

# Validation



# Agenda

- **Discuss validation in web applications**
  - Client-side validation with javascript
  - Server-side validation
- **Introduce ASP.NET validation architecture**
  - Adding validation to a form
  - Client-side validation
  - Server-side validation
- **Inspect validation controls**
  - Different validation types
  - Displaying validation summaries
  - Performing custom validation
- **Validation groups**

# Form Validation

- **Form validation is commonplace on the web**
  - Data entered by users must be validated before it is used by the server
  - Email addresses need to be validated
  - Phone numbers must be of the correct form
  - Passwords must be entered twice correctly, and so on...

## Sample web validation form

### Account information:

First name:

Last name:

E-mail address:

(e.g. joe@foo.com)

Password:

(4 to 10 characters)

Re-enter password:

### Shipping Address

Company:


Address Line 1:

Address Line 2:

City:

State/Province:

Zip/Postal Code:

Country:  

Day Phone:

← oops!

Please correct the following errors:

- E-mail must be of the form joe@develop.com.
- The two passwords you entered did not match, please reenter
- Day Phone must be filled in.

← oops!

Submit

# Client-side Validation

- **Data validation can occur on the client side using scripting**
  - Reduces round-trips to the server by correcting mistakes before submitting them
  - Provides user with immediate feedback on data input
  - Requires client browser support scripting
  - Can be easily subverted to send bad data to the server anyway (should never use client-side validation without server-side validation too)

## Client-side script validation example

```
<html><head><script language="javascript">
function checkForm() {
    var ret = false;
    if (document.all["cname"].value == "")
        document.all["err_cname"].style.visibility = "visible";
    else if (document.all["email"].value == "")
        document.all["err_email"].style.visibility = "visible";
    else
        ret = true;
    return ret;
}
</script></head>
<form name="SIGNUP" method="post"
        onSubmit="return checkForm()">
<table cellpadding="0" cellspacing="1" border="0">
<tr valign="top"><td align="right"><b>Name:</b></td>
    <td><input id="cname" /></td>
    <td><span id="err_cname"
        style="visibility:hidden;color:red">
        Please enter a name here</span></td></tr>
<tr valign="top"><td align="right"><b>E-mail address:</b></td>
    <td><input id="email" /></td>
    <td><span id="err_email"
        style="visibility:hidden;color:red">
        Please enter your email here</span></td></tr>
</table>
<input type="submit" value="sign up!" />
</form></body></html>
```

# Server-side Validation

- **Data validation should always occur on the server**
  - Before data is actually used on the server, it should be validated
  - Even with client-side validation, validation on the server should still occur as a precaution
  - If an error is encountered, the form should be presented to the user again for corrections

# Validation Observations

- **Some observations about how validation is performed on the web**
  - Error messages often appear adjacent to input elements
  - A list of all errors is often displayed in summary
  - Client-side validation is useful to avoid round trips
  - Server-side validation should always be performed
  - The error message displayed for a given field may change based on the error
  - Field validation may be dependent on the value of another field



# Validation in ASP.NET

- **ASP.NET provides a set of controls to perform validation**
  - Add validation to a form by adding one or more of these controls
  - Display message boxes and summary paragraphs
  - Perform client-side validation when possible
  - Always perform server-side validation

# Adding a validation control to a form

- **To add a validation control to a form:**
  - Add control where the error message should appear
  - ControlToValidate points to control containing data
  - Display field indicates whether control occupies space when valid
  - ErrorMessage contains error string shown in summary
  - Inner text contains string to display at location

```
<asp:TextBox id="foo" runat="server"/>
<asp:RequiredFieldValidator id="fooValidator"
    ControlToValidate="foo"
    Display="Static"
    InitialValue=""
    runat="server"
    ErrorMessage="The foo field must be completed.">***
</asp:RequiredFieldValidator>
```

# ValidationSummary

- **Use validation summary control to display error messages for a page**
  - Message box option available
  - Paragraph or bulleted list formats
- **Harvests error messages of all validation controls on form**
  - Displays ErrorMessage text of each control in message box / paragraph

```
<asp:ValidationSummary id="valSum" runat="server"  
    HeaderText="Please correct the following errors:"  
    ShowMessageBox="True"/>
```

