SPA Single Page Application Framework-L1

|  |  |
| --- | --- |
| **Assessment Name** | **TCA BAS-MS-SPA-L1** |
| **Total No. of questions** | **20** |
| **Marks** | **20** |
| **Percentage for Certification** | **70% & above** |
| **Duration of assessment** | **30 minutes** |
| **Topics** | **No. of questions** |
| **SPA Framework Features** | **7** |
| **SPA Template Overview** | **4** |
| **Backbone Template** | **3** |
| **Breeze-Angular Template** | **1** |
| **Hot Towel Template** | **2** |
| **KnockoutJS Template** | **3** |
|  | |

**Learning Contents:**

|  |  |
| --- | --- |
| **Sub Topics** | **Learning Contents** |
| **SPA Framework Features** | <http://dotnetguts.blogspot.in/2013/11/single-page-application-spa-overview.html> |
| <http://msdn.microsoft.com/en-us/magazine/dn463786.aspx> |
| <http://www.asp.net/single-page-application/overview/introduction/knockoutjs-template> |
| <http://msdn.microsoft.com/en-us/magazine/dn605877.aspx> |
| <http://knockoutjs.com/> |
| <http://knockoutjs.com/documentation/introduction.html> |
| <http://en.wikipedia.org/wiki/Single_page_application> |
| <http://channel9.msdn.com/Events/TechDays/Techdays-2012-the-Netherlands/2159> |
| **SPA Template Overview** | <http://www.asp.net/single-page-application/overview/templates/backbonejs-template> |
| <http://www.asp.net/single-page-application/overview/introduction/knockoutjs-template> |
| <http://www.asp.net/single-page-application/overview/templates/breezeknockout-template> |
| <http://www.asp.net/single-page-application/overview/templates/breezeangular-template> |
| <http://www.asp.net/single-page-application/overview/templates/emberjs-template> |
| **KnockoutJS Template** | <http://www.asp.net/single-page-application/overview/introduction/knockoutjs-template> |
| **Hot Towel Template** | <http://www.asp.net/single-page-application/overview/templates/hottowel-template> |
| **Breeze-Angular Template** | <http://www.asp.net/single-page-application/overview/templates/breezeangular-template> |
| **Backbone Template** | <http://www.asp.net/single-page-application/overview/templates/backbonejs-template> |

1. Breeze/Knockout Template Project. The template is packaged as **a Visual Studio Extension** (VSIX) file.
2. Backbone Template. The application is written **in TypeScript (.ts files)** which are compiled into JavaScript (.js files).
3. **Application** is defined in application.ts.
4. The **application.start** method creates the modal views and attaches event handlers for application-level events, such as user sign-in.
5. Instead of creating inter-dependencies among these components, the template uses **a "pub/sub" model**: The events object, defined in **events.ts**, acts as an event hub for **publishing and subscribing** to application events.
6. In Backbone.js, a **router** provides methods for routing client-side pages and connecting them to actions and events.
7. There are two kinds of views, **activable views** and **modal dialog views**. **Activable** views are invoked by the router.
8. Typically, the server renders the initial page and then sends and **receives JSON data**.
9. **AngularJS** is an open source library from Google for building Single Page Applications (SPAs)
10. The KnockoutJS template uses **Knockout** for data binding and raw **AJAX** for data access.
11. The Breeze/Angular template uses Angular for data binding and **Breeze** for data access.
12. DTO is an acronym for **Data Transfer Object**, a design pattern used in transferring data through internal or external interfaces. A DTO protects the application's internal data by acting as dummy storages, the whole logic is implemented only in actual Domain Objects (DO).
13. Ember.js **uses data binding** to synchronize the page with the latest data. That way, you don't have to write any of the code that walks through the JSON data and updates the DOM. Instead, you put declarative attributes in the HTML that tell Ember.js how to present the data.
14. The EmberJS template uses a combination of jQuery, Ember.js**, Handlebars.js** to create a smooth, interactive UI.
15. "Single-page application" (SPA) is the general term for a web application that loads a single HTML page and then updates the page dynamically, instead of loading new pages. After the initial page load, the SPA talks with the server through AJAX requests.
16. Server/Client update
17. But **Knockout** is not the only JavaScript library for creating rich client applications.
18. Knockout.js uses a pattern called "Model-View-ViewModel."
19. Many of the properties in the model classes are of type "ko.observable". Observables are how Knockout does its magic.
20. Knockout uses **declarative** bindings. You bind page elements to data by adding a "data-bind" attribute to the element.

