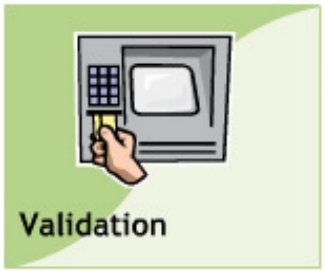
ACCP I7.1 – SEMESTER 3

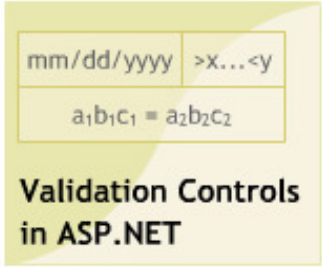
BEGINNING ASP.NET

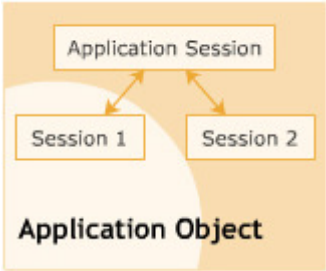
Session 5 – Validation, Cookies, Sessions and Application object

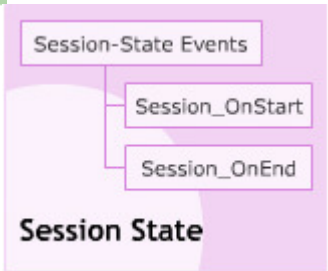
Objectives

- Validation
- Validation Controls
- Application object
- Session state
- Cookies
- Global.asax









Needs For Validation

- Validation is the process of the determining the accuracy of the provided entity.
- Input data needs to be validated before accepted
- Validation of the data determines its correctness in terms of attributes such as type, value, format.

Overview of Validation Controls

BaseValidator Class

- Validation controls can be included on a Web Form and assign them to the specific input Web server controls to validate the values entered by the user.

Name	Description
`ControlToValidate` property	Specifies or retrieves the identifier of the input Web server control to be validated.
`Display` property	Specifies or retrieves the behavior of the error message to be displayed in a validation control. The behavior can be either None, Static, or Dynamic.
`ErrorMessage` property	Specifies or retrieves the text for the error message to be displayed when the validation is not performed.
`IsValid` property	Specifies or retrieves a value indicating whether the related input control passes through the validation process or not.
`Validate()` method	Validates the related input control and modifies the `IsValid` property.

