70-486: Developing ASP.NET MVC Web Applications

Candidates for this exam are professional developers who use Microsoft Visual Studio 20122015 7 and Microsoft .NET FrameworkASP.NET to design and develop web solutions. Candidates should have a minimum of three to five years of experience developing Microsoft ASP.NET MVC–based solutions and knowledge of Microsoft Azure Web Apps.

Candidates should also have the following experience:

- Designing and developing web applications in an ASP.NET MVC model
- Planning and designing user interaction solutions based on business requirements
- Experience with the full software development life cycle of web applications
- Developing and deploying to multi-tier environments, including Azure
- Designing and developing asynchronous solutions

Objective Domain

Note: This document shows tracked changes that are effective as of January 15, 2018.

Design the application architecture (15-20%)

Plan the application layers

Plan data access; plan for separation of concerns,; appropriate use of models, views, controllers, view components, and view service dependency injection; choose between client-side and server-side processing; design for scalability; ; choose between ASP.NET Core and ASP.NET; choose when to use .NET standard libraries; understand differences between DNX and DNX core

Design a distributed application

Design a hybrid application; plan for session management in a distributed environment; plan web farms; run Microsoft Azure services on-premises with Azure Pack; enable deferred processing through Azure features (including queues, scheduled, and and on-demand jobs, Azure Functions, and Azure Web Jobs, etc.).)

Design and implement the Azure Web Apps life cycle

Identify and implement Start, Run, and Stop events are triggered; code against application events in your applications; identify configure startup tasks, including (IIS, app pool configuration, and [app pool], third-party tools)

Configure state management

Choose a state management mechanism including (in-process and , out of process, and Redis--based state management); plan for scalability; use cookies or local storage to maintain state; apply configuration settings in web.config files; implement sessionless state (including for example, Qquery sStrings;), configure middleware to enable session and application state in ASP.NET Core)

Design a caching strategy.

Implement page output and donut caching (performance oriented)caching and; implement data caching; create cache profiles; implement HTTP caching; implement Azure Redis caching; plan a content delivery network (CDN) strategy, for example, Azure CDN

Design and implement a Web Socket strategy

Read and write string and binary data asynchronously; choose a connection loss strategy; decide when to use Web Sockets; implement SignalR; enable web socket features in an Azure Web App instance

~~Design HTTP modules and handlers~~

~~Implement synchronous and asynchronous modules and handlers, choose between modules and handlers in IIS~~

Design a configuration management solution

Manage configuration sources, including XML, JSON, and INI files; manage environment variables; implement Option objects; implement multiple environments using files and hierarchical structure; manage sensitive configuration; react to runtime configuration changes; implement a custom configuration source; secure configuration by using Azure Key Vault; use the Secret Manager tool in development to keep secrets out of your code for configuration values

Interact with the host environment

Work with file system using file providers; work with environment variables; determine hosting environment capabilities; implement native components, including PInvoke and native dependencies for hosts including Linux and Windows; use ASP.NET hosting on an Open Web Interface for .NET (OWIN)-based server

Compose an application by using the framework pipeline

Add custom request processing modules to the pipeline; add, remove, and configure services used in the application; design and implement middleware; design for kestrel, Http.sys web server and IIS; design and implement startup filters

Design the build and deployment architecture (10-15%)

Design a browser artifact build strategy

Design a JavaScript build pipeline using Gulp, Grunt, npm and Bower; design an artifact build strategy using Less, Sass and Font Awesome; design and implement a bundling and minification strategy for broswer artifacts, including JavaScript, CSS and images

Design a server build strategy

Manage NuGet dependencies; target runtimes, including the full .NET Framework, .NET core, and .NET standard; manage debug and release configurations, including compilation and optimization options; include or exclude files from build; manage build sources, including content, resources, and shared files; implement metadata for projects, including version, release notes, and descriptions; define other build options, including xmlDoc and warningsAsErrors; work with static files in ASP.NET core

Design a publishing strategy

Implement application publishing using dotnet.exe; manage publishing options in csproj; implement additional tooling; implement pre-publish and post-publish scripts; implement native compilation; publish to Docker container image

Implement an Azure deployment strategy

Deploy Azure Web App using supported deployment models including FTP, Kudu, Web Deploy, and Visual Studio Publishing Wizard; provision ARM-based resources while deploying applications; implement deployment environments, including dev, test, and prod in Azure; use deployment slots for staging sites; deploy to Azure Stack

Implement a on-premises deployment strategy

Deploy application to IIS using Web Deploy, xcopy, and Visual Studio Publishing Wizard; deploy application to Windows Nano Server, deploy application to IIS Hosted Web Core, deploy application to HTTP.sys web server; deploy application to Kestrel on Windows and Linux; implement reverse proxying to Kestrel using IIS and Nginx

Design the User Experience (~~20~~15-~~25~~20%)

Create elements of the user interface for a web application

Create and apply styles by using CSS; structure and lay out the user interface by using HTML~~, including HTML5~~; implement dynamic page content based on a design~~;~~ ~~use client framework package management; leverage CSS processors to simplify application style maintenance~~

Design and implement UI behavior

Implement client-side ~~forms~~ validation; use JavaScript to manipulate the DOM; extend objects by using prototypal inheritance; use AJAX to make partial page updates~~; implement the UI by using JQuery~~