**KnockoutJS template**

* The Knockout MVC Template is part of ASP.NET **and Web Tools 2012**.2
* An **observable** is a "JavaScript object that can notify subscribers about changes." When the value of an observable changes, Knockout updates any HTML elements that are bound to those observables.
* On the client side, the Knockout.js library handles page updates from **AJAX** requests. Knockout uses data binding to synchronize the page with the latest data. That way, you don't have to write any of the code that walks through the **JSON** data and updates the DOM. Instead, you put declarative attributes in the HTML that tell Knockout how to present the data.
* A big advantage of this architecture is that it separates the presentation layer from the application logic. You can create the Web API portion without knowing anything about how your web page will look. On the client side, you create a "view model" to represent that data, and the view model uses Knockout to bind to the HTML. That lets you easily change the HTML without changing the view model. (We'll look at Knockout a bit later.)
* The SPA template uses a combination of **jQuery** and **Knockout**.js to create a smooth, interactive UI. Knockout.js is a JavaScript library that makes it easy to bind HTML to data. Knockout.js uses a pattern called "**Model-View-ViewModel**."

1. The model is the domain data (ToDo lists and ToDo items).
2. The view is the HTML document.
3. **The view-model is a JavaScript object** that holds the model data. The view-model is a code abstraction of the UI. It has no knowledge of the HTML representation. Instead, it represents abstract features of the view, such as "a list of ToDo items".

* The view is data-bound to the **view-model**. Updates to the view-model are automatically reflected in the view. Bindings work the other direction as well. Events in the DOM (such as clicks) are data-bound to functions on the view model, which trigger AJAX calls.
* Many of the properties in the model classes are of type "ko.observable". **Observables** are how Knockout does its magic. From the Knockout documentation: An observable is a "JavaScript object that can notify subscribers about changes." When the value of an observable changes, Knockout updates any HTML elements that are bound to those observables.
* The main HTML for the page is defined in Views/Home/Index.cshtml. Because we are using data-binding, the HTML is only a template for what actually gets rendered. Knockout uses **declarative** bindings. You bind page elements to data by adding a "data-bind" attribute to the element. Here is a very simple example, taken from the Knockout documentation:

<p>There are <span data-bind="text: myItems().count"></span> items<p>

* Knockout provides a number of different binding types. Here are some of the bindings used in the SPA template:

1. **foreach**: Lets you iterate through a loop and apply the same markup to each item in the list. This is used to render the to-do lists and to-do items. Within the foreach, the bindings are applied to the elements of the list.
2. **visible**: Used to toggle visibility. Hide markup when a collection is empty, or make the error message visible.
3. **value**: Used to populate form values.
4. **click**: Binds a click event to a function on the view model.
5. **Backbone Template**

* Backbone.js SPA template. This template provides an initial skeleton for developing a Backbone.js application in ASP.NET MVC. Out of the box it provides basic user login functionality, including user sign-up, sign-in, password reset, and user confirmation with basic email templates.
* The application is written in TypeScript (.ts files) which are compiled into JavaScript (.js files).
* **Application:** is defined in application.ts. This object initializes the application and acts as the root namespace. It maintains configuration and state information that is shared across the application, such as whether the user is signed in.
* The **application.start** method creates the modal views and attaches event handlers for application-level events, such as user sign-in
* **Events:** Backbone provides built-in **events** with components such as **Model**, **Collection**, and **View**. Instead of creating inter-dependencies among these components, the template uses a **"pub/sub"** model: The events object, defined in events.ts, acts as an event hub for publishing and subscribing to application **events**. The **events** object is a singleton.
* **Router**: In Backbone.js, a router provides methods for routing **client-side pages** and connecting them to **actions** and **events**.
* **Views**: There are two kinds of views, **activable views** and **modal dialog views**. Activable views are invoked by the router. When an activable view is shown, all other activable views become inactive. To create an activable view, extend the view with the **Activable** object. Modal views are implemented as [Twitter Bootstrap](http://twitter.github.com/bootstrap/) modal dialogs. The Membershipand Profile views are modal views. Model views can be invoked by any application events. For example, in the Navigation view, clicking the "My Account" link shows either the Membership view or the Profile view, depending on whether the user is logged in. The Navigation attaches click event handlers to any child elements that have the data-command attribute.
* **Models:** The models all have three basic things: **default attributes**, **validation rules**, and a **server-side end point**
* **Plugins:** The ~/Scripts/application/lib folder contains a few handy jQuery plug-ins. The form.ts file defines a plug-in for working with form data. Often you need to serialize or deserialize form data and show any model validation errors. The form.ts plug-in has methods such as serializeFields, deserializeFields, and showFieldErrors. The flashbar.ts plug-in gives various kinds of feedback messages to the user. The methods are $.showSuccessbar, $.showErrorbar and $.showInfobar. Behind the scenes, it uses Twitter Bootstrap alerts to show nicely animated messages.

