70-486: Developing ASP.NET MVC Web Applications

The following tables show where changes to exam 70-486 have been made to include updates that relate to MVC 5, Visual Studio 2013, and Windows Azure tasks. These changes are effective as of April 30, 2014.

1. Design the application architecture

Tasks currently measured	Tasks Added/Changed post *April 2014*
Plan the application layers Plan data access; plan for separation of concerns; appropriate use of models, views, and controllers; choose between client-side and server side processing; design for scalability	No Change
Design a distributed application Design a hybrid application (on premise vs. off premise, including Windows Azure); plan for session management in a distributed environment; plan web farms	No Change
Design and implement the Windows Azure role life cycle Identify and implement Start, Run, and Stop events; identify startup tasks (IIS configuration [app pool], registry configuration, third-party tools)	No Change
Configure state management Choose a state management mechanism (in-process and out of process state management); plan for scalability; use cookies or local storage to maintain state; apply configuration settings in web.config file; implement sessionless state (for example, QueryString)	Modified subtask: Choose a state management mechanism (in-process and out of process state management, ViewState)
Design a caching strategy Implement page output caching (performance oriented); implement data caching; implement HTTP caching	Added subtask: Implement Azure caching
Design and implement a Web Socket strategy	Added subtask: Implement SignalR

Read and write string and binary data asynchronously (long-running data transfers); choose a connection loss strategy; decide a strategy for when to use Web Sockets	
Design HTTP modules and handlers Implement synchronous and asynchronous modules and handlers; choose between modules and handlers in IIS	No Change

2. Design the user experience

Tasks currently measured	Tasks Added/Changed post *April 2014*
Apply the user interface design for a web application Create and apply styles by using CSS; structure and lay out the user interface by using HTML; implement dynamic page content based on a design	No Change
Design and implement UI behavior Implement client validation; use JavaScript and the DOM to control application behavior; extend objects by using prototypal inheritance; use AJAX to make partial page updates; implement the UI by using JQuery	No Change
Compose the UI layout of an application Implement partials for reuse in different areas of the application; design and implement pages by using Razor templates (Razor view engine); design layouts to provide visual structure; implement master/application pages	No Change
Enhance application behavior and style based on browser feature detection Detect browser features and capabilities; create a web application that runs across multiple browsers and mobile devices; enhance application behavior and style by using vendor-specific extensions, for example, CSS	No Change
Plan an adaptive UI layout Plan for running applications in browsers on multiple devices (screen resolution, CSS, HTML); plan for mobile web applications	No Change

3. Develop the user experience

Tasks currently measured	Tasks Added/Changed post *April 2014*
Plan for search engine optimization and accessibility Use analytical tools to parse HTML; view and evaluate conceptual structure by using plugs-in for browsers; write semantic markup (HTML5 and ARIA) for accessibility, for example, screen readers	No Change
Plan and implement globalization and localization Plan a localization strategy; create and apply resources to UI including JavaScript resources; set cultures; create satellite resource assemblies	No Change
Design and implement MVC controllers and actions Apply authorization attributes and global filters; implement action behaviors; implement action results; implement model binding	Modified subtask: Apply authorization attributes, global filters, and authentication filters Added subtask: Specify an override filter
Design and implement routes Define a route to handle a URL pattern; apply route constraints; ignore URL patterns; add custom route parameters; define areas	No Change
Control application behavior by using MVC extensibility points Implement MVC filters and controller factories; control application behavior by using action results, viewengines, model binders, and route handlers	No Change
Reduce network bandwidth Bundle and minify scripts (CSS and JavaScript); compress and decompress data (using gzip/deflate; storage); plan a content delivery network (CDN) strategy, for example, Windows Azure CDN	No Change

4. Troubleshoot and debug web applications

Tasks currently measured	Tasks Added/Changed post *April 2014*
Prevent and troubleshoot runtime issues Troubleshoot performance, security, and errors; implement tracing, logging (including using attributes for logging), and debugging (including IntelliTrace); enforce conditions by using code contracts; enable and configure health monitoring (including Performance Monitor)	No Change
Design an exception handling strategy Handle exceptions across multiple layers; display custom error pages using global.asax or creating your own HTTPHandler or set web.config attributes; handle first chance exceptions	No Change
Test a web application Create and run unit tests, for example, use the Assert class, create mocks; create and run web tests	Modified subtask: Create and run web tests (including using Browser Link) Added subtask: Debug a web application in multiple browsers and mobile emulators
Debug a Windows Azure application Collect diagnostic information by using Windows Azure Diagnostics API Implement on demand vs. scheduled; choose log types, for example, event logs, performance counters, and crash dumps; debug a Windows Azure application by using IntelliTrace and Remote Desktop Protocol (RDP)	Modified subtask: Debug a Windows Azure application by using IntelliTrace, Remote Desktop Protocol (RDP), and remote debugging Added subtask: Interact directly with remote Windows Azure websites using Server Explorer

5. Design and implement security

Tasks currently measured	Tasks Added/Changed post *April 2014*
Configure authentication Authenticate users; enforce authentication settings; choose between Windows, Forms, and custom authentication; manage user session by using cookies; configure membership providers; create custom membership providers	Added subtask: Configure ASP.NET Identity
Configure and apply authorization Create roles; authorize roles by using configuration; authorize roles programmatically; create custom role providers; implement WCF service authorization	No Change
Design and implement claims-based authentication across federated identity stores Implement federated authentication by using Windows Azure Access Control Service; create a custom security token by using Windows Identity Foundation; handle token formats (for example, oAuth, OpenID, LiveID, and Facebook) for SAML and SWT tokens	Modified subtask: Handle token formats (for example, oAuth, OpenID, Microsoft Account, Google, Twitter, and Facebook) for SAML and SWT tokens
Manage data integrity Apply encryption to application data; apply encryption to the configuration sections of an application; sign application data to prevent tampering	No Change
Implement a secure site with ASP.NET Secure communication by applying SSL certificates; salt and hash passwords for storage; use HTML encoding to prevent cross-site scripting attacks (ANTI-XSS Library); implement deferred validation and handle unvalidated requests, for example, form, querystring, and URL; prevent SQL injection attacks by parameterizing queries; prevent cross-site request forgeries (XSRF)	No Change