Lab Guide

From Jelly to AngularJS

Mitch Stutler and Brad Tilton

Default Login / Password:

admin / Knowledge16

itil / Knowledge16

employee / Knowledge16

This
Page
Intentionally
Left
Blank

Lab Goal

Lab 1
Load AngularJS

You will start with a custom Jelly UI page that displays a list of projects from your ServiceNow instance. In this lab, you will create a copy of the original UI page and load AngularJS into our new page.

Acceptance Criteria

The user will have a new UI page with AngularJS loaded. To confirm that AngularJS has been loaded correctly, you will display a variable from our AngularJS app on the UI page.

Create New UI Page

1. Log into your instance.
2. Find and open the UI page named project_jelly.
3. Change the name of this UI page to project_angular and do an 'Insert and Stay'. This newly created page is what you will be modifying going forward.

Configure UI Page

1. Add the Angular reference:
    a. Near the bottom of the HTML field add a script reference to the AngularJS Google CDN. Make sure this is before the closing Jelly tag.

```
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.5/angular.min.js"></script
>
```

2. Create an Angular module:
    a. In the Client Script field you need to create an Angular module.
    b. Add this line:

```
var projectApp = angular.module('projectEditor', []);
```

3. Create an Angular controller:
    a. Below your module in the Client Script field, you will need to create an Angular controller.
    b. Add the following code to your Client Script:

©2016 ServiceNow, Inc. All rights reserved.

```
projectApp.controller('ProjectController', [
'$scope',
'$http',
function($scope, $http) {
    $scope.helloWorld = 'Hello, World!';
}]);
```

4. Configure HTML field to access variables from our Angular app:
    a. First, you'll call our app by adding the ng-app="projectEditor" to our container div on line 40.
    b. Next, you'll call the controller by adding ng-controller="ProjectController" to the row div on line 41.
    c. To ensure that you have Angular set up correctly, you will display the helloWorld variable you declared in your client script. To do this, add the following line to the HTML field below the row div where you called your controller:

```
<h3>{{helloWorld}}</h3>
```

5. Verify that Angular has been configured correctly:
    a. Click the 'Try It' UI Action to view your UI Page.
    b. If you successfully followed this lab, you should see the text 'Hello, World!' directly above your list of projects.

©2016 ServiceNow, Inc. All rights reserved.

Lab Goal

Lab 2
Dynamically Display Data

With Jelly we typically use <j:while> tags to dynamically loop through an array of data. In this lab, you will convert the existing <j:while> configuration to use Angular's ng-repeat attribute. To do this, you will use a hard-coded array of project data that will be defined in your Client Script.

Acceptance Criteria

You will be see a short list of project data that you defined in your Client Script.

Define Your Hard-coded Data

1. Add the below code to your Angular controller in your Client Script. This should be placed directly below the helloWorld variable declaration.

```
$scope.projects = [{'sys_id': '82bfc50c930202003b4bb095e57ffb82',
'short_description': '3 ServiceNow Integration with OBIEE', 'number': 'PRJ0000004',
'state': 'Pending', 'percent_complete': 0, 'start_date': '2016-01-17 18:30:00',
'end_date': '2016-06-03 19:30:00'},{'sys_id': '268d010c930202003b4bb095e57ffbfc',
'short_description': '2 ServiceNow New Reports interface development',  'number':
'PRJ0000002', 'state': 'Work in Progress', 'percent_complete': 0, 'start_date':
'2016-01-17 18:30:00', 'end_date': '2016-05-31 19:30:00'},{'sys_id':
'd8eb0dc8930202003b4bb095e57ffb0f', 'short_description': '1 Workday Integration
with SAP',  'number': 'PRJ0000001', 'state': 'Work in Progress',
'percent_complete': 6, 'start_date': '2015-11-27 18:30:00', 'end_date': '2016-04-10
19:30:00'}];
```

Replace Jelly Code

1. Since you are now handling the project data in your client script, the <g:evaluate> tags are no longer necessary. Remove lines 28-34 in your HTML field.

