FIT5032

Internet Applications Development

Week 5A: Validation

Murray Mount / ABM Russel

shutterstock.com • 1285499095

Unit Topics

Week	Activities	Assessment
0		No formal assessment or activities are undertaken in week 0
1A	Intro to Web development and ASP.NET	Note: Studio classes commence in week 1
1B	The front end, user experience, accessibility and ASP.NET Scaffolding	
2	Introduction to C# & Version Control	
3	Entity Framework	
4	Fundamentals of Client side Javascript	Studio assessment task 1 due
5A	Validation	
5B	Security and Identity	
6	Sending Email, File Upload and Signal R	Studio assessment task 2 due
7	Web Optimisations & Evolution of ASP.NET CORE	
8A	Modern JavaScript Web Development Approaches	
8B	Testing and Deployment in Cloud	Studio assessment task 3 due
9	Review & Revision	Final Portfolio and Learning Summary due
	SWOT VAC	No formal assessment is undertaken in SWOT VAC
	Examination period	LINK to Assessment Policy:http://policy.monash.edu.au/policy-bank/academic/education/assessment/assessment-in-coursework-

Today

- Recap: JavaScript
- Validations
- Usability Features

Recap: JavaScript

JavaScript Libraries and Framework

- JavaScript library: pre-written JavaScript which allows for easier development of JavaScript applications. Examples of JavaScript libraries are
  - jQuery, jQueryUI
  - Google Maps Platform, Leaflet.js
  - d3.js
- JavaScript framework: A framework defines the entire application design. Examples of JavaScript frameworks are
  - AngularJS
  - VueJs
  - React

Validation

Why are ASP.Net validators used in web applications

Why are ASP.Net validators used in web applications

A. To make it difficult for the user to use the application

B. To make if difficult to hack the application

C. All of the answers (except none)

D. None of the answers

Validations Concepts

- Input Validation
  - correct format and data type
  - not null fields
  - data types
    - dates, numeric and text fields
  - data is within valid ranges
    - e.g. age (ranges)
  - specific format
    - email addresses or post codes etc.

Validations for Security

- Validation to minimise security issues
    - e.g. code injection attacks
- Server and Client side validation
    - Client side validation is easier to by pass for a knowledgeable user
    - Client side validation is recommended to reduce round trips to the server
    - Server side validation for additional security

Validation in ASP.NET MVC

- Main aspects to implementing validation in ASP.Net MVC Applications
  - Validation in Models
  - Validation in Views
  - Validation Error Messages

Validation in Models

- Models can be annotated to support validation in ASP.Net MVC Applications
    - The field in the model is annotated with the relevant annotation
    - [Required(ErrorMessage = "Please Enter Name")]
- Many useful validation attributes can be found in the **System.ComponentModel.DataAnnotations** namespace.

Built-in Validation Support

- There are many built-in Validation types supported
  - Range Validation
    - [Range(0, 1000, ErrorMessage = "Enter price between 0 to 1000”)]
  - Data Type Validation
    - [DataType(DataType.Date)]
  - Length Validation
    - [StringLength(255, MinimumLength = 8)]
  - Regular Expression Validation
    - [EmailAddress]

Popular Built-in Attribute Validation

Attribute	Functionality
[Compare]	Validates two properties in a model match.
[EmailAddress]	Validates the property has an email format.
[Range]	Validates the property value falls within the given range.
[RegularExpression]	Validates that the data matches the specified regular expression.
[Required]	Makes a property required.
[StringLength]	Validates that a string property has at most the given maximum length
[Url]	Validates the property has a URL format.
[CreditCard]	Validates the property has a credit card format.

