CONESTOGA

COURSE OUTLINE

Course Programming: Microsoft Web Technologies (2015-2016)

Code / Version PROG2230 (103)

Total Hours 75

Credits 5

PreRequisite(s) INFO1570 (100) Technology Infrastructure: Fundamentals
or PROG1245 (100) Programming: Web Foundations
or PROG1247 (100) IT Support
and PROG1780 (100) Programming: Fundamentals
or PROG1783 (100) IT Support Prog Fundamentals
or PROG8010 (100) Programming Software Dev Tech

CoRequisite(s)

Course Description

This course teaches students how to develop web applications using ASP.NET. Students learn how to create dynamic web pages using data from a relational database and how to update data on a relational database with information provided by the user through a web form. The students will use application-wide variables, code and style sheets, object-oriented constructs, classes, and session variables.

PLAR Eligible: Yes

Course Outcomes

Successful completion of this course will enable the student to:

1. Write ASP.NET web applications using C# and Microsoft's MVC techniques.
2. Generate models from an existing database and annotate using metadata classes applied to partial classes.
3. Generate views for a controller action and adapt as required for the application.
4. Use HTML Helpers to speed view development and improve adaptability.
5. Validate and edit user data using annotations and self-validating models.
6. Use localization resource files to translate field-names, error messages and text on a page.
7. Access and maintain records in a relational database using Entity Framework and LINQ.
8. Secure access to portions of a web site using user names, roles and controller annotations.
9. Submit asynchronous requests to modify individual fields on a page
10. Use an automated Unit Test facility to verify that the code functions to requirements.

Essential Employability Skills addressed in this course			Taught	Reinforced	Assessed
Communication	n	Communicate clearly, concisely and correctly in the written, spoken, and visual form that fulfills the purpose and meets the needs of the audience			
	n	Respond to written, spoken, or visual messages in a manner that ensures effective communication			
Numeracy	n	Execute mathematical operations accurately			
Critical Thinking and Problem Solving	n	Apply a systematic approach to solve problems	X	X	X
	n	Use a variety of thinking skills to anticipate and solve problems	X	X	X
Information Management	n	Locate, select, organize, and document information using appropriate technology and information systems	X	X	X
	n	Analyze, evaluate, and apply relevant information from a variety of sources		X	

CONESTOGA

COURSE OUTLINE

Course Programming: Microsoft Web Technologies (2015-2016)

Code / Version PROG2230 (103)

Essential Employability Skills addressed in this course			Taught	Reinforced	Assessed
Interpersonal	n	Show respect for the diverse opinions, values, belief systems, and contributions of others			
	n	Interact with others in groups or teams in ways that contribute to effective working relationships and the achievement of goals			
Personal	n	Manage the use of time and other resources to complete projects		X	
	n	Take responsibility for one's own actions, decisions, and consequences		X	

Unit Outcomes

Successful completion of the following units will enable the student to:

1.0 ASP.NET MVC Overview

- 1.1 Decompose a URL in MVC, identifying how the controller, method and parameters are specified.
- 1.2 State the purpose of the controller, the view and the model.
- 1.3 Use ViewData and ViewBag to convery data from a controller to a view.
- 1.4 Use the Model property in ViewData to pass a strongly-typed object to a view.
- 1.5 Create an ASP.NET MVC project.
  - 1.5.1 Associate the project with a database.
  - 1.5.2 Generate the Entity Framework models for the database.
  - 1.5.3 Generate a controller with list, insert, update and delete views for a model.

2.0 Controllers

- 2.1 Examine the structure of a controller: naming conventions, base class, actions/methods
- 2.2 Create and modify entries in ViewData and ViewBag
- 2.3 Display ViewData values on a view.
- 2.4 Create a simple controller and views to examine the relationship between the URL and the controller actions.
- 2.5 Pass data to a controller's action using QueryString variables
- 2.6 Use HTML encoding to sanitize data before displaying it.
- 2.7 Pass parameters to a controller action in the URL

3.0 Views

- 3.1 Modify or add menu items to provide access to controllers.
- 3.2 Generate a view from a controller action, using the scaffolding templates.
- 3.3 Invoke a view from a controller using convention or by directing it to a specific view.
- 3.4 Display data on a view that is passed using ViewData or ViewBag
- 3.5 Use the Model property of ViewData to create strongly-types views.
- 3.6 Create a view model to convey more diverse information than the data models.
- 3.7 Use Razor syntax to write C# code blocks in a view and to access data passed in ViewData or ViewBag.
  - 3.7.1 Understand how Razor determines when to transition from markup to code and back again
  - 3.7.2 Describe how Razor's HTML encoding fights XSS and how/when to override it.
- 3.8 Use a common layout for all pages on a site.
  - 3.8.1 Identify where, in the layout, guest site content will be inserted.

3.8.2 Override the standard site layout for a web page.

3.8.3 Use C# code blocks and ViewBag to adapt layout content to differentiate individual pages.

3.8.4 Design named sections on a layout for specialized content from guest pages that are mandatory or optional.

4.0 Entity Framework Models

4.1 Write like-named partial classes to isolate generated models from their annotations.

4.2 Examine the interaction of Entity Framework when scaffolding a controller.

4.3 Identify the primary key property in a model using Entity Framework's conventions.

4.4 Examine how data passed by a controller influences how that is conveyed to the user.

4.4.1 Create a SelectList collection in ViewBag and display it as a drop-down in the view.

4.4.2 Display existing database values on a view.

4.4.3 Use MVC conventions to convey data seamlessly from controller to view and back again on user post-back.

4.5 Determine how a controller differentiates a user's request for a new page and a page being posted back with the user's data.