**Server Side:** **Controlllers:** In a single page application, the server plays only a small role in the user interface. Typically, the server renders the initial page and then sends and receives JSON data. The template has two MVC controllers: HomeController renders the initial page, and SupportsController is used to confirm new user accounts and reset passwords. All other controllers in the template are ASP.NET Web API controllers, which send and receive JSON data. By default, the controllers use the new WebSecurity class to perform user-related tasks. However, they also have optional constructors that let you pass in delegates for these tasks. This makes testing easier, and lets you replace WebSecurity with something else, by using an IoC Container.

BreezeJS is an open source library for managing rich data in a JavaScript client. Breeze handles querying, caching, change tracking, validation, and more.

1. **Breeze/Angular template**

* [AngularJS](http://angularjs.org/) is an open source library from Google for building Single Page Applications (SPAs). It offers data binding, dependency injection, and screen management. Combine it with [BreezeJS](http://www.breezejs.com/?utm_source=ms-spa), another open source library for **data** **modeling** and **data** **management**, and you have the essential ingredients for a great HTML/JavaScript client app.
* The KnockoutJS template uses **Knockout for data binding** and raw **AJAX for data access**. The Breeze/Angular template **uses Angular for data binding** and **Breeze for data access**. These libaries enable additional capabilities, including page navigation and history. Breeze bundles the changes and sends them together as a single request to the Web API controller's SaveChanges method. That's different from KockoutJS SPA template, which makes PUT, POST, and DELETE requests for each item individually.

1. **Breeze/Knockout template**

* The Breeze SPA template generates an application with the same user experience, but it has a different implementation, using **Breeze** **for data management**. The basic architecture is the same as the KockoutJS SPA template. However, the implementation is much simpler: The DTOs were deleted, and most of the Entity Framework details have been delegated to Breeze.NET.

1. **EmberJS template**

* The EmberJS SPA Template uses the [Ember](http://emberjs.com/) JavaScript library to handle page updates from AJAX requests. Ember.js uses **data binding** to synchronize the **page** with **the latest data**. That way, you don't have to write any of the code that walks through the JSON data and updates the DOM. Instead, you put **declarative attributes** in the HTML that tell Ember.js how to present the data.
* On the server side, the EmberJS template is almost identical to the [KnockoutJS SPA template](https://docs.microsoft.com/en-us/aspnet/single-page-application/overview/introduction/knockoutjs-template). It uses ASP.NET MVC to serve HTML documents, and ASP.NET Web API to handle AJAX requests from the client
* The EmberJS template uses a combination of **jQuery**, **Ember**.**js**, **Handlebars**.**js** to create a smooth, interactive UI.
* The Ember SPA template is a re-implementation of the Knockout SPA template, using EmberJS and Handlebars templating.

1. **Hot Towel template**

Hot Towel SPA template. This template brings in several JavaScript libraries, including Breeze, Knockout, RequireJS and Twitter Bootstrap

* ASP.NET MVC
* ASP.NET Web API
* ASP.NET Web Optimization - bundling and minification
* [Breeze.js](http://breezejs.com/) - rich data management
* [Durandal.js](http://durandaljs.com/) - navigation and View composition
* [Knockout.js](http://knockoutjs.com/) - data bindings
* [Require.js](http://requirejs.org/) - Modularity with AMD and optimization
* [Toastr.js](http://jpapa.me/c7toastr) - pop-up messages
* [Twitter Bootstrap](http://twitter.github.com/bootstrap/) - robust CSS styling

Hot Towel provides a logger module in the services folder. The logger module is ideal for logging messages to the console and to the user in pop up toasts.