AngularJS - Best Practice

Target Audience: CAST Administrators

Summary: This document provides a guide based on CAST's field experience on how to configure AngularJS based applications using CAST AIP. These configurations are used all versions of CAST AIP 8.2.x and above for AngularJS offering (see Reference Materials below for links) to overcome limitations such as missing links. The applicability of this guide should have assessed for newer versions of CAST AIP and Extension.

Introduction to AngularJS

This section gives a brief overview of the framework.

AngularJS is an open source web application framework. It was originally developed in 2009 by Misko Hevery and Adam Abrons. It is now maintained by Google.

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. AngularJS's data binding and dependency injection eliminate much of the code you would otherwise have to write. And it all happens within the browser, making it an ideal partner with any server technology.

Core Features
Following are most important core features of AngularJS

- **Data-binding** It is the automatic synchronization of data between model and view components.
- **Scope** These are objects that refer to the model. They act as a glue between controller and view.
- **Controller** These are JavaScript functions that are bound to a particular scope.
- **Services** AngularJS come with several built-in services for example $https: to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app.
- **Filters** These select a subset of items from an array and returns a new array.
- **Directives** Directives are markers on DOM elements (such as elements, attributes, css, and more). These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives (ngBind, ngModel...)