AngularJS

Application Building Blocks

Objectives

- Accessing REST services
- UI - Router
- Project

AngularJS

Accessing REST Services

Server Access - Protocol

- Javascript(Browser based or mobile app) needs to access a server for storage and retrieval of data
- The communication protocol is HTTP
- Send a request using a Verb with data(resource)
- Verbs in HTTP (methods)
  - GET - Get resource(s)
  - POST - Store a resource
  - PUT - Update a resource
  - DELETE - Delete a resource

Server Access - Data format

- Used XML as data format - the X is Ajax over SOAP(Simple Object Access Protocol) on HTTP
- Parsing XML is cumbersome
- JSON - Javascript Object Notation
- JSON is more native to Javascript
- Easy forming and parsing of objects
- Today JSON is the standard even for Ajax
- We use REST over HTTP today
- REST is just a standard way of making requests (more later)

Server Access - Ajax

- NO full page refresh in browser
- Some event in browser sends a request and gets XML (the way originally thought of) or HTML page snippet
- Then XML blended with HTML or straight HTML is inserted into the DOM without page refresh
- We get JSON and we mix it with HTML template (using Angular script and Moustache template)
- Loading parts and pieces of HTML snippets are managed by AngularJS router (more later)

Server Access - REST

- REST - Representational State Transfer
- Not a strict standard
- The basic or operations that are building blocks of an application

URL	Method	Description
/categories	**GET**	**Retrieve List of categories**
/category/id	**GET**	**Retrieve a category**
	PUT	**Update a category**
	DELETE	**Delete a category**
/category	**POST**	**Save a new category**

Server Access - AngularJS

- AngularJS provides a HTTO based server access encapsulation ($http)
- $http service is at a lower level and more versatile
- Supports all HTTP methods
- Methods are invoked asynchronously
- Uses call back functions for handling successful and erroneous operations with promises (sure to get called)
- Can handle error status codes properly

$http

- Basic method call $http(config) -
  - Config Parameters
    - method - string (get/post/put etc.)
    - url - string
    - data - Json object
    - …
- Useful form of $http
  - $http.get(url)
  - $http.post(url, data)
  - $http.put(url, data)
  - $http.delete(url)

$http - promise handlers

- Promises (callbacks)
  - success(data, status, headers, config)
    - data- data returned from server (JSON)
    - status - response status code (200 OK)
    - headers - response headers
    - config - object used for generating request
  - error(data, status, headers, config)
    - data- data returned from server (JSON)
    - status - response status code (40X - client, 50X-server)
    - headers - response headers
    - config - object used for generating request

$http.get sample

```
$http.get('/GADemo/categories')
        .success(function(data, status, headers, config){
            $scope.categories = data;

        }).error(function(data, status, headers, config){
            switch(status) {
                case 400 : {
                    $scope.message = "Some error occured";
                }
                case 500 : {
                    $scope.message = "Internal error occured";
                    break;
                }
            }
            console.log(status);
 });
```

$http.post Sample

```
<div ng-controller="saveCategoryController">
    <form name="createcategory", ng-submit="saveCategory(category)">
        <table>
            <tr>
                <td>Code</td>
                <td><input type="text" ng-model="category.code"/></td>
            </tr>
            <tr>
                <td>Name</td>
                <td><input type="text" ng-model="category.name"/></td>
            </tr>
            <tr>
               <td>Description</td>
               <td><input type="text" ng-model="category.description"
/></td>
            </tr>
            <tr>
                <td><input type="submit" value="save"/></td>
                <td></td>
            </tr>
        </table>
    </form>
    {{message}}
</div>
```

$http.post Sample

```
app.controller('saveCategoryController', function($scope, $http){
    $scope.category = {};
    $scope.message = '';

    $scope.saveCategory = function(category) {
        $http.post('/GADemo/category', category)
                .success(function(data, status, headers, config){
                    $scope.message = data.message + ' with id: ' + data.id;
                    $scope.category = {};
                }).error(function(data, status, headers, config){
                    $scope.message = data.message;
                });
    };

});
```

