", p);
        proxy.Close();
```

What is WCF Contract?

WCF Contracts

Service	Describe the operation a service can perform.
Data	Describe a data structure.
Message	Defines the structure of the message on the wire

A contract specifies:
Operations can be called by a client.
The type of input parameters or data required to call the operation.
What type of processing or response message the client can expect.
Structure of the message
Behavior of the service

Types of Contract

ServiceContract:

Defines Operations, Behaviors
What does your service do?

OperationContract:
Applied on methods which are available for client

DataContract:
Used for Serializing Custom Defined types from service to client.
Defines Schema and Versioning Strategies
What object data needs to flow back and forth?

MessageContract:
Used for embedding custom types in message.
Allows defining application-specific headers and body content
Allows control over the SOAP structure of messages

FaultContract:
Used for exposing exceptions from service to client.

Service Contract

```
using System.ServiceModel;
[ServiceContract]
public interface ICalculate
      {
[OperationContract]
double Add( double a, double b);
[OperationContract]
double Subtract( double a, double b);
      }
```

Service Contract: Duplex

```
      [ServiceContract(Session=true,
    CallbackContract=typeof(ICalculatorResults))]
      public interface ICalculatorProblems
      {
[OperationContract(IsOneWay=true)]
void SolveProblem (ComplexProblem p);
      }
      public interface ICalculatorResults
      {
[OperationContract(IsOneWay=true)]
void Results(ComplexProblem p);
      }
```

[OperationContract]

Defines operations provided by the service
Properties
IsOneWay
IsInitiating
IsTerminating
Action

WCF Services

Client Service Interaction

One Way:
Datagram-style delivery
Request-Reply
Immediate Reply on same logical thread
Duplex
Reply "later" and on backchannel (callback-style)

Operation Contract: OneWay

```
[ServiceContract]
public interface IOneWayCalculator
{
[OperationContract(IsOneWay=true)]
void StoreProblem (ComplexProblem p);
}
```

Data Contract

[DataContract]

```
public class ComplexNumber
        {
    [DataMember]
    public double Real = 0.0D;
    [DataMember]
    public double Imaginary = 0.0D;

    public ComplexNumber(double r, double i)
        {
        this.Real = r;
        this.Imaginary = i;
        }
        }
```

Serialization

WCF supports two serializers out-of-the-box
DataContractSerializer (DCS)
XmlSerializer (XS)
Custom serializers can be written and hook it in Services

DataContractSerializer

DataContractSerilizer supports
CLR Primitive Types: Int32, Decimal, Boolean
Other Primitive Types: Byte Array, DateTime, etc
Enums
[DataContract], [CollectionDataContract]
IXMLSerializable
Arrays and Collections
[Serializable], Iserializable
Generic and Nullable Types
Suppcrts Classes, interfaces (as Object) and structures

Not supported
XMLElementAttribute and XMLAttributeAttribute
sUse XmlSerializer instead

DataContract Serializable Objects

Data Contract
Decorated public and private fields and properties
In System.Runtime.Serialization namespace
[DataContract] attribute required
Serializable
All public and private fields, except [NonSerialized]
Use [OnDeserializing] instead of the constructor
[Serializable] attribute required
ISerializable optional
XML Serializable
Must implement IXmlSerializable.

Collections

Collections are interchangeable
All list collections of the same type are considered to have the same data contract

```
[DataContract(Name = "PurchaseOrder")]
public class PurchaseOrder1
{
    [DataMember]
    public Collection<Item> Items;
}
```

```
[DataContract(Name = "PurchaseOrder")]
public class PurchaseOrder2
{
    [DataMember]
    public List<Item> Items;
}
```

Same applies to dictionaries of the same type
Concrete classes and interfaces

[CollectionDataContract]
More control over collection serialization

[MessageContract]

```
[MessageContract]
public class ComplexProblem
      {
      [MessageHeader]
public string operation;
      [MessageBody]
public ComplexNumber n1;
      [MessageBody]
public ComplexNumber n2;
      [MessageBody]
public ComplexNumber solution;
// Constructors…
}
```

[FaultContract]

```
[ServiceContract(Session=true)]
      public interface ICalculator
            {
[OperationContract]
[FaultContract(typeof(DivideByZero))]
ComplexProblem SolveProblem (ComplexProblem p);
            }
;