4.5.1 Use the default Binder application to map data from a view to the parameters required by a controller.

4.5.2 Declare parameters to a controller as optional with a default value or null-able.

4.5.3 Query ModelState to determine if data provided to a controller action is valid.

4.5.4 Rebuild a view with the user's data and error messages when the model's data is not valid.

5.0 Use HTML Helpers (code-snippet generators) on a view to produce adaptable HTML code and controls.

5.1 Determine when to use POST or GET.

5.1.1 Evaluate the advantages and disadvantages with regard to URL recall and replaying an update.

5.2 Write HTML to interact with and pass parameters to a controller's action.

5.2.1 Use an HTML Helper to specify the route to a specific controller's action.

5.3 Use HTML Helpers to:

5.3.1 Control Get/Post

5.3.2 Display and filter validation errors.

5.3.3 Link data from a controller action to controls that present data as HTML labels, drop-downs, text boxes, validation messages, text areas, hidden/password testboxes, radio buttons, checkboxes and list-boxes.

5.4 Use conventions when using helpers to assist model binding.

5.4.1 Reach inside ViewData items to display their property values.

5.4.2 Use lambda expressions to specify strongly-typed model properties to be rendered

5.4.3 Use helpers to extract and display model metadata

5.5 Use HTML Helpers to determine the type of input element to display, based on a property's data type and annotations.

5.6 Use helpers to generate routing links on a page.

6.0 Data Annotations

6.1 Use annotations in a data model to:

6.1.1 Make a field required.

6.1.2 Specify a maximum and minimum field length

6.1.3 Check a value against a regular expression pattern

6.1.4 Check a value to a specified range

6.1.5 Validate a value against the database.

6.1.6 Compare two fields

6.2 Customize error messages from annotations in a model

6.3 Decompose model state to retrieve the user's input, the errors associated with each property and errors with the model itself.

- 6.4 Write controller code to recognize when a model has errors.
- 6.5 Use annotations to access centralized custom validation logic.
  - 6.5.1 Draft validations to support default and custom error messages.
  - 6.5.2 Create a centralized class library to validate or modify data and to provide common data access routines.
- 6.6 Write a self-validating model.
- 6.7 Use annotations to modify how data & when data is displayed:
  - 6.7.1 Display name and display order
  - 6.7.2 ScaffoldColumn to hide properties during edit.
  - 6.7.3 DisplayFormat to conditionally format a property
  - 6.7.4 Force s property into ReadOnly
  - 6.7.5 Force a DataType, to cause the view to adapt
  - 6.7.6 Hide input

7.0 Authenticate Users, Control Access and Secure the Site

- 7.1 Secure a controller or an action in a controller:
  - 7.1.1 To only users that are logged in [Authorize]
  - 7.1.2 To users that are logged in as members of a specific role: [Authorize (Roles="xxx")]
- 7.2 Interact with the registration and login facilities in the site
- 7.3 Use external providers to authenticate users.
  - 7.3.1 Identify and assess attack vectors these open up.
  - 7.3.2 Implement SSL to secure communication with the user & external validators.
- 7.4 Describe how passive and active injection in Cross-Site Scripting attack a web site.
  - 7.4.1 Describe actions to prevent or reduce damage from an XSS attack
- 7.5 Describe how Cross-Site Request Forgery attacks a site and the vectors that it opens.
  - 7.5.1 Describe how social engineering factors into this attack
  - 7.5.2 Implement anti-forging tokens, POST for database changes and a HTTP Referrer authorization attribute.
- 7.6 Describe the threat vector and preventive actions for other security faults:
  - 7.6.1 Cookie theft
  - 7.6.2 Over-Posting
  - 7.6.3 Open Redirection
  - 7.6.4 Custom errors, stack trace & debug left in development mode.

8.0 jQuery

- 8.1 Structure a simple jQuery function
  - 8.1.1 Identify the selectors available to jQuery and contrast these to CSS3 selectors
- 8.2 Implement an AJAX action link to replace one or more HTML elements with an asynchronous response from the server.
  - 8.2.1 Return a partial view or plain text
  - 8.2.2 Describe how the data-dash attribute is used in unobtrusive jQuery

9.0 Unit Testing

- 9.1 Use test-drive development to incrementally test and develop code.
- 9.2 Use refactoring to clean up tested code without breaking the unit tests.
- 9.3 Structure test methods using the "Act, Arrange, Assert" discipline.
  - 9.3.1 Identify when a test is testing more than one behavior and break it into separate behavior tests.

CONESTOGA

COURSE OUTLINE

Course Programming: Microsoft Web Technologies (2015-2016)

Code / Version PROG2230 (103)

Required Student Resources

Optional Student Resources

Galloway, Wilson, Allen, Matson. Professional ASP.NET MVC 5 (2014). Wrox.

Microsoft Visual Studio 2013 Pro or Ultimate, Update 4 or 5 (free via MSDN-AA at dreamspark.com)

Software: Microsoft SQL Server 2014 Express edition (free from http://www.microsoft.com/express/Database/ or http://dreamspark.com)

Evaluation

The minimum passing grade for this course is 55 (D).

In order to successfully complete this course, the student is required to meet the following evaluation criteria:

Assignments (8 @ 5%)	40.00
Mid-Term Exam	30.00
Final Exam	30.00
	100.00 %

Other

Conestoga College is committed to providing academic accommodations for students with documented disabilities. Please contact the Accessibility Services Office.

The policies and procedures in the Conestoga College Student Guide and in the IT Programs' Student Handbook apply to this course.

Prepared By David Turton

School Information Technology

Date 2015-06-16