$http

- Demo - List, save, list-show-update

- Exercise
- Create a page with list on top
- Edit and new category features with forms

AngularJS

UI Router

Single Page App

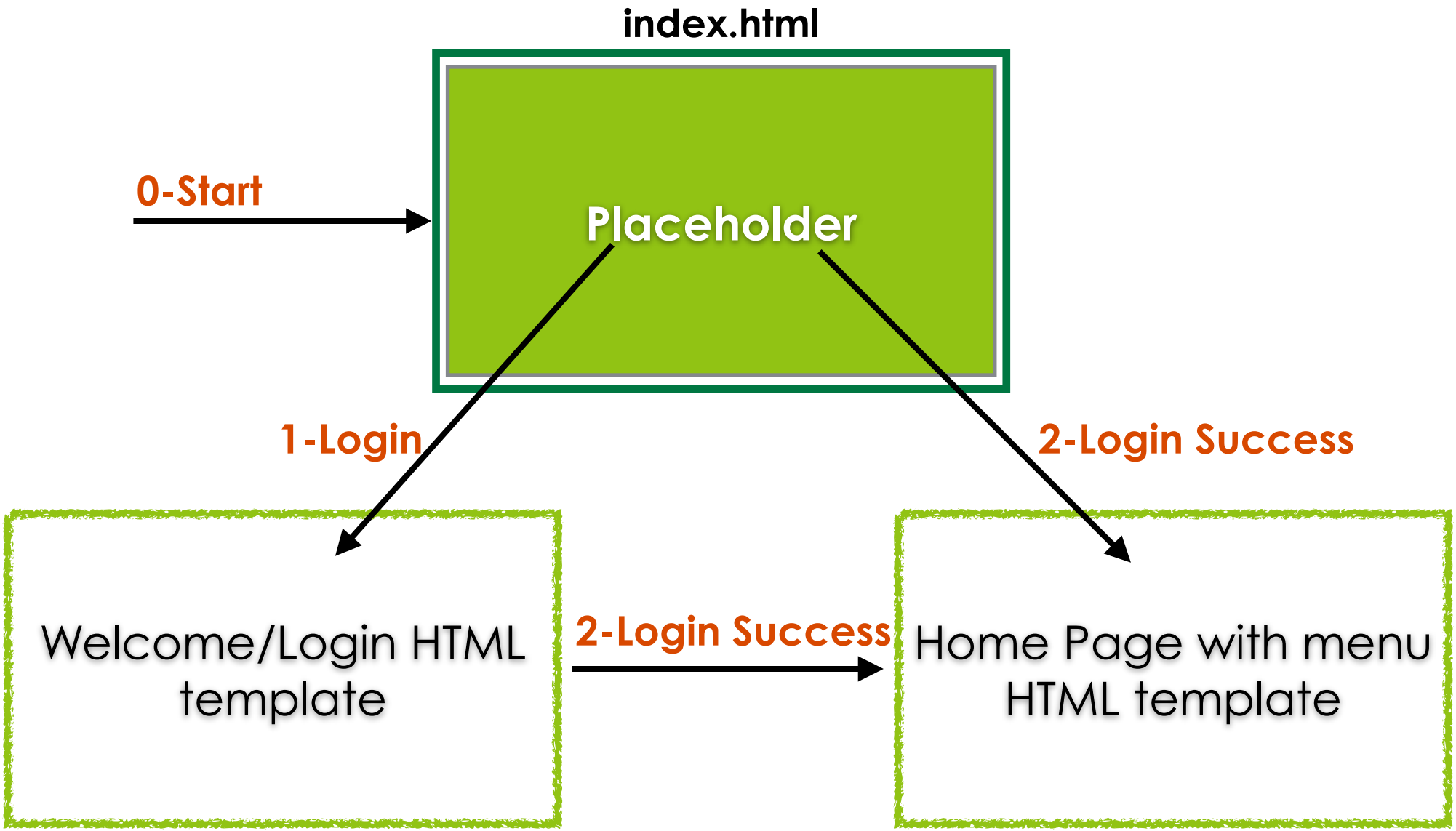
- We tried to create a single page where we can add/update categories and view the list
- Complex and long html page with show/hide logic on forms
- With AngularJS you will be able to modularize the app
- Create multiple html template snippets in files
- Create multiple controllers