## Validation control rendering

```
<asp:RequiredFieldValidator  
  runat="server"  
  ErrorMessage="Enter a Name"  
  ControlToValidate="_name"  
  Display="Static">  
***  
</asp:RequiredFieldValidator>
```

Renders as

```
<span  
  controltovalidate="_name"  
  errormessage="Enter a Name"  
  evaluationfunction="RequiredFieldValidatorEvaluateIsValid"  
  initialvalue=""  
  style="color:Red;visibility:hidden;">  
***  
</span>
```

```
<asp:RequiredFieldValidator  
  runat="server"  
  ErrorMessage="Enter a Name"  
  ControlToValidate="_name"  
  Display="Dynamic">  
***  
</asp:RequiredFieldValidator>
```

Renders as

```
<span  
  controltovalidate="_name"  
  errormessage="Enter a Name"  
  display="Dynamic"  
  evaluationfunction="RequiredFieldValidatorEvaluateIsValid"  
  initialvalue=""  
  style="color:Red;display:none;">  
***  
</span>
```

# Page Validation

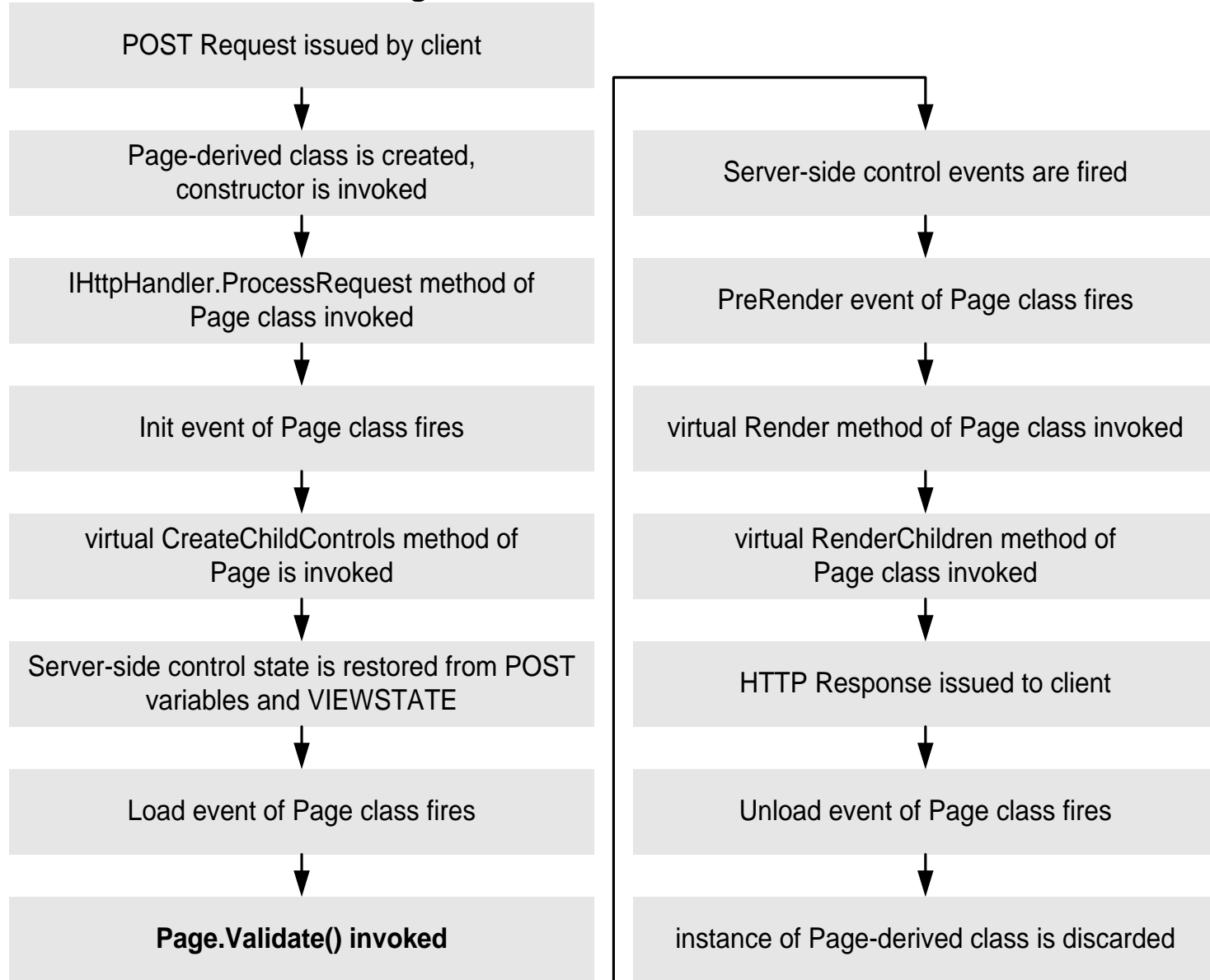
- **Server-side validation managed by Page class**
  - Collection of validation controls in Validators collection
  - Each control implements IValidator
  - Page exposes IsValid property
    - Intersection of all IsValid properties of validation controls
  - Page exposes Validate() method
    - Invokes all Validate() methods of validation controls

```
public interface IValidator
{
    string ErrorMessage {get; set;}
    bool IsValid {get; set;}
    void validate();
}
```