try {
 return n1 / n2;
      }
catch (DivideByZeroException e) {
DivideByZero f = new DivideByZero ("Calculator");
throw new FaultException<DivideByZero>(f);
}
```

WCF Binding

Bindings & Channels

A Binding is composed of a set of Binding Elements.
Each Binding Element belongs to a Binding Layer
Binding Layer defines
Transport
Encoding
Protocol
A Channel is a concrete implementation of a Binding Element.

Defining Endpoints

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
  <system.serviceModel>
    <services>
      <service serviceType="CalculatorService">
        <endpoint address="Calculator"
                  binding="basicHttpBinding"
                  contract="ICalculator" />
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

Bindings

Binding	Security[Default]	Session[Default]	Transaction	Duplex
BasicHttpBinding	None,Transport,Message	None	n/a	
WSHttpBinding	Transport,Message,Mixed	None,Transport,Reliable Session	n/a	
WSDualHttpBinding	Message	Reliable Session	Yes	
WSFederationHttpBinding	Message	None,Reliable Session	No	
NetTcpBinding	Transport,Message	Reliable Session,Transport	Yes	
NetNamedPipeBinding	Transport	None,Transport	Yes	
NetMsmgBinding	Message,Transport,Both	None	No	
NetPeerTcpBinding	Transport	None	Yes	
MsmgIntegrationBinding	Transport	None	n/a	

Web Service Bindings

BasicHttpBinding
for Simple Object Access Protocol(SOAP) version1.1 compatibility
WSHttpBinding
for SOAP version 1.2 with WS*
WSDualHttpBinding
for callbacks over HTTP with WS* support (not interoperable)
WSFederationHttpBinding
for federated security and single sign-on (SSO) scenarios

Cross-Process / Machine Bindings

NetNamedPipeBinding
for in-process or same-machine calls
NetTcpBinding
for same-machine or cross-machine calls
NetPeerTcpBinding
for same-machine or cross-machine peer-to-peer messaging

Messaging Bindings

NetMsmqBinding
for reliable, transacted, and persistent messaging over MSMQ
MsmqIntegrationBinding
for MSMQ interoperability with earlier technologies

NetTcpBinding

Default binding is faster than http-based
Binary-encoded SOAP directly over TCP

NetMsmqBinding

Requires one way operations
Messages sent to the message queue rather than the endpoint
Multiple WCF messages can be mapped to one MSMQ message if SessionMode= Required.
Binary-encoded SOAP over MSMQ

NetNamedPipeBinding

Binary-encoded SOAP over named pipes
communication between processes on the same Windows machine.

Custom Bindings

```
            <bindings>
            <customBinding>
        <binding configurationName="Binding1">
            <reliableSession bufferedMessagesQuota="32"
                inactivityTimeout="00:10:00"
                maxRetryCount="8"
                ordered="true" />
            <httpsTransport manualAddressing="false"
                maxMessageSize="65536"
                hostnameComparisonMode="StrongWildcard"/>
            <textMessageEncoding maxReadPoolSize="64"
                maxWritePoolSize="16"
                messageVersion="Default"
                encoding="utf-8" />
        </binding>
    </customBinding>
</bindings>
```

Prasanna Wadekar

Custom Binding

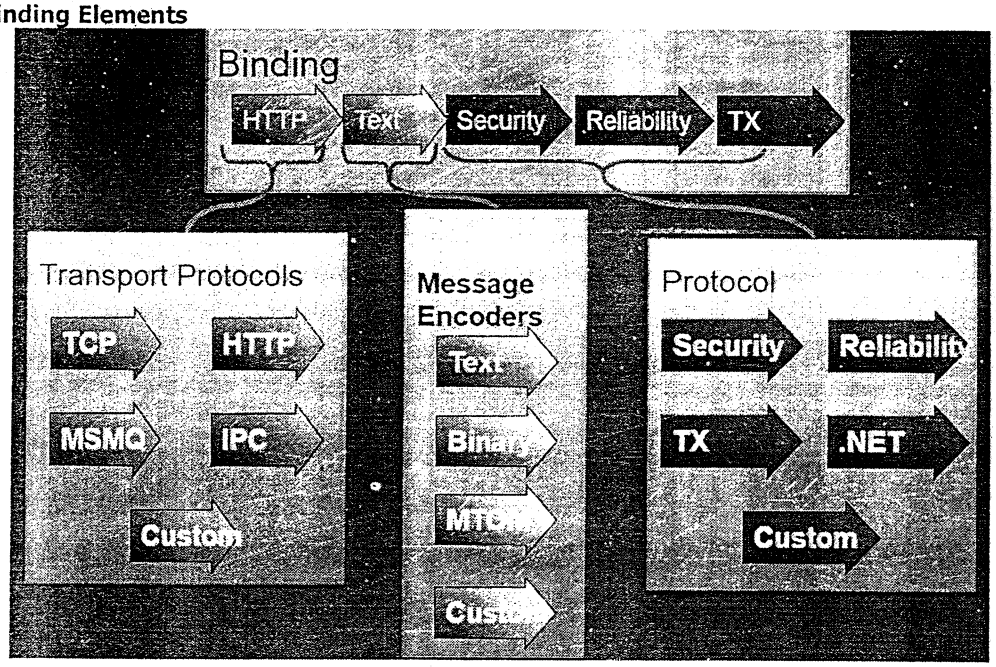
```
System.ServiceModel.Channels.Binding customBinding= new CustomBinding (
      new SecurityBindingElement(),
      new MtomMessageEncodingBindingElement(),
      new TcpTransportBindingElement());
```

Custom
- Use CustomBinding
- Fill with binding elements

Inherit
- Derive from system-provided bindings

User-defined
- Derive from Binding
- Add your own or system-provided elements

Bindings & Binding Elements



WCF Hosting

Service Host is an Operating System Process
Responsible for lifetime and Service Context of WCF Service.
Service host knows very little about the WCF service that runs in its memory space
Any operating system process can be a service host.
Console, Windows Forms, WPF Application
Windows NT Service
IIS
WAS (Windows Activation Service)

Self-Hosting

Console, Windows Forms, WPF Application
A ServiceHost instance must be initialized to expose endpoints for a service
Core methods:
Open() – opens channel listeners
Close() – Closes channel listeners

Service Clients

Through code or configuration
Service knowledge
Extract metadata
SvcUtil.exe
Custom tool
Generates code and config

Client Typed Proxy

ClientBase<TChannel>