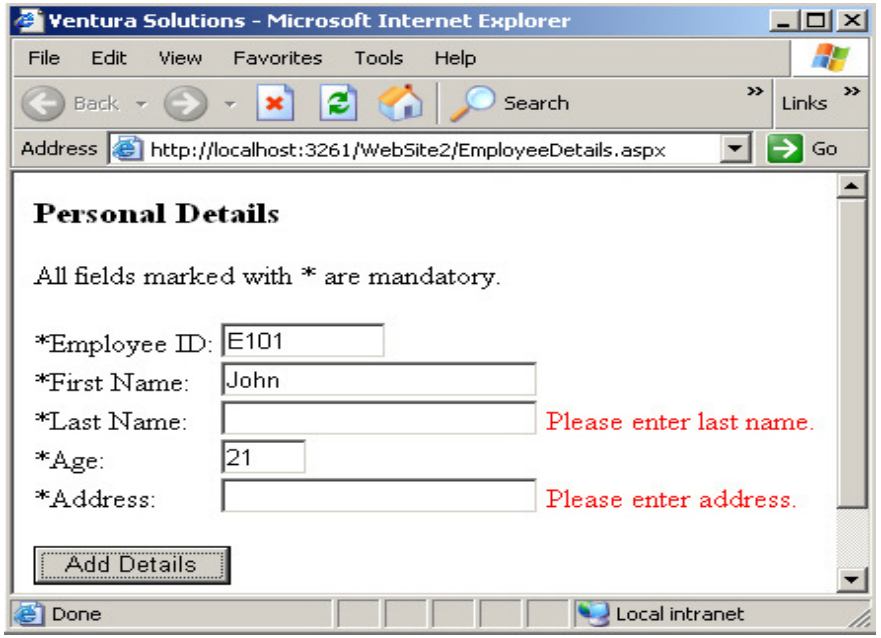
Validation Controls Provided by ASP.NET

Control Name	Description	Example where it can be used
CompareValidator	Compares the value of one input control with another other input control or other constant value based on the comparison operator associated with it.	In Confirm Password field that is used to match the value entered by the user in the Password field.
CustomValidator	Evaluates the value of the input to check if it is valid according to the specified logic.	A control to check whether the value entered is a prime number or not.
RangeValidator	Checks whether the input value is within the specified range limit.	To assign grades to students on the basis of marks scored.
RegularExpressionValidator	Checks if the input value is in the specified format.	To check whether the zip code of a city is entered in the required format.
RequiredFieldValidator	Ensure that the input control is not left blank by the user.	To check whether values are entered in the mandatory fields of a form.
ValidationSummary	Allows reviewing of all error messages received from the validation controls.	To display a list of error messages that occurred during validation of an employee registration.

Validation Controls in ASP.NET

RequiredFieldValidator

- The *RequiredFieldValidator* control ensures that the user enters data in the input control.
- The *RequiredFieldValidator* control is specially used when the user should be forced to enter data in the input control.
- **Properties**
  - *InitialValue*
  - *SetFocusOnError*
  - *Text*



Validation Controls in ASP.NET

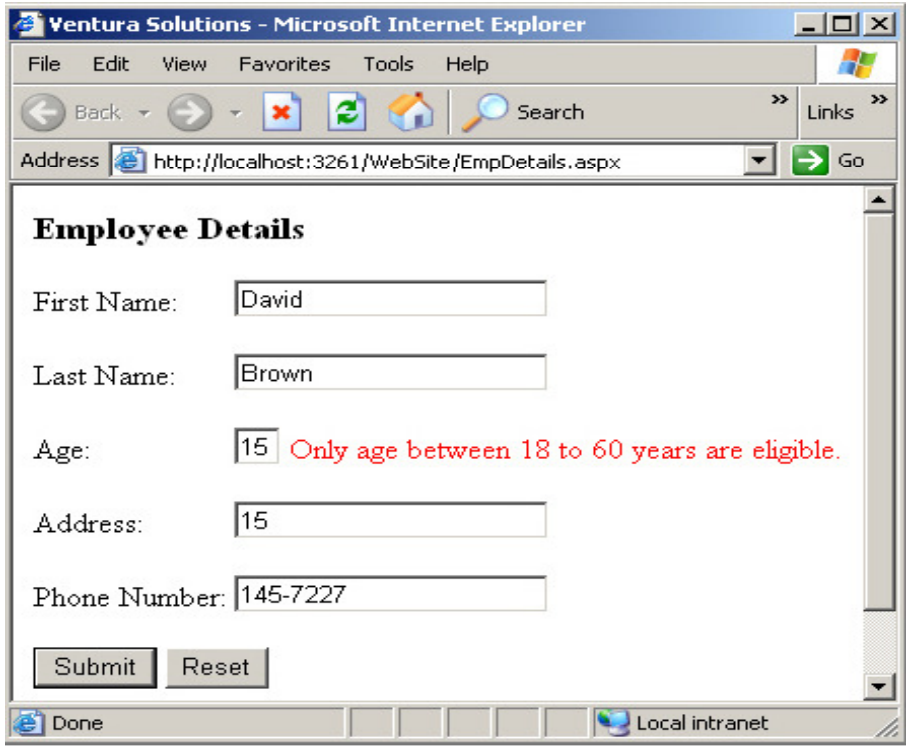
CompareValidator

- The ***CompareValidator*** control compares the value of one input control with another input control or a constant value
- **Properties**
  - Operator
  - ValueToCompare
  - Text