Common Regular Expression Syntax

Character	Meaning
^	Matches beginning of input. If the multiline flag is set to true, also matches immediately after a line break character.
$	Matches end of input. If the multiline flag is set to true, also matches immediately before a line break character.
*	Matches the preceding expression 0 or more times. Equivalent to {0,}.
+	Matches the preceding expression 1 or more times. Equivalent to {1,}.
?	Matches the preceding expression 0 or 1 time. Equivalent to {0,1}.
{n}	Matches exactly n occurrences of the preceding expression. N must be a positive integer.

Why do we have Required Field Validators

Why do we have Required Field Validators

A. We shouldn't have null/empty values in web applications

B. There are some fields that can't be null/empty

C. It improves the data quality that is gathered

D. It makes sure that the user knows they must fill in a value

E. All the answers

Why do we have range Validators

Why do we have range Validators

A. To improve the data quality

B. All data in web applications should be within a given range

C. Some data must be between a certain range of values

D. All the answers (except none)

E. None of the answers

Why do we have compare validators

Why do we have compare validators

A. To check two input fields are consistent

B. To check one input field against a given value

C. To improve data quality in web applications

D. All the answers (except none)

E. None of the answers

Why do we have regular expression validators

Why do we have regular expression validators

A. There are many regular expressions that people use in web applications

B. To compare one input control against the pattern in another one

C. To check emails, phone numbers etc.

D. Complex patterns can't be checked using a Compare Validator

E. All the answers

Why do we have custom validators

Why do we have custom validators

A. To combine the functionality of the standard validators into one validation control

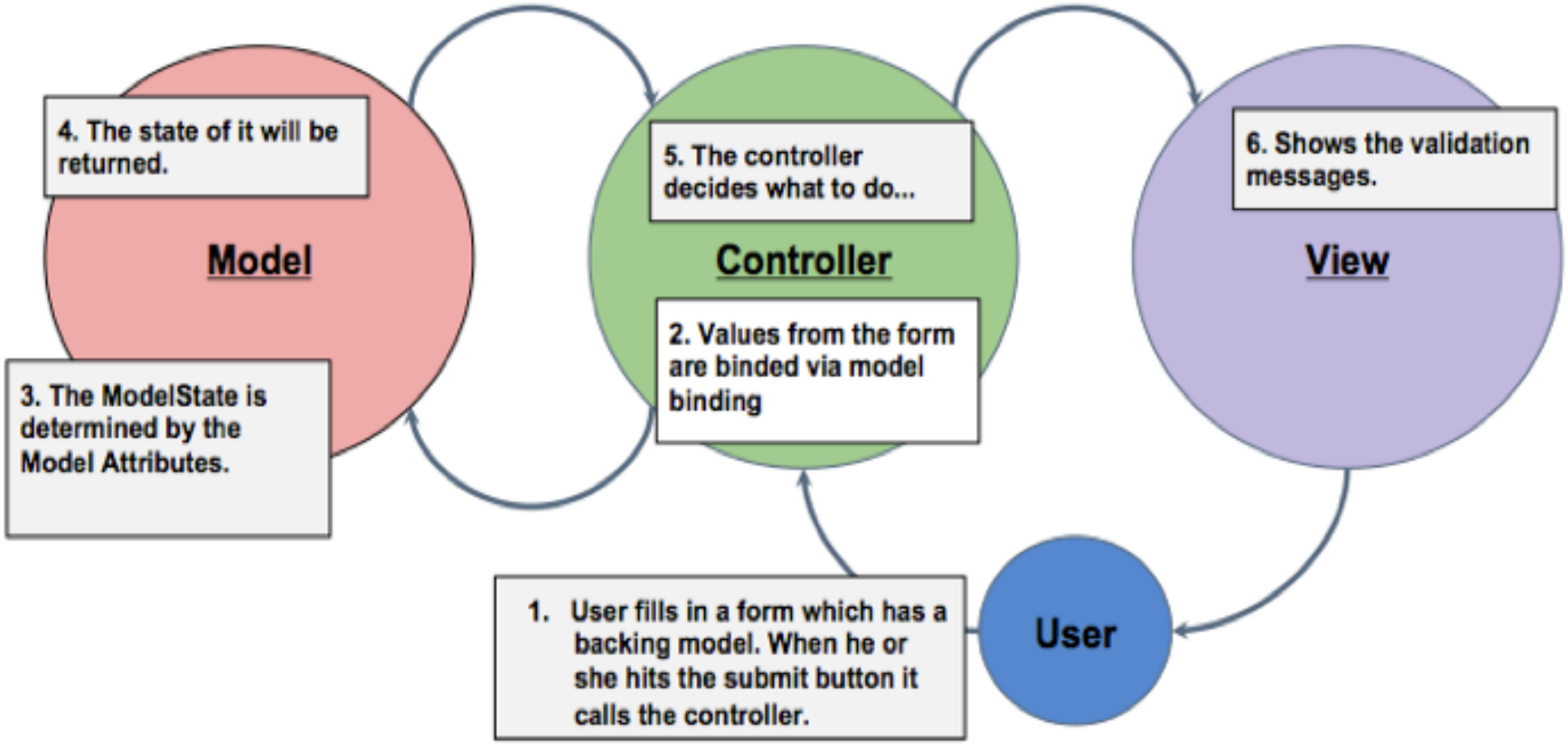
B. To validate Custom data from a web application, ensuring data quality and integrity