```
Internal class MyCustomerServiceClient: Client<IMyCustomerService>,
IMyCustomerService
    {
public MyCustomerServiceClient(string endpointConfigurationName)
      {
      }

public MyCustomerClient(Binding binding,EndpointAddress endpointAddress)
:base(binding,endpointAddress)
{
}
```

ChannelFactory<TChannel>

```
internal class MyCustomerServiceClient:ClientBase<IMyCustomerService>,IMCustomerService
           {
           }
ChannelFactory<ICustomerService>.CreateChannel(new NetTcpBinding(), new
EndpointAddress(NETTCP_URI + SERVICE_NAME));
```

Hosting Multiple Services

```
                public static void Main()

                {
ServiceHost serviceHost1 = new ServiceHost(typeof(GoodStockService));
            serviceHost1.Open();
ServiceHost serviceHost2 = new ServiceHost(typeof(GreatStockService));
            serviceHost2.Open();
            ……..
            ……..
            serviceHost1.Close();
            serviceHost2.Close();
}
```

Windows Applications

Windows Forms or WPF
Exposing services from client machines
Must be aware or synchronization context
UI thread or not
If ServiceHost is opened on a non-UI thread, services operate on a new thread
If on UI thread, services join it unless UseSynchronizationContext set to false

Windows Services

Host environment initialized
when machine starts, can be configured to restart on failure
Open and close ServiceHost instances on start and stop of the Windows Service

IIS Hosting

IIS provides Hosting Infrastructure
Process activation
Recycling
Identity management
Steps for IIS Hosting
create an IIS Virtual Application,
create an SVC file to define the service implementation
Include <System.ServiceModel> section in web.config

WAS Hosting

Scenarios in which services to be developed for
Offline client server interaction (disconnected environment)
Chat services (quickly sending numerous small messages as a part of a larger conversation)
IIS Hosting can not provide above functionality
Windows Activation Service (WAS)
WAS is the hosting infrastructure built into Vista and Windows Server 2008
WAS supports multiple protocols through a listener adapter architecture

WAS can communicate over HTTP, TCP, MSMQ, and namedpipes using a consistent mechanism, thereby improving system reliability

Hosting a service in WAS is similar to hosting in IIS.

Additional Steps:

Install (or confirm the installation of) the WCF activation components.

Configure WAS to support a non-HTTP protocol.

Install (or confirm the installation of) the WCF activation components.

To install the WCF non-HTTP activation components

Click the Start button, and then click Control Panel.

Click Programs, and then click Programs and Features.

On the Tasks menu, click Turn Windows features on or off.

Find the .NET Framework 3.5 node, select and then expand it.

Select the WCF Non-Http Activation Components box and save the setting.

Configure WAS to support a non-HTTP protocol.

To support net.tcp activation, the default Web site must first be bound to a net.tcp port

```
%windir%\system32\inetsrv\appcmd.exe set site "Default Web Site" -
+bindings.[protocol='net.tcp',bindingInformation='808:*']
```

enable net.tcp for the application

