SPA Workshop Script.

Introduction

**Comparison of SPA Vs. Traditional Web Applications**

There have been various methods over the years for composing pages. There are very traditional includes, such as in PHP, ASP, etc.

In the J2EE world there has been JSTL for includes, and libraries like Tiles for composing pages.

Even before all of this, designers and developers were using frames to try and avoid rebuilding the entire page.

\*\* Need images / slides to show the page composition techniques.

With an SPA, even when a template engine isn’t used, the entire page is rendered once, and from there only the portions of the page concerned with what is changing are updated. Users spend an increasing amount of time interacting with an application before it has to go back to the server. The blink of the page render is gone and replaced by progress bars or animations.

Responsibility also changes. The server can now focus on areas like services and security. Server side developers can focus on data storage and service interfaces without knowing anything about the look, feel or even design of the front end. Services can be designed to be a mix of public and private APIs.

On the front end, the responsibility for page composition is taken on, as well as routing logic and some business logic. It should be noted here that what doesn’t change, is the need to validate all data arriving at the server. Do not calculate the shopping cart total on the front end and then trust the value ☺.

For developers, there is much more clarity of role. Work can be more easily divided into roles; with developers work on the server, look & feel, and front end application logic.

\*\* Image for this would be nice.

**Framework Tour and Comparison**

There are a growing number of frameworks for developing client based applications, whether “SPA” style or not. We’re going to look at a couple of them and talk about their popularity, differences and finish with some warnings about relying too heavily on these frameworks.

All of these frameworks work to provide a structured MVC/MV\* pattern for building applications. Each usually provides more or less features with the trade being control of implementation and architecture or ease of use in the basic case.

**Angular.JS**

Angular is a library currently supported by Google. It’s widely popular and based on adding additional markup to your html. By placing markup you give Angular instructions and information about your page.

Angular is one of the larger frameworks available, because of all the features it provides. It’s Very declarative and inspired by ideas like Silverlight, are exemplified by the two way data binding and html markup control. Angular has no dependencies, but wants to stand on its own; so it doesn’t work well with other technologies like JQuery.

= Still more to add here =

Backbone.JS **@TODO**

Ember.JS @TODO

Knockout.JS @TODO

Market Share **@TODO**

Trends and other Info **@TODO**

Resources for getting this together:

<http://www.funnyant.com/choosing-javascript-mvc-framework/>

<http://dailyjs.com/2013/12/12/javascript-survey-results/>

<http://dailyjs.com/files/2013-survey-summary.pdf>

Stack overflow keyword counts?

Session 1. (1 hour)

Using Backbone to structure code.

Will show creation of a basic Model, Collection, and Views. This portion takes a lot from the JS Meetup presentation.

\*\* Get code more ‘presentable’ then showing in IDE. **@TODO**

Lecture Outline:

1. Backbone Idea (Uses \_.extend)
2. Models
3. Collections of Models.
4. View of single Model.
5. Collection View.

Exercise Outline:

1. Attendees will create the mode, collection and views for a new module in the application.
2. Attendees will be provided with the starting code and the solution will be available.
3. The requirements will be Maven, Java, and Tomcat.
4. Everything will be hosted on GitHub, including the solutions.
5. ***We should allow attendees to arrive early if they would like help setting up their development environment, or we will provide computers that are set up?***

Demo Outline:

1. Backbone Collection of Collections
2. Using a sub-view to manage views.
3. Time allowing, using Events between views.

Q&A

1. Attendees can ask questions, make suggestions, etc. Any question that is large or will be covered later will be written on the whiteboard and held until later or the end where a larger Q&A time is available.

This section is designed to be easy for the attendees to complete the exercise; with the hope that they finish it quickly and we can spend time on Events and answering questions. An early success should help attendees become involved, and seeing events (wiring) come into place should keep them interested.

Session 2. (1 hour)

Modular Development and Routing.

Will show how to organize code into modules and manage scope. Will also show how to use Routing in Backbone.

\*\* Get code more presentable than showing in IDE. **@TODO**

Lecture outline:

1. Organizing code into modules.
2. Using Modules to manage variable scope.
3. Using Routes in Backbone to manage application logic.

Exercise outline:

1. Users will move their previously written code into a module and include it in the main application.
2. Users will add an additional route to show their new functionality within the application.
3. Attendees will be provided with the starting code and the solution will be available.
4. The requirements will be Maven, Java, and Tomcat.
5. Everything will be hosted on GitHub, including the solutions.
6. ***We should allow attendees to arrive early if they would like help setting up their development environment, or we will provide computers that are set up?***

Demo outline:

1. Handling the back button and other routing techniques.

Q&A:

1. Attendees can ask questions, make suggestions, etc. Any question that is large or will be covered later will be written on the whiteboard and held until later or the end where a larger Q&A time is available.

Session 3. (1 hour)

Interfacing to various back ends.

Will show and talk about writing RESTful services and reusing older services. A brief explanation of JSON and the techniques required for managing non-blocking IO.

Lecture outline:

1. RESTful basics, managing CRUD (create, read, update, delete)
2. How RESTful services interface with Backbone / JQuery.
3. Managing JSON.
4. Using old services and frameworks.
5. Interfacing with a legacy application by adding a restful interface.

Exercise outline:

1. Attendees will create a REST service and inject the existing service layer.
2. Attendees will wire the new REST service to their application.

Demo outline:

1. Reusing an existing SOAP service.

Q&A:

1. Attendees can ask questions, make suggestions, etc. Any question that is large or will be covered later will be written on the whiteboard and held until later or the end where a larger Q&A time is available.

Session 4. (1 hour)

Developing in Backbone.

Will start to cover some more advanced Backbone techniques like Validation and Testing which are needed for real world development.

Lecture outline:

1. Validation in Backbone.
2. Testing in Javascript. (QUnit)

Exercise outline:

1. Attendees will add validation to their module’s Model objects.
2. Attendees will add tests to their models.

Demo outline:

1. Demo using Mocking to help with development and testing.
2. Automated testing with Grunt/Phantom.js

Q&A:

1. Attendees can ask questions, make suggestions, etc. Any question that is large or will be covered later will be written on the whiteboard and held until later or the end where a larger Q&A time is available.

Advanced Topics Presentation (15 minutes)

At the end before closing, we’ll cover some things we didn’t really have time for but that attendees will be concerned about. If everything moved quickly, security can be done as a final exercise where we show it and then work with attendees to add it to their application. *This means that a solution with security has to be ready.*

Presentation outline:

1. Will show how to secure the application and services. (Security)
2. Will show the application on a tablet, to demo how the pick list could be used in a warehouse.

Closing (30 minutes)

We’ll take 30 minutes to try and answer form frequently asked questions about SPAs. Hopefully this brief talk will prime attendees for asking questions and talking about the exercises and how they solved them.

FAQ entries:

1. Different front ends for mobile vs. desktop.
2. Deploying SPA’s as part of a class deployment process.
3. Development environment.

Depending on questions and talking by attendees, there will be a summary. We’ll show a list of additional resources on the board and probably as part of the *handout?*