C. To create specific validation code that can't easily be done using normal validators

D. So Customer details can be validated

E. All the answers

Model Validation



Validation in Views

- Autogenerated View include the Validation helpers

```
@Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })
```

- A customised Error message can be given

Validation Error Messages

- Standard Error Messages can be added in the Model Annotation

- A customised Error message can be given in the view

```
@Html.ValidationMessageFor(model => model.Name,
"Please Enter a Name", new { @class = "text-danger" }
```

Validation Error Messages (Validation Summary)

- A Summary of the Validation Errors (excluding those already given) can be made

```
@Html.ValidationSummary(true, "", new { @class = "text-danger" })
```

- All the error messages can be summarised by setting excludePropertyErrors = false

```
@Html.ValidationSummary(false, "", new { @class = "text-danger" })
```

How can Standard Error Messages be added in the Model

How can Standard Error Messages be added in the Model

A. @html.ValidationMessageFor(model =>model.Name, "Please Enter a Value", new {@class = “text-danger"})

B. @html.ValidationSummary(true, "", new {@class = “text-danger"})

C. All of the above (except none)

D. None of the above

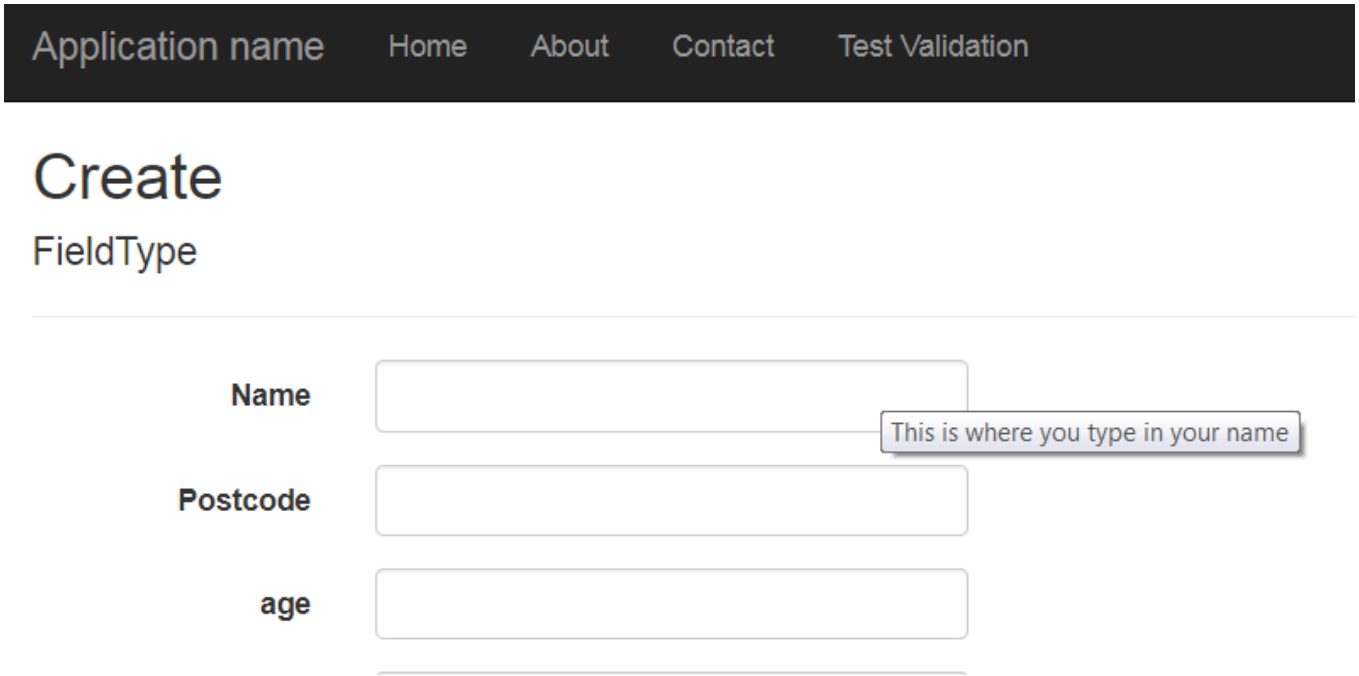
Usability Features

Usability

- Overall Application design motivated by what the user wants or needs
    - Not what the technology can deliver.
- Once User Stories provide the design
    - usability features can further enhance the usability of the application.
- We'll look at features such as
    - tool tips
    - tab index on user input
    - and relevant hot keys.

Tool Tips

- Displayed Text when user hovers over element
  - Implemented by Html title attribute