- Create pages in a top down fashion
- Use html templates and place holders
- Manage the state and include or remove templates from a single page

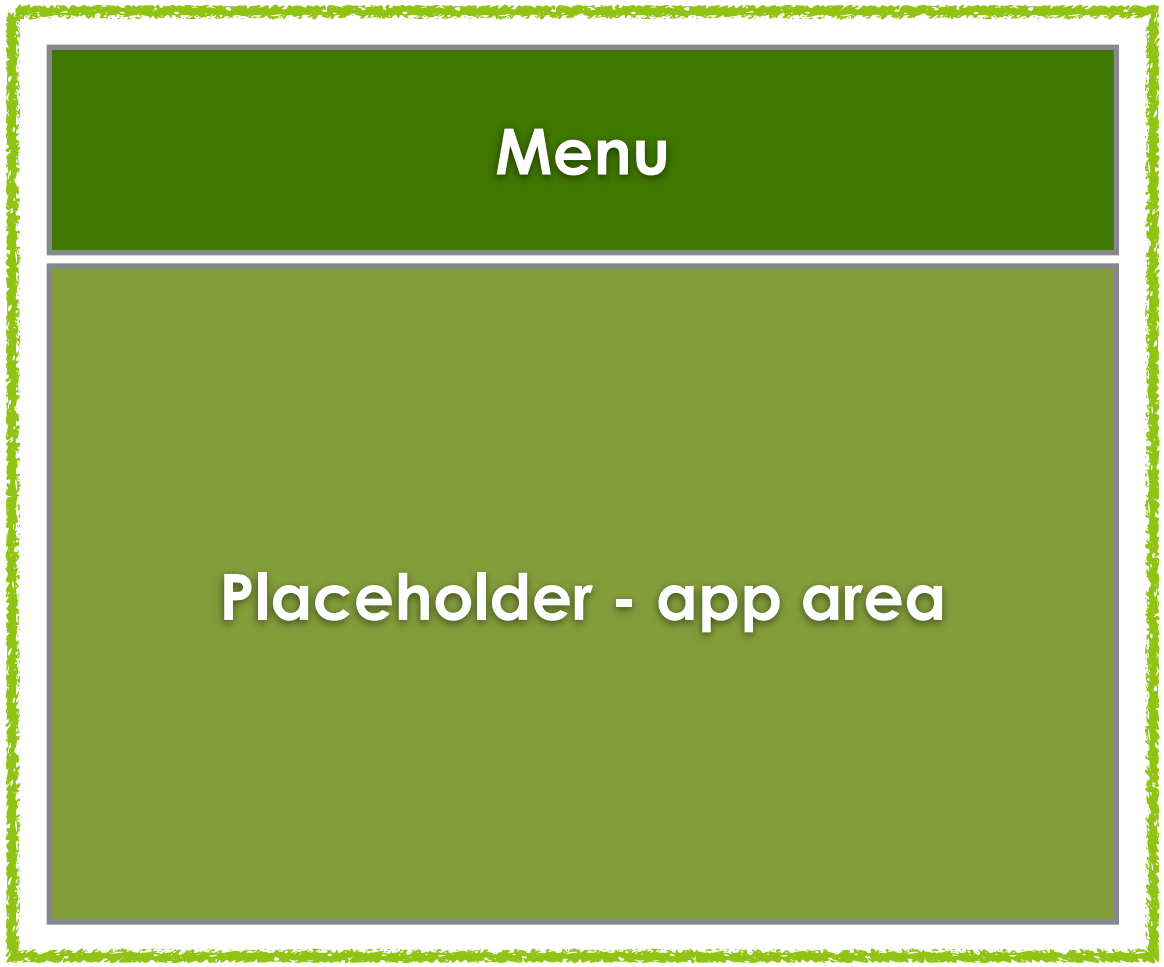
Angular Routers

- The concept of Single page App is facilitated by html templates
- Angular provides 2 ways of managing place holders and snippets
- ngRouter - single place holder, trivial navigation
- uiView - multiple place holders, state based navigation
- uiView is useful and versatile
- We will cover ui-view in this section
- Let us plan the categories and products app page templates and see how the navigation works