~~Compose~~ Design the UI layout of an application

Implement partial views and view components for reuse in different areas of the application; design and implement pages by using Razor ~~syntax~~Pages; design and implement layouts to provide visual structure; define and render optional and required page sections; create and use tag and HTML helpers to simplify markup

~~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~~

~~Plan an adaptive UI layout~~

~~Plan for running applications in browsers on multiple devices (screen resolution, CSS, HTML), plan for mobile web applications~~

Plan a responsive UI layout

Plan for applications that run on multiple devices and screen resolutions; use media queries and Bootstrap's responsive grid; detect browser features and capabilities; create a web application that runs across multiple browsers and mobile devices; enable consistent cross-browser experiences with polyfills

Plan mobile UI strategy

Implement mobile specific UI elements such as touch input, low bandwidth situations, and device orientation changes; define and implement a strategy for working with mobile browsers

Develop the User Experience (15-20%)

Plan for search engine optimization and accessibility

Use analytical tools to parse HTML; provide an xml sitemap and robots.txt file to improve scraping; write semantic markup for accessibility, for example, screen readers; use rich snippets to increase content visibility

Plan and implement globalization and localization

Plan a localization strategy; create and apply resources to UI including JavaScript resources; set cultures; implement server side localization and globalization ~~create satellite resource assemblies~~

Design and implement MVC controllers and actions

Apply authorization attributes, ~~global~~ filters ~~(e.g., including~~ global, ~~, and~~ authentication, and overriddable filters~~); specify an override filter~~; ~~use~~ choose and implement custom HTTP ~~codes~~status codes and responses; implement action results; implement MVC areas; implement Dependency Injection for services in controllers~~; implement model and property binding; use best practices in model binding~~

Design and implement routes

Define a route to handle a URL pattern; apply route constraints; ignore URL patterns; add custom route parameters; define areas; define routes that interoperate with Single Page Application frameworks such as Angular~~;~~

Control application behavior by using MVC extensibility points~~`~~

Create custom middleware and inject it into the pipeline; implement MVC filters and controller factories; control application behavior by using action results, model binders, and route handlers; inject services into a view

~~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, Azure CDN~~

Design and implement serialization and model binding

Serialize models and data using supported serialization formats, including JSON, XML, protobuf, and WCF/SOAP; implement model and property binding, including custom binding and model validation; implement web socket communication in MVC; implement file uploading and multipart data; use AutoRest to build clients

Troubleshoot and Debug Web Applications (20-25%)

Prevent and troubleshoot runtime issues

Troubleshoot performance, security, and errors;
implement tracing, logging, and debugging ~~(~~including IntelliTrace~~)~~; enable and configure health monitoring ~~(~~including Performance Monitor~~)~~; configure and use App Insights runtime telemetry

Design an exception handling strategy

Handle exceptions across multiple layers; use MVC middleware to configure error handling; use different exception handling strategies for different environments; create and display custom error pages; configure a custom pipeline for error handling; handle first chance exceptions; configure and use App Insights; log application exceptions~~;~~

Test a web application

Create and run unit tests, for example, use the Assert class, create mocks and stubs; create and run web tests including using Browser Link; debug a web application in multiple browsers and mobile emulators; use Azure DevTest Labs; use Visual Studio Team Services

Debug an Azure application

Collect diagnostic information by using Azure App Insights; choose log types, for example, event logs, performance counters, and crash dumps; stream logs directly to Visual Studio from a deployed site; debug an Azure application by using Visual Studio and~~Studioand~~ remote debugging; interact directly with remote Azure websites using Server Explorer

Design and Implement Security (~~20~~15-~~25~~20%)

Configure authentication

Authenticate users; enforce authentication settings; implement ASP.NET Core Identity; enable Facebook, Google and other external providers; implement account confirmation, password recovery, and multi-factor authentication~~. P~~; performPerform authentication using Azure Active Directory, Azure Active Directory B2C, Azure Active Directory B2B, and Microsoft Identity~~.~~; ~~choose between Windows, Forms, custom authentication, and organizational accounts including work/school accounts and Active Directory–based providers;~~ manage user session by using cookies; acquire access tokens using the Microsoft Authentication Library (MSAL)~~configure membership providers; create custom membership providers; configure ASP.NET Identity~~

Configure and apply authorization

Create roles; authorize roles programmatically; configure and work with custom UserStores using middleware; configure controllers and actions to participate in authorization

Design and implement claims-based authentication ~~across federated identity stores~~

~~Implement federated authentication by using Azure Access Control Service;~~perform authentication and authorization using tokens ~~such~~including ~~as~~ OpenID, OAuth, JWT, SAML, bearer tokens, etc. ~~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; secure data using Azure Key Vault; ~~implement encryption for data protection using the data protection APIs~~ in transit and at rest

Implement a secure site ~~with ASP.NET~~

Secure communication by applying SSL certificates; require SSL for all requests; enable SSL hosting in the development environment; implement SSL using Azure Load Balancers; 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); ~~utilize~~use Azure Security Center to monitor Azure resources; implement Cross Origin Resource Sharing (CORS)~~CORS~~; implement protection against open redirect attacks