HtmlAttibutes for Tool Tips

- Html Helper in ASP.Net MVC can take additional parameters (either directly) or via HtmlAttributes specifying pass through parameters for the HTML

```
@Html.TextBox("MyTextbox", new { title = "I'm a Tooltip!”})
```

Or for some Html helpers using the htmlattributes

```
@Html.EditorFor(model => model.Name, new { htmlAttributes =
new { title = "This is where you type in your name"} })
```

Tab Index

- Allows a user to tab through user input fields in a specified order
- Html has a tabindex attribute so:
    - use htmlAttribute with tabindex to the relevant ASP.Net MVC Html helper for the input element

```
@Html.EditorFor(model => model.phone, new { htmlAttributes = new { @class = "form-control", tabindex = 1 } })
```

.

Hot Keys

- Hotkeys allow the user to jump to a specific input element based on the key pressed (e.g. ALT-d)
    - Html has an attribute accesskey so:
    - use htmlAttribute with accesskey to the relevant ASP.Net MVC Html helper for the input element
- For the focus to jump to the date field when the Alt-d keys are pressed (on Windows):

```
@Html.EditorFor(model => model.todaysDate, new { htmlAttributes
= new { @class = "form-control" , accesskey = "d" } })
```

ASP.NET Usability features include

ASP.NET Usability features include

A. Tool tips

B. Tab index

C. Hot keys

D. All of the above (except none)

E. None of the above

Lecture Summary

- Recap: JavaScript
- Validations
- Usability Features

Week 6 Studio Overview

- Data Annotation for ViewModel Validation
- JQuery Unobtrusive Validation
  - jquery.js
  - jquery.validate.js
  - jquery.validate.unobtrusive.js

```
public class FormOneViewModel {
[Required]
[Display(Name = "First Name")]
public string FirstName { get; set; }
public string LastName { get; set; } }
```

Next week: Validations

- Security and Identity