Validation Controls in ASP.Net

RangeValidator

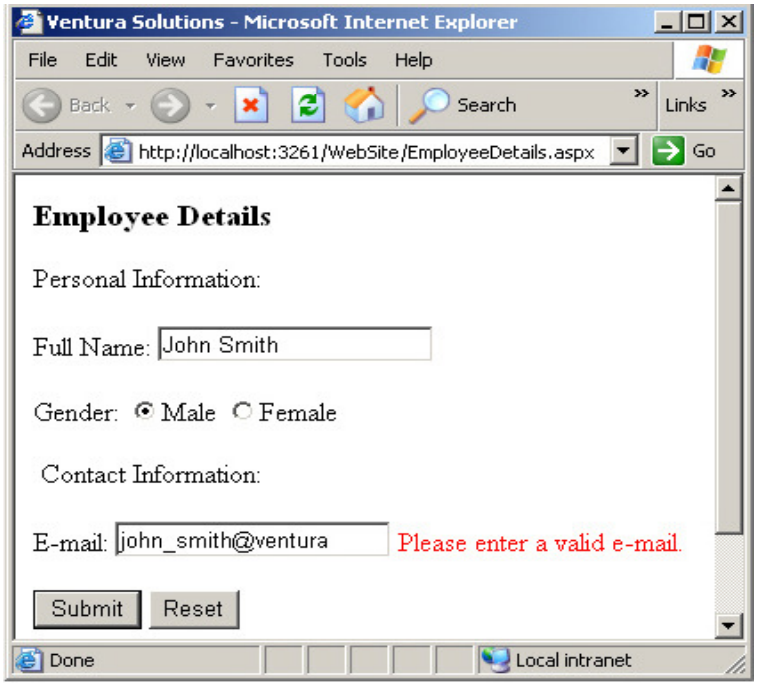
- The *RangeValidator* control checks whether the value provided by the user is within a specified range.
- **Properties**
  - ErrorMessage
  - IsValid
  - Text
  - MaximumValue
  - MinimumValue
  - Type



Validation Controls in ASP.NET

RegularExpressionValidator

- *RegularExpressionValidator* checks for the validation of the value in the input control with a pattern of an expression.
- **Properties**
  - ControlToValidate
  - Display
  - ErrorMessage
  - IsValid
  - ValidationExpression



Validation Controls in ASP.NET

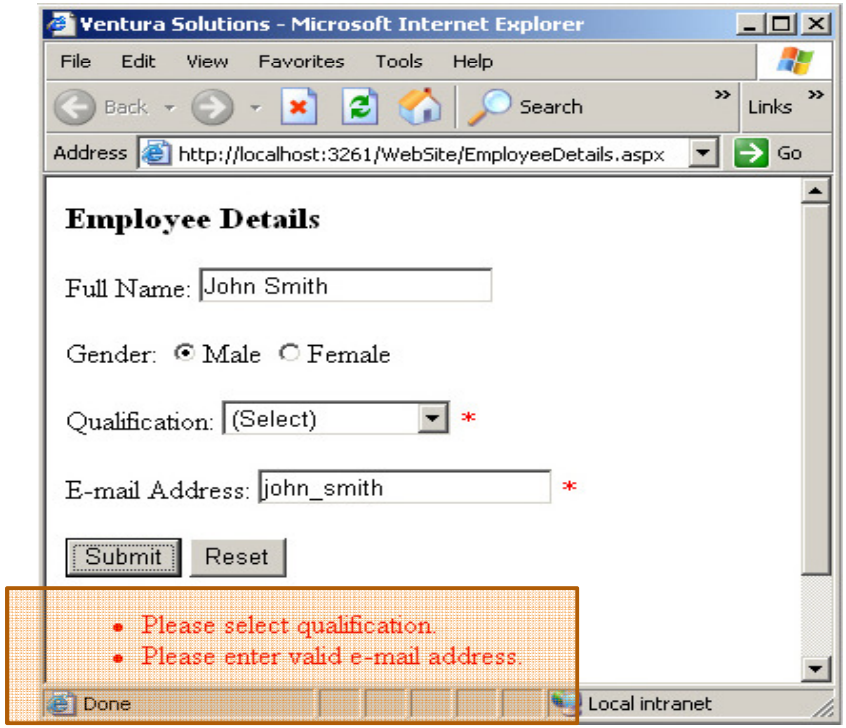
CustomValidator

- *CustomValidator* control checks input data to determine whether the value entered is valid or not according to the specified logic
- **Properties**
  - ControlToValidate
  - ClientToValidate
  - ErrorMessage
  - ValidateEmptyText