Router & Navigation



Router & Navigation



Home Page Template HTML

- Loaded with Menu for category, products & logout
- Loaded with default home contents
- Based on menu click or actions gets various templates to work with Categories and Products

Router & Navigation - Implementation

- index.html just defines a placeholder
- A div tag with ui-view directive

```
<body ng-app="rapp">
  <div ui-view>

  <div>
</body>
```

- We are going to keep all the scripts inside index.html
- Or we can create a javascript file and include it in the index.html

Router & Navigation - Implementation

- The script

```
var app = angular.module('rapp', ['ui.router']);
app.config(function($stateProvider, $urlRouterProvider){
        $urlRouterProvider.otherwise('/login');
        $stateProvider
         .state('login', {
           url : '/login',
           templateUrl: 'login-tpl.html',
           controller : 'loginController'
         })
         .state('home', {
           url : '/home',
           templateUrl : 'home-tpl.html'
         })
       ...
}
```

Router & Navigation - Implementation

- Create an app module with ui-router injected in
- Configure the app with injected stateProvider and urlRouterProvider
- For urlRouterProvider default view will be login (this will make the login template loaded into index.html
- Then for stateProvider set the states with a name (‘login’) and configure with url, templateUrl or template (html), controller etc.
- login-tpl.html

```
<p>Login credentials enter here!</p>
<input type="button" value="Login"
                ng-click="loginClicked()">
```

Router & Navigation - Implementation

- Controller Script in index.html for login…

```
app.controller('loginController',
        function($scope, $state){ //$state injected
  $scope.loginClicked = function() {
    //is login success?
    $state.go('home'); //change the status
  }
});
```

- When the status is changed (on successful login) as per state definition the template home-tpl.html is loaded

Router & Navigation - Implementation

- The home template file has a menu and a placeholder to show the result of clicking various menus

```
<div >
    <a ui-sref="home.categories">Categories</a>
          
    <a ui-sref="#">Products</a> <!– done later –>
</div>
<div ui-view>

</div>
```

- For anchor tag use ui-sref instead of href and provide the state to get into
- But before that we need to load the default home page content available in defhome-tpl.html (dashbaord?)

Router & Navigation - Implementation

- default home content below menu

```
<h1>Default home contents</h1>
```

- To load this by default

```
$urlRouterProvider.when('/home', '/home/default');
```

- When ‘/home’ is asked for set the state to ‘/home/default’
- So, we need to add a new state called ‘home.default’

```
.state('home.default', {
          url : '/default',
          templateUrl : 'defhome-tpl.html'
        })
```

- So, after successful login the home sate with default content will be shown

Router & Navigation - Implementation

- Handling the Categories menu - we have provide a state of home.categories in ui-sref

```
.state('home.categories', {
            url : '/categories',
            templateUrl : 'catlist-tpl.html'
          })
```

- in catlist-tpl.html we are going to have a link that will jump to a show category page

```
<h2>category list here</h2>
<a ui-sref="home.catshow({id:10})">Show at 10</a>
```

- When link is clicked the status will change to home.catshow and a parameter (id of category) is passed

Router & Navigation - Implementation

- State in index.html

```
.state('home.catshow', {
          url : '/category/:id',
          templateUrl : 'catshow-tpl.html',
          controller : 'catshowController'
        });
```

- The controller catshowController

```
app.controller('catshowController',
                  function($scope, $stateParams){
        $scope.catid = $stateParams.id;
       //id passed is set to scope
});
```

- catshow-tpl.html

```
<p>displaying the category for id {{catid}}</p>
```

Demo & Exercise

- Demo - Setup Category navigation

- Exercise - Setup Product navigation

Final Exercise

- Grails app with Category and Product
- Create a Single page with CRUD

Thank You!

Bala Sundarasamy
bala@ardhika.com