```
%windir%\system32\inetsrv\appcmd.exe set app "Default Web Site/<WCF
Application>" /enabledProtocols:http,net.tcp
```

Prasanna Wadekar 9970248407

Advanced Web Programming - Notes

* Languages
  - **compiled**
    - compiler is required
    - compiles the whole source code into an executable
    - errors are detected at the time of compilation
    - compilation cannot proceed with error
    - faster
    - eg: c, c++
  - **interpreted**
    - interpreter is required
    - interprets the code line by line
    - errors are detected at run time (executing the code)
    - execute can not proceed with error
    - slower
    - e.g.: HTML, JavaScript, (all scripting languages)

* **Applications**
  - web applications (web site)
    - browser is compulsory
    - mainly HTML
    - browser can understand HTML
    - platform indenpendant
    - slower
    - eg: HTML
  - **Native applications**
    - OS + CPU
    - mainly ASM language
    - CPU can understand only ASM
    - platform specific
    - faster
    - eg. C,C++

PRASANNA

* **Web Application**
  - HTML: design UI
  - CSS: to add decoration in HTML code
  - JavaScript: to add programming logic in HTML

* **HTML**
  - ML: tags + data (content within tags)
    - **tag:**
      - word enclosed in < and >
      - types
        - starting tag:
        - closing tag:
        - empty tag:
</br> =

- root tag:
  - also known as document type/tag/element
  - tag which starts a document and which ends the document

- **Parts**
  - head
    - title: shows the title on tab/window
    - meta: to add extra information (
    - style: to add CSS styles
    - link: to link external documents
    - script: to add JavaScript code
    - base: to decide the base url of the website
      - to configure the anchor tags
  - **body**
  - Text Elements
    - inline
      - these elements do not create new line
      - eg: span, superscript etc
    - **block**
      - these elements create a new line at the end
      - eg: para, div, h1
  - **Link elements**
    - anchor
  - Table
    - head
    - body
    - footer
  - **List**
    - ul
    - ol
    - definition list (dl)
  - **Form**
    - input
    - select
    - textarea

*** Default Attributes**
- style: to add the CSS properties
- id: to identify the tag uniquely
- class: to identify similar type of element having same requirements

*** CSS**

- Cascading Style Sheet
- is introduced in HTML 4.0 to add decoration
- CSS3
- **Types**

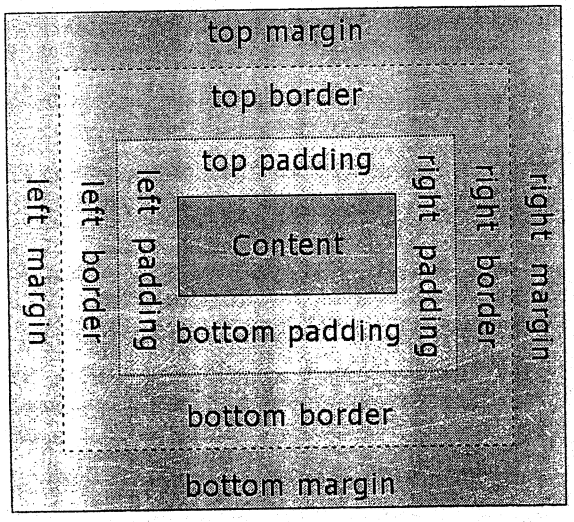
- inline style
  - to add decoration in a tag
  - **Disadvantage:**
    - repeat the same info for every tag
- **Internal Style**
  - to target multiple elements at once
  - internal to the current document
  - disadvantage:
    - repeat the same info for every page
- **External Style**
  - to target multiple pages at once
  - link tag is used to link external css file with the html document
  - eg: <link href=" <file name>" type=" text/css" rel=" stylesheet" >
- browser default style
  - this style is by default applied on every html page
  - this style is browser dependant

- **Order**
  - browser default style: 1
  - external style: 2
  - internal style: 3
  - inline style: 4

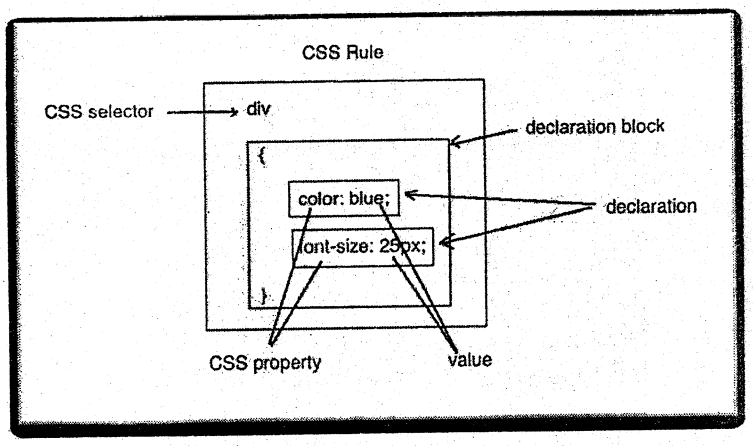
*** CSS selector**

- which can target multiple elements/tags at once
- types of selector
  - element selector
    - selects similar type of elements
    - e.g.: div{color:red;} -> all div elements will be in red color
  - group selector (,)
    - can select multiple type of elements
    - e.g.: div, p {"} -> all div and para will be targeted
  - id selector (#)
    - to target only a specific tag (having specific id)
    - eg: div#specialdiv {..}
  - class selector (.)
    - to target elements having same class name
    - eg: div.greendiv {..}
  - pseudo selector: (:)
    - which gets executed in a specific condition
    - eg:
      - :hover -> when mouse pointer is on top of this element
      - :nth-child -> for n-child

CSS Box Model →



CSS Rule →



***JavaScript**

- is a interpreted language
- browser acts as interpreter
- is a object based language
- usage:
  - to add programming logic in HTML
  - these days it is also used by third party software like Unity3D to add programming logic in mobile games
  - these days it is also used on server to communicate with Web server to create dynamic web pages

*** Variables**

- data type can NOT be assigned in JavaScript
- JS decides the data type by looking at the variable"™s CURRENT value
- variables can be redeclared

- eg:

```
name = "sunbeam"□;  // string
salary = 6.7;      // number
count = 10;        // number
canVote = true;    // boolean
```

- **Scopes**
  - Global scope
    - create a variable without using var keyword inside a function
    - or declare a variable outside any function
    - e.g.: name = "sunbeam"□;
  - Local scope
    - declare a variable with var keyword inside a function
    - e.g.: var myname = "sunbeam"□;
  - var keyword is used to declare a variable
    - if var is used inside a function: it creates local var

```
function function1() {

var myname = "test"□; // local
}
```

    - if var is used outside a function: it create global var

```
var myname = "test"□; // global

function function1() {
}
```

- **function**
  - function is a keyword used to declare a function
  - syntax:

```
function <function name> (<number of parameters>) {
   // function body
}
```

  - a function can be called by passing variable number of parameters
    - if less number of params are passed: then remaining will be assigned undefined value
    - if more number of params are passed: then extra params will be ignored

```
function add(num1, num2) {
  console.log("num1 = "□ + num1);
  console.log("num2 = "□ + num2);
```

```
}

add();      // num1 = undefined, num2 = undefined
add(10);    // num1 = 10, num2 = undefined
add(10, 30); // num1 = 10, num2 = 30
add(1,2,3,4); // num1 = 1, num2 = 2: rest of the params will be ignored
```

- **JS pre-defined words**
  - undefined:
    - a var is declared but not defined
    - the value is not set
  - NaN:
    - not a number
    - mathematical operations are performed on a non-number value
    - eg: undefined + undefined = NaN
      undefined + 10 = NaN
      "test "□ * "test "□ = NaNs

- **Browser**
  - special application which can understand HTML
  - it is used to render/display html pages
  - **components**
    - UI component
    - data storage
      - store data
      - cookies
      - HTML5
        - session storage
        - local storage
    - networking component
    - javascript engine
    - rendering engine
      - Apple Safari: webkit
      - Google Chrome: webkit -> Blink
      - Mozilla Firefox: Gecko
      - Opera: gastro
      - IE: -

* **URL**

- Uniform Resource Locator
- to identify any resource from internet uniquely
- contains components
  - scheme: (http)
    - generally known as protocol

- decided how to fetch the resource from internet
- eg: http, https, file, ftp, tftp etc
- domain information: (www.mywebsite.com)
- www: sub-domain
- mywebsite: domain name (gets resolved into a unique IP address by DNS)
- com: top level domain
- port number: (80)
- identifies the service running on the server
- if port number is not provided then the default port number for specified scheme is used
- eg: http:80, https:443, ftp:21
- file / path: (myfolder/myfile.html)
- path to the resource
- query string: (firstname=test&seconname=test)
- is the way to provide input to a page
- it always present in key=value format
- e.g.: firstname=test and secondname=test
where: firstname, secondname are keys
test, test are values
- multiple key-value pairs are separated by &
- hash (#) component: (#top)
- used to link a part of same document

eg: http://www.mywebsite.com:80/myfolder/myfile.html?firstname=test&seconname=test
eg: http://www.mywebsite.com:80/myfolder/myfile.html#top

*** Object**

- collection of different data members (properties) and functions associated with them
- **2 ways to create an object**
- by using new Object
- Object is a kind of template given by JavaScript
- eg:

```
var person = new Object();    <- Object
person.name = "test";         <- property
person.address = "Karad";     <- property
console.log("name : " + person.name);

var person1 = new Object();
person1["name"] = "test1";
person1["address"] = "Pune";
console.log("name : " + person["name");
```

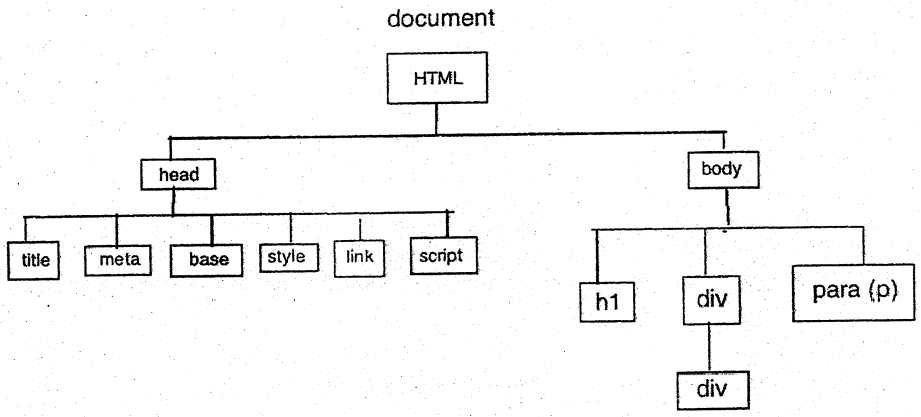
- json
- JavaScript Object Notation
- the simplest way to create an object

- these days, json is used to transfer data from one entity to another
  - used to send data from server to client (mobile Facebook application)
  - because
    - json is light weight text file
    - easy to parse (extracting information) json response
- eg:

```
var person = {
    "name":"test" ,
    "address":"karad"
};
console.log("name : " + person.name);
```

- DOM

- Document Object Model



- hierarchy of tags
- W3C has 3 types
  - Core DOM
    - language independent
  - HTML DOM
  - XML DOM

- JavaScript built in objects

- these objects are created by browser at the time of loading the page
- document
  - representation of HTML DOM
    - using DOM object (document), html tags can be accessed/modified at run time
  - it contains JS object of every html tag inside the html file
  - using document any html tag can be accessed by using:
    - getElementById(): returns a tag which has the same id as that of the input
    - getElementsByTagName(): by matching the tag names
    - getElementsByName(): by matching the name attribute

- getElementsByClassName(): by matching the class name
- every tag contains property named innerHTML which can be used to set / get the contents of that tag except input (value)
- sessionStorage:
- localStorage:
- navigator:

*** **HTML5 functionality** ***

* **Web Storage**
- used to store data temporarily or permanently
- key-value pairs can be stored in side web storage
- only ASCII character (string) can be stored in web storage
- implemented and maintained by browser internally
- stored in different location for different browser
- **Types**
  - session storage
    - stores the data temporarily
    - till the session is alive
    - session:
      - the duration of browser
      - it gets created at the time of opening browser and gets closed at the time of closing browser
  - local storage
    - data gets stored permanently
    - data upto 5MB can be stored in local storage

* **Geolocation**
- to get the current location in terms of Latitude and Longitude
- ways (providers) to retrieve the current position
  - GPS hardware
  - IP address
  - cell triangulation (A-GPS)

* **Multimedia**
- video

* **Canvas**

* **Pop up box**
- alert
  - used to display readonly message
  - e.g. alert("hello..");
- prompt
  - used to get a single line input from user
  - e.g. var name = prompt("enter name..");
- confirm

- used to get confirmation from user
- e.g. var answer = confirm("are you hungry?"); // true or false

*** Date object**

*** timer functions**
- used to perform functionality after an interval
- setInterval()
  - used to call a function every after some interval
  - setInterval(<function name>, <interval in mili-seconds>);
- clearInterval()
  - stops calling a function which is started by setInterval()
- setTimeout()
  - used to call a function only once after few mili-seconds
  - e.g. setTimeout(<function name>, <interval in mili-seconds>);
- clearTimeout()
  - stops calling the function which is started by setTimeout()

*** jQuery**
- javascript library of commonly used functions
- benefits
  - free and open source
  - has a very good community

*** XML**
- is a Markup Language
  - made up of elements and data
- eXtended Markup Language
- standardized language by w3c
- imp
  - the elements are user-defined: xml writer must define the xml elements
  - simple text file format
  - xml is interpreted
  - an interpreter is require: by default browser is used to interpret the xml
- usage
  - structuring the data
  - to transfer data from one entity to another in commonly understandable format
  - to store very small amount data
- xml vs html
  - html is used for designing user interface for browser
    xml is NOT used to design UI
  - html has pre-defined tags
    xml has user-defined elements
  - html is case in-sensitive : <table></Table>
    xml is case sensitive

* XML element
- word enclosed in < and >
- types
  - starting element: <name>
  - closing element: </name>
  - empty element
    - normal syntax: <name></name>
    - shorthand: <name />
  - root element
    - which starts and closes the document
    - also known as document type or document element
- rules to create an element
  - every file must have one and only one root element
  - Special characters like space can not be used in element
    - <First Name> <- incorrect element
    - <FirstName>, <First_Name>
  - only _ can be used in element
    - <First_Name>
  - elements cannot start with number
    - <1Name> <-- incorrect element-->
    - <Name>
  - every starting element must be closed with closing element
    - <Name>test <- incorrect -->
    - <Name>test</Name>
  - element name is case sensitive
    - <Name> and <name> are two different element

* Attribute
- it adds more information in xml element
- it is presented in name=value pair
  - e.g. phoneType="work"
- rules
  - can not be repeated
    - e.g. <Phone phoneType="work" phoneType="special"></Phone>
      <Phone phoneType="work"></Phone>
  - in name=value format
  - can NOT have multiple values at the same time
    - e.g. <Phone phoneType="work","special"></Phone>
  - can NOT have a child attribute
- attribute-only element
  - having only attributes
    - e.g. <Person name="test" address="Karad" />
- mixed type element
  - has both: child element and attribute
  - <Person name="test" address="Karad">
    <Phone phoneType="special">4544</Phone>
    <Phone phoneType="work">4544</Phone>

<Phone phoneType="personal">45464</Phone>
</Person>

*** Parts**
- XML Header
  - <?xml version="1.0" encoding="UTF-8" ?>
  - it is optional
  - where:
    - version: specification version used to create this xml
    - encoding: type of characters using in this xml
      - UTF: Unicode Transformation Format
        - UTF-8: 8 bit or 1 byte
        - UTF-16: 16 bit or 2 bytes (wide character)
- XML body
  - collection of user-defined elements

*** CDATA**
- the characters included in the CDATA section are not parsed
- special meaning of these characters will not be used while presenting xml
- <![CDATA[> <data> <]]>

* Well formed xml document
- the xml is following all the syntactical rules specified by W3C

*** Valid XML**
- a well formed xml which follows all the logical rules as well
- logical rules can be defined by using DTD or XML Schema
- every Valid xml must be well formed xml but every well formed xml may NOT be valid xml

*** DTD**
- Document Type Definition
- used to define the logical rules
- types
  - a root element (!DOCTYPE)
    - used to defined a root element
    - <!DOCTYPE <element name> [<logical rules>]>
  - elements
    - simple element (!ELEMENT)
      - <!ELEMENT <element name> (#PCDATA)>
      - where
        - #PCDATA: all parcelable characters
    - element having child element(s) where order is imp
      - <!ELEMENT <element name> (<child element1>, <child element2>...)>
      - element is made up of sequence of child elements
      - child elements must be present in the pre-defined order
    - element having child element(s) where order is not imp
      - <!ELEMENT <element name> ANY>

- where
  - ANY: any element in any order
- empty element
  - element having no data
  - <!ELEMENT <element name> EMPTY>

- **Wild characters**
  - *: element can appear zero or more times
  - +: element can appear one or more times
  - ?: element can either appear only once or absent

- **attributes**
  - !ATTLIST is used to define an attribute
  - <!ATTLIST <element name>
    <attribute name> CDATA <type> >
  - e.g.