Validation Controls in ASP.Net

ValidationSummary

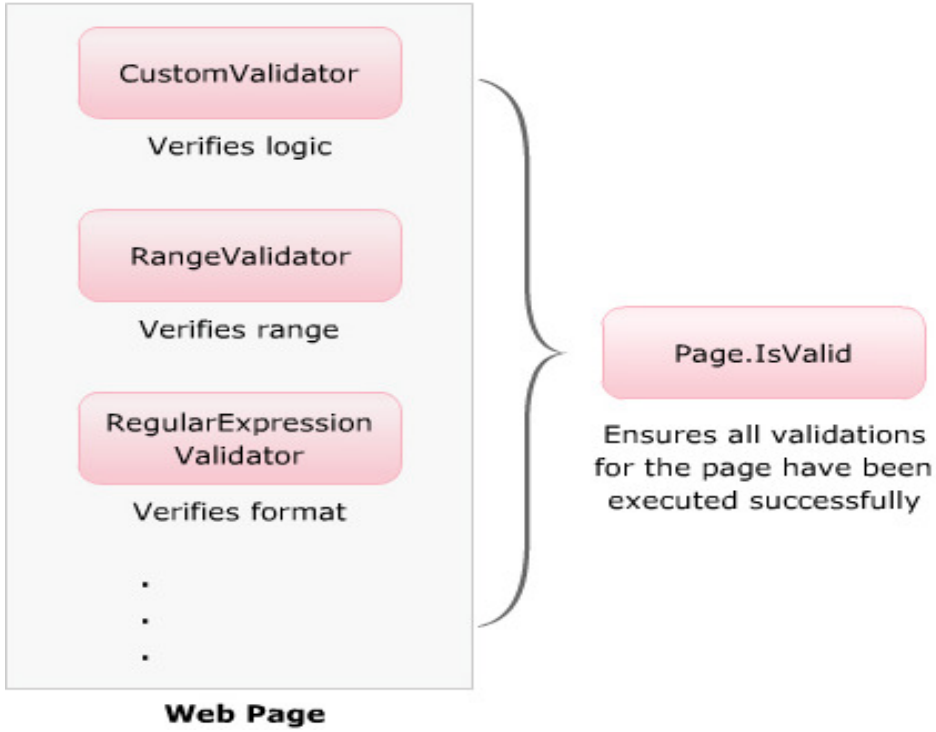
- The *ValidationSummary* control allows users reviewing error messages from all the validation controls on the Web page.
- **Properties**
  - DisplayMode
  - EnableClientScript
  - HeaderText
  - ShowMessageBox
  - ShowSummary



Validation Controls in ASP.NET

Page.IsValid Property

- The *Page.IsValid* property retrieves a value which indicates whether or not the validation of the pages has successful.
- The *Page.IsValid* property retrieves the value as true when all the validation controls within the page are successfully validated.



13

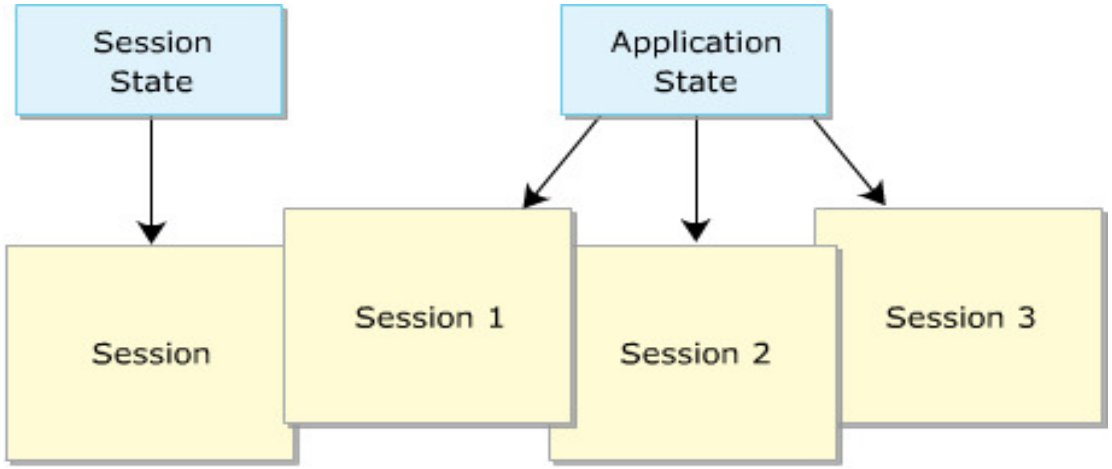
Application, Session and Cookies

'Maintaining states in the stateless world'