2. Add a placeholder table header after line 44. This will be in place for a future step in the lab. Paste the following line:

```
<td class="normal-cell"></td>
```

3. Replace the table body with the following code:

©2016 ServiceNow, Inc. All rights reserved.

```
<tbody>
    <tr ng-repeat="proj in projects">
        <td colspan="6">
            <table>
                <tr>
                    <td class="normal-cell"><a class="project-link"
href="pm_project.do?sys_id={{proj.sys_id}}" style="color: #91b4c5; font-weight:
600;">{{proj.number}}</a><div>{{proj.short_description}}</div></td>
                    <td class="normal-cell">{{proj.state}}</td>
                    <td class="normal-cell">{{proj.percent_complete}}</td>
                    <td class="normal-cell">{{proj.start_date}}</td>
                    <td class="normal-cell">{{proj.end_date}}</td>
                    <td class="normal-cell"></td>
                </tr>
            </table>
        </td>
    </tr>
</tbody>
```

4. Validate your changes from this lab:
    a. Click ‘Try It’ to view your updated UI page. You should see a list of 3 projects in your table.

©2016 ServiceNow, Inc. All rights reserved.

Lab Goal

With Jelly we used a GlideRecord query inside of <g:evaluate> tags to grab the project data. With Angular, we will use a REST message to retrieve this data.

Acceptance Criteria

You will be see a full list of projects similar to the original UI Page.

Lab 3
Retrieve Data with REST

Update Controller

1. Replace the controller code in your Client Script with the below code:

```
projectApp.controller('ProjectController', [
'$scope',
'$http',
function($scope, $http) {
    $scope.isLoaded = false;
    $scope.loading = true;
    $scope.url = '/api/now/table/pm_project';
    $http.defaults.headers.common.Accept = "application/json";

    $scope.updateRecordList = function() {
          $http({
                method: 'GET',
                url: $scope.url +
"?sysparm_query=active=true^ORDERBYDESCend_date&sysparm_limit=15&sysparm_display_va
lue=true"
          }).
          success( function(data, status) {
                $scope.loading = false;
                $scope.isLoaded = true;
                $scope.projects = data.result;
                for (var i=0; i< $scope.projects.length; i++) {
                      $scope.projects[i].showNotes = false;
                }
                console.log(data.result);
          }).
          error ( function(data, status) {
                $scope.projects = [{"number": "Error fetching list"}];
          });
    };

    $scope.updateRecordList('');
```

©2016 ServiceNow, Inc. All rights reserved.

```
}] );
```

2. Remove the Hello World test in the HTML since we removed it from our controller:
    a. Delete the <h3>{{helloWorld}}</h3> that should be on line 35.

3. Add an attribute to our tbody tag that will only show the table after the data has been retrieved:
    a. Add the attribute ng-show="isLoaded" to our tbody tag on line 47.

4. Validate your changes from this lab:
    a. Click ‘Try It’ to view your updated UI page. You should see a list of 15 projects that looks similar to the original Jelly UI page.

©2016 ServiceNow, Inc. All rights reserved.

Lab Goal

With Jelly we would need to use GlideAjax or reload the UI page and use a Processing Script if we wanted to update any of the records we were viewing. With Angular, we can make a REST call to update the record without leaving or reloading the page.

Acceptance Criteria

You will be able to update the Work Notes of any of the projects in the list from your UI page.