```
<!ATTLIST Phone
    phoneType CDATA #IMPLIED>
```

where
- type = #IMPLIED -> optional -> ignore
- type = "value" -> default value -> default value
- type = #REQUIRED -> compulsory

*** DTD**

- Types
  - internal DTD
    - the DTD definition written in the same xml file
    - the file becomes cluttered with xml content and DTD content
  - external DTD
    - a new file with .dtd extension must be created with all the definitions
    - external DTD can NOT start with DOCTYPE
    - to load external DTD SYSTEM keyword is used
    - <!DOCTYPE <element name> SYSTEM "<dtd file name>">
  - mixed DTD
    - combination of both internal and external DTD
    - <!DOCTYPE <element name> SYSTEM "<dtd file name>" [
      <internal DTD rules>
      ]>

- **Limitations**
  - DTD has its own syntax
  - DTD can not differentiate between character and number
  - values can not be restricted using DTD
  - DTD can not add any restriction on type of data and number of times it gets repeated

- DTD can not understand the xml namespace

*** xml namespace**
- group which separates multiple elements having same logical structure
- xmlns keyword is used
  - syntax:
    <table xmlns:<prefix>="http://<namespace name>"></table>
- to put an element inside a namespace
  - <prefix:<element name>>
  - e.g. <p1:table>..</p1:table>
- rules
  - every namespace can be used inside the same element or child elements where it is defined
  - by default all the child elements are the part of same namespace of their parent element
  -

*** XML Schema**
- xml schema is another way to define logical rules
- data types
  - string: accepts any character including number and special symbols
  - integer: positive and negative
  - decimal: float values
  - positiveInteger:
  - negativeInteger:
  - date:
  - time:
  - dateTime:
  - positiveDecimal:
  - negativeDecimal:
- custom types
  - add restrictions

- element Types
- Simple element
  - does NOT contain any child element
  - does NOT contain any attribute
- Complex element
  - may contain at least a single child element
  - may contain at least a single attribute

- rules
- xml schema must be written outside the xml file (externally)
- the file extension must be .xsd (xml schema definition/document)

- Element
- use: xs:element
- syntax:
  <xs:element name="<name of element" type="<element type>" />

- to create custom type

```
<xs:element name="<element name>" >
<xs:simpleType>
  <xs:restriction base="<data type>">
      <!— add the required condition —>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- types
  - simple element

```
<xs:element name="<name of element>" type="<element type>" />
```

  - element having child element(s) in a specific order
    - all the child element(s) must be present

```
*  <xs:element name="<element name>" >
<xs:complexType>
  <xs:sequence>
      <!— child element(s) —>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- element having child element(s) where order is not imp
  - all the child element(s) must be present

```
*  <xs:element name="<element name>" >
<xs:complexType>
  <xs:all>
      <!— child element(s) —>
    </xs:all>
  </xs:complexType>
</xs:element>
```

- element having child element(s) where only one child element must be present

```
*  <xs:element name="<element name>" >
<xs:complexType>
  <xs:choice>
      <!— child element(s) —>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