# Server-side Validation

- **Server-side validation happens during the post-back**
  - Initial values of controls are not validated
  - Validation occurs just after Load event
  - On failure, server-side validation controls that fail validation render as visible span elements
    - No other action is taken
  - You must check IsValid of Page before using data

## Server-side Validation During Post-Back

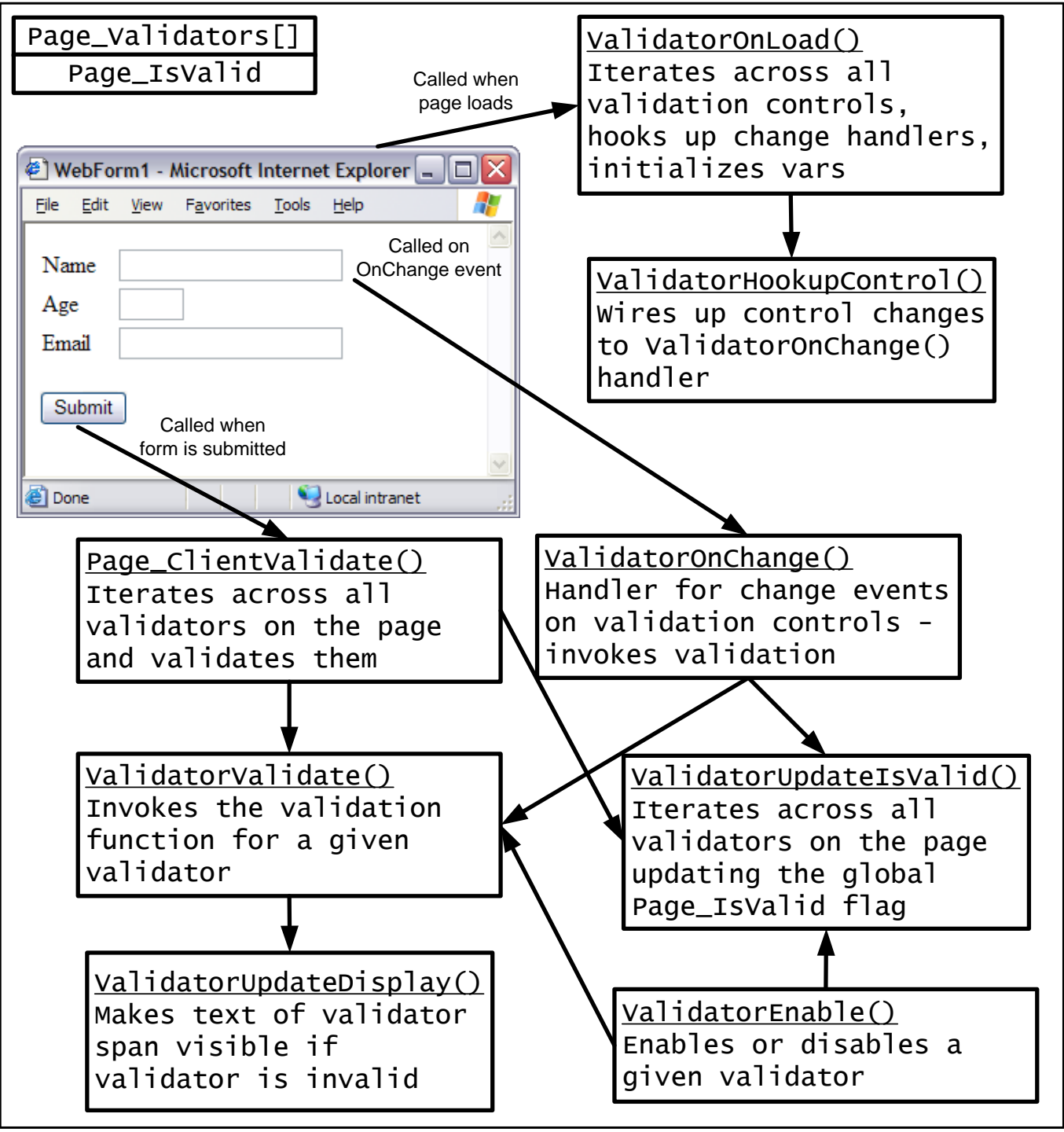


# Client-side Validation

- **Client-side validation is implemented in JavaScript and occurs in up-level browsers only**
  - A JavaScript file (WebUIValidation.js) is read in from the system.web.dll assembly as a resource
  - Each validation control is translated into a span element with custom attributes, and a style of "color:Red;visibility:hidden"
  - A global script variable, `Page_Validators` is set up with all of the span elements created
  - The `ValidatorOnLoad()` function is called at startup to wire up validation handlers
  - Can disable by manually setting the `clienttarget` attribute of the `@Page` directive to "downlevel"



Script elements  
of client-side  
validation



# Controls to Validate

- **Only a subset of the available controls work with validation controls**
  - A control must have a `ValidationPropertyAttribute` to indicate which property to read for validation for it to work with validation controls