Lab 4
Update Notes with REST

Update HTML

1. Add buttons to each row that will allow you to open a notes section for any given row:
    a. On line 57, you should have an empty table cell that looks like this: <td class="normal-cell"></td>. You are going to replace that with the below code:

```
<td class="normal-cell">
    <button ng-click="toggleNotes($index)" ng-show="!proj.showNotes">Add
Notes</button>
    <button ng-click="saveNotes($index)" ng-show="proj.showNotes">Save
Notes</button>
</td>
```

2. In your nested table, you will add a new row that will contain a hidden notes input div that will become visible when you click the 'Add Notes' button from the previous step.
    a. Directly after the <tr> block that is displaying the project data, add the below code. This should be on line 62.

```
<tr>
    <td colspan="6">
        <div ng-show="proj.showNotes">
            <div class="col-md-6"></div>
            <div class="col-md-5">
                <label>Notes:</label>
                <textarea class="form-control" rows="5" ng-
model="proj.notes"></textarea>
            </div>
            <div class="col-md-1"></div>
        </div>
    </td>
```

©2016 ServiceNow, Inc. All rights reserved.

```
</tr>
```

Update Client Script

1. You will need new functions in your controller to handle the actions associated with the button clicks and updating the Work Notes with a REST Message. Add the below code to your controller below the $scope.updateRecordList function declaration:

```
$scope.updateNotes = function(id, notes) {
    $http({
          method: 'PATCH',
          url: $scope.url + "/" + id,
          headers: {'Content-Type': 'application/json'},
          data: {'work_notes': notes}
    }).
    success( function(data, status) {
          $scope.status = "Successfully updated Work Notes.";
    }).
    error ( function(data, status) {
          $scope.status = "Error updating Work Notes.";
    });
};

$scope.toggleNotes = function(idx) {
    $scope.projects[idx]['showNotes'] = !$scope.projects[idx]['showNotes'];
};

$scope.saveNotes = function(idx) {
    $scope.updateNotes($scope.projects[idx]['sys_id'],
$scope.projects[idx]['notes']);
    $scope.toggleNotes(idx);
    $scope.projects[idx]['notes'] = '';
};
```

Validate Changes

1. Validate your changes from this lab:
    a. Click ‘Try It’ to view your updated UI page. You should now see a similar list to the previous lab, except now there should be buttons titled ‘Add Notes’ on the far right of each row.
    b. Click one of the ‘Add Notes’ buttons and you should see a Notes text input appear below that row.
    c. Enter test notes into the text input and then click ‘Save Notes’.
    d. Navigate to that project record by clicking the link on the project number and check the ‘Notes’ form section. You should see your test notes in the Activities area for that project.

©2016 ServiceNow, Inc. All rights reserved.

Lab Goal

In this lab, you will create an Angular directive that will add a more aesthetically pleasing graphic to display the Percent Complete.

Acceptance Criteria

You will see the same list as the previous lab, except there will be a graphic instead of integer for the Percent Complete column.

Lab 5 Percent Complete Angular Directive

Update Client Script

1. You will need to define the custom directive in your Client Script outside of the controller definition.
    a. Paste the following code below your controller in your Client Script:

```
projectApp.directive('pctComplete', function() {
    return {
          restrict: 'E',
          replace: true,
          scope: {
                value: '='
          },
          template: ' <div class="c100 p{{value}}">\
          <span>{{value}}%</span>\
          <div class="slice">\
          <div class="bar"></div>\
          <div class="fill"></div>\
          </div>\
          </div>'
    };
});
```

Update HTML

1. There should already be a style sheet named ‘angular-progress’. You will need to call this style sheet near the top of the HTML field.
    a. Paste the following line directly below the Bootstrap call. This line should be on line 4.

©2016 ServiceNow, Inc. All rights reserved.

```
<link href="c04180ac0fb61200aff186bce1050e42.cssdbx?" rel="stylesheet"
type="text/css" />
```

2. Call the new directive where you are displaying the Percent Complete cell for each row.
    a. Replace line 55 with the below code:

```
<td class="normal-cell"><pct-complete value="proj.percent_complete"></pct-
complete></td>
```

Validate Changes

1. Click 'Try It' to view your updated page. You should now see the new Percent Complete graphic.

®2016 ServiceNow, Inc. All rights reserved.