- element having repeating child element
  - occurrence indicator
    - minOccurs: minimum number of times the element should me present
      - = 0: element becomes optional
    - maxOccurs: maximum number of times the element should me present
      -> unbounded: infinity

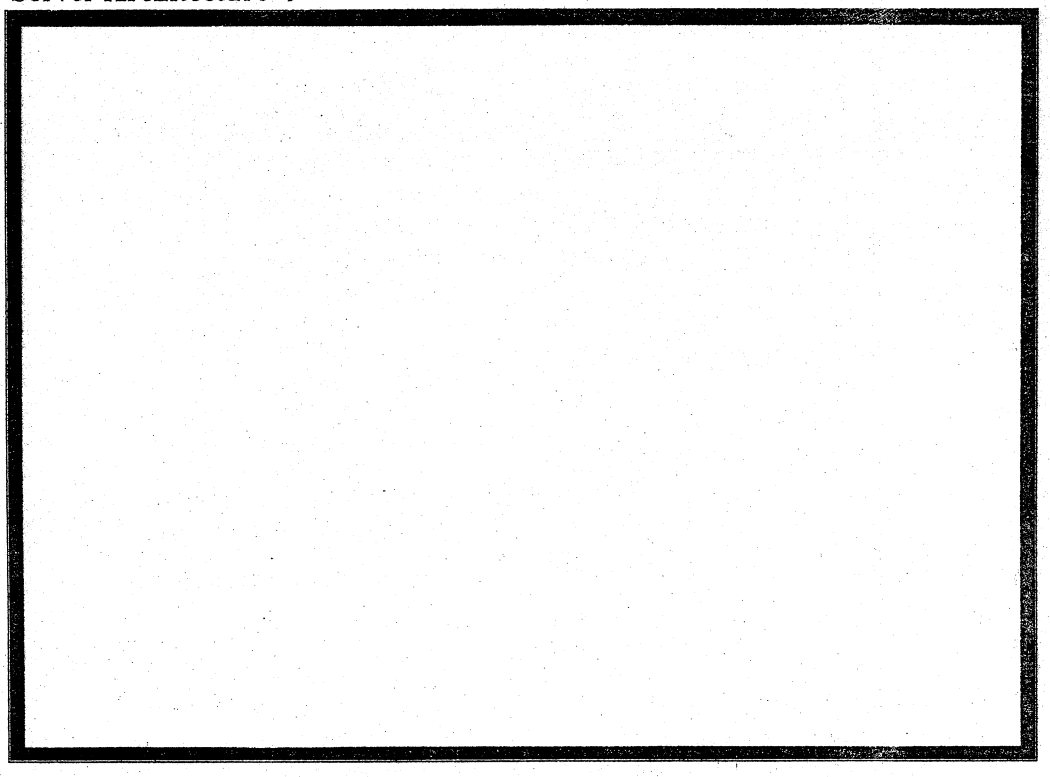
- element having an attribute

*** Path**

- File system path
  - Absolute path
    c:\windows\calc.exe
  - Relative path
    c:\ -> windows\calc.exe
    c:\windows -> calc.exe

*** PHP**

Client – Server Architecture →



- Personal Home Page
- PHP: Hypertext Preprocessor
- **Features**
  - using PHP, dynamic web pages can be designed

- PHP is a scripting language
- PHP is a interpreted language
- Why PHP
- its very simple to learn and use
- Open source and free
- it can be used to design huge web site (having many pages)
- supports wide range of Database like MySQL, Oracle, SQL Server etc

*** Language Features**

- code must be written inside <?php ?>
- if page contains only PHP code then ?> is optional
- if page contains both PHP as well as HTML then ?> is compulsory
- Syntax
- all the keywords, function names and class names are case in-sensitive
- variable names are case sensitive
- Variables
- variable name must start with $
- $name = "sunbeam";
- variable can not decide its data type
- php decides the data type by looking at the current value
- data types
- string:
-
- int: (integer) -> whole numbers
- float: decimal numbers
- bool: boolean (true/false)
- Array:
- NULL:
- object: instance of a class
- Array
- collection of similar or dis-similar data type
- syntax: $array = array(<value1>, <value2>....);
- count(): returns the length of array
- types
- single dimensional
- multi-dimensional
- associative array
- function
-
- scope
- local
- declared inside function
- global
- declared outside of any function
- global variables can NOT be used directly inside function
- global keyword must be used to use a global variable inside a function

```
$num = 100;
    function function1() {
        global $num;        ///  <- global keyword is required here
        echo "Num = $num";
    }
```

- static
  - that retains it value

* **Superglobals**
- associative array
- $_GET:
  - array which holds all the input parameters sent using GET method
- $_POST:
  - array which holds all the input parameters sent using POST method
- $_REQUEST:
  - holds all the input parameters present in Request
- $_SERVER:
  - holds parameters sent through HTTP Request