  - The `value` attribute of a control must correspond to its value to validate for client-side validation to work

## Controls That Work With Validation

HTMLInputText

HTMLTextArea

HTMLSelect

HTMLInputFile

TextBox

DropDownList

ListBox

RadioButtonList

# Validation Controls

- **Five built-in validation controls in ASP.NET**
  - RequiredFieldValidator - ensures field is not empty
  - CompareValidator - compare one field to another (or a constant)
  - RangeValidator - constrain a field to a range of values
  - RegularExpressionValidator - matched against a RegEx
  - CustomValidator - you supply validation algorithm
- **ValidationSummary control used to display errors for all controls at a common location on the page (or in a dialog)**

## CompareValidator control example

```
<table cellpadding="0" cellspacing="1" border="0">
  <tr>
    <td align="right"><b>Name:</b></td>
    <td><asp:TextBox id="cname" runat="server"/></td>
  </tr>
  <tr>
    <td align="right"><b>Age:</b></td>
    <td><asp:TextBox id="age" runat="server"/></td>
    <td><asp:CompareValidator id="ageValidator"
      ControlToValidate="age"
      ValueToCompare="21"
      Type="Integer"
      Operator="GreaterThanEqual" runat="server">You must be >= 21!
    </asp:CompareValidator></td>
  </tr>
  <tr>
    <td></td>
    <td><input value="Enter" type="submit" /></td>
  </tr>
</table>
```