Application Object

Introduction

- An application state stores global information used across multiple session and requests.
- The Application object generally holds information that will be used by multiple pages of the application



Application Object

The HttpApplicationState Class

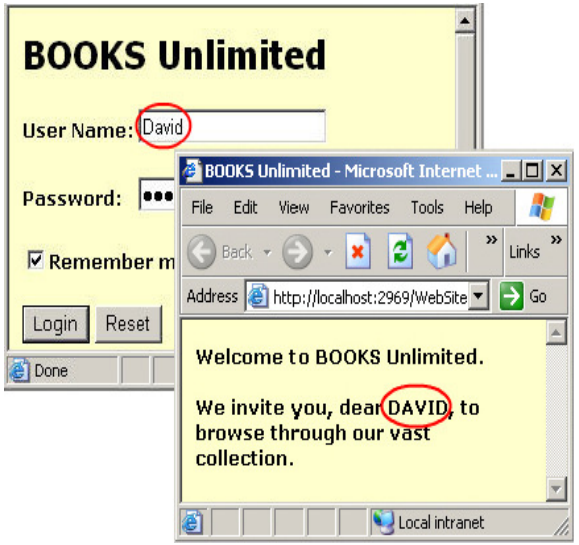
- Application objects are used to reference instances of *HttpApplicationState* class
- The *Lock()* method prevents other users from altering the variables stored in the Application object.
- The *Unlock()* method is used to unlock the locked variables stored in the Application object.

Properties	Method
AllKeys	Add
Contents	Clear
Count	Remove
Item	RemoveAll
StaticObjects	RemoveAt

Cookies

Session Cookies

- Session cookie is also referred to as a transient cookie.
- Session cookies are stored temporarily in the memory. One the browser is closed, the session cookie cannot be retained.



Cookies

Cookies - Example

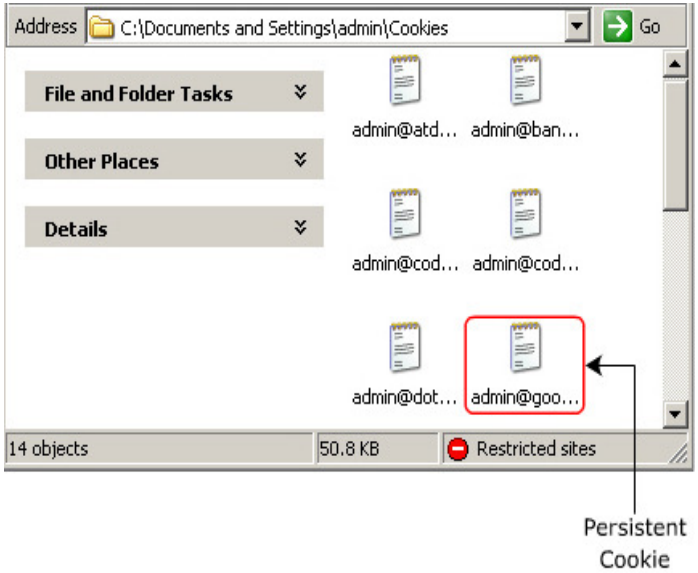

```
protected void btnAdd_Click(object sender, EventArgse)
    {
        Response.Cookies["Login"]["UserName"] =
txtUserName.Text;
    }
```

```
    protected void btnView_Click(object sender, EventArgs e)
    {
        if (Request.Cookies["Login"] == null)
        {
            Response.Write("No cookie.");
        }
        else
        {
            Response.Write("Cookie information: " +
Request.Cookies["Login"]["UserName"]);
        }
    }
```

Cookies

Persistent Cookies

- Cookies storing such information that is remembered across multiple session are referred to as **persistent cookies** (or **permanent cookies** or **stored cookies**).
- They have an *expiry date* and is stored on user's hard disk *until the expiry date* or until they are *manually deleted.*



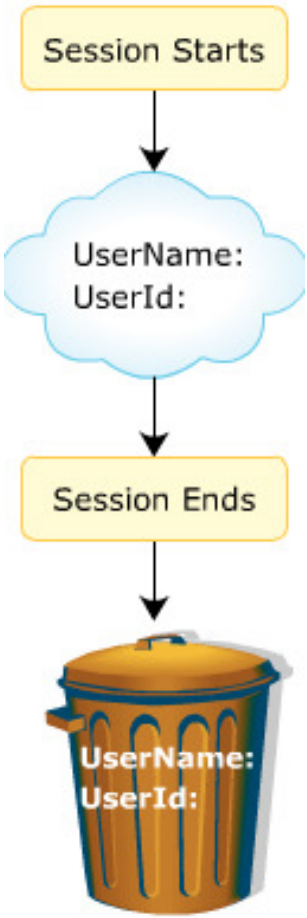
Cookies

Persistent Cookies Creation


```
HttpCookie userInfoCookie = new HttpCookie("UserInfo");
userInfoCookie.Values["UserName"] = "John";
userInfoCookie.Values["LastVisit"] = DateTime.Now.ToString();
userInfoCookie.Expires = DateTime.MaxValue;
Response.Cookies.Add(userInfoCookie);
```

Session Variables (1)

- Session variables are used to store information about a single user session
- Session variables are cleared as soon as the user's session at the site comes to an end
- Sessions are identified by a session identifier, which is unique.
- The sessions identifier can be read using SessionID property.



Session Variables (2)

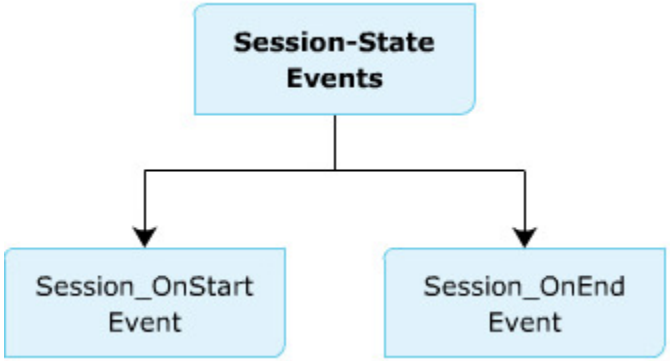
- Session variables are used to store information about a single user session. This information is ***available to all pages*** in the application.
- *Example*:

```
Session ["UserID'"] = "101";
if(Session["UserlD"] =null)
{
  Response.Write{"Page has expired.
}
```

Session Management

Session-State Events

- Handled in Global.asax file for tracking and managing users' session states



Session ID

- Each session has an ID which is unique and can be retrieved by Session.SessionID
- By default SessionID is stored on clients machine using cookies

Global.asax File

- It is called as an ASP.NET *application file* , which contains the code for *responding to application or module-level events* in one central location.
- It is an optional file and created only if the application or session events need to handled

Global.asax File

- While creating ASP.NET application in Visual Studio 2005 IDE , a Global.asax file is automatically added.
- There is no more than one Global.asax file

```
<%@ Application Language="C#" %>
<script runat="server">
void Application_Start(object sender, EventArgs e)
 {
  // Code that runs on application startup
 }
 void Application_End(object sender, EventArgs e)
 {
  //  Code that runs on application shutdown
 }
 void Application_Error(object sender, EventArgs e)
 {
  // Code that runs when an unhandled error occurs
 }
 void Session_Start(object sender, EventArgs e)
 {
  // Code that runs when a new session is started
 }
 void Session_End(object sender, EventArgs e)
 {
  // Code that runs when a session ends.
  // Note: The Session_End event is raised only when
  // the sessionstate mode is set to InProc in the
  // Web.config file. If session mode is set to
  // StateServer or SQLServer, the event is not raised.
 }
</script>
```

Summary – Workshop Activities

- Validation in ASP.NET
- Built-in ASP.NET Validation Controls (RequiredFieldValidator, CompareValidator, etc.)
- Properties and methods of the Application object
- Session states
- Cookies
- Global.asax file