## RegularExpressionValidator control example

```
<table cellpadding="0" cellspacing="1" border="0">
  <tr>
    <td align="right"><b>Email address:</b></td>
    <td><asp:TextBox id="Email" runat="server"/></td>
    <td><asp:RegularExpressionValidator
      id="EmailRegexValidator"
      ControlToValidate="Email"
      Display="static"
      ErrorMessage="E-mail must be of the form foo@bar.com."
      InitialValue="" width="100%" runat="server"
      ValidationExpression=
        "[\w-]+@[\w-]+\.(com|net|org|edu|mil)$">
    </asp:RegularExpressionValidator></td>
  </tr>
  <tr>
    <td></td>
    <td><input value="Enter" type="submit" /></td>
  </tr>
</table>
```

## Some Regular Expression characters and their meaning

Character	Meaning
[...]	Match any one character between brackets
[^...]	Match any one character not between brackets
.	Match any one character not between brackets
\w	Match any word character [a-zA-Z0-9_]
\W	Match any whitespace character [\t\n\r\f\v]
\s	Match any non-whitespace character [^\t\n\r\f\v]
\d	Match any digit [0-9]
\D	Match any non-digit character [^0-9]
[b]	Match a literal backspace
{n,m}	Match the previous item >= n times, <= m times
{n,}	Match the previous item >= n times
{n}	Match the previous item exactly n times
?	Match zero or one occurrences of the previous item {0,1}
+	Match one or more occurrences of the previous item {1,}
*	Match zero or more occurrences of the previous item {0,}
	Match either the subexpression on the left or the right
(...)	Group items together in a unit
^	Match the beginning of the string
\$	Match the end of the string
\b	Match a word boundary
\B	Match a position that is not a word boundary

## CustomValidator control example

```
<html>
<body>
<form runat="server">
<table cellpadding="0" cellspacing="1" border="0">
  <tr>
    <td align="right"><b>Enter a number:</b></td>
    <td><asp:TextBox id="num" runat="server"/></td>
    <td><asp:CustomValidator id="numValidator"
      ControlToValidate="num"
      Display="static"
      InitialValue=""
      ClientValidationFunction = "MyClientValidationFunction"
      OnServerValidate = "MyServerValidationFunction"
      width="100%" runat="server">Wrong! Try again...
    </asp:CustomValidator></td>
  </tr>
  <tr>
    <td></td>
    <td><asp:Button Text="Enter" runat="server" /></td>
  </tr>
</table>
</form>
</body>
</html>
```

## Custom Validator script blocks

```
<script language="javascript">
<!--
function MyClientValidationFunction(source, args)
{
    if (args.Value % 3)
        args.IsValid=false;
    else
        args.IsValid=true;
}
-->
</script>

<script language="C#" runat="server">
void MyServerValidationFunction(object source,
                               ServerValidateEventArgs e)
{
    e.IsValid = false;
    try
    {
        int num = Convert.ToInt32(e.Value);
        if (num % 3 == 0)
            e.IsValid = true;
    }
    catch (Exception)
    {}
}
</script>
```



# Validation groups

- **Enforce validation on based on events**
- **Indicated via “ValidationGroup” property**
  - Supported by all Validation and Postback controls
  - Controls in ValidationGroup validate with postback
- **Programmatic Support for Validating Groups**
  - If (Page.Validate(“group\_name”)) Then
  - Page.IsValid evaluates ValidationGroup Postback

# Validation groups

```
<asp:TextBox ID="TextBox1" runat="server" />
<asp:RequiredFieldValidator validationGroup="Group1"
    ErrorMessage="Please enter a value"
    ControlToValidate="TextBox1" runat="server" />

<asp:TextBox ID="TextBox2" runat="server" />
<asp:RequiredFieldValidator validationGroup="Group2"
    ErrorMessage="Please enter a value"
    ControlToValidate="TextBox2" runat="server" />

<asp:Button Text="Group1" validationGroup="Group1" runat="server" />
<asp:Button Text="Group2" validationGroup="Group2" runat="server" />
```

# Summary

- **Validation controls in ASP.NET simplify adding validation to pages**
- **Controls perform both client side validation (when possible) and server side validation**
- **Four different types of validation available (plus custom)**
- **Summary control displays errors in a convenient way**
- **Validation groups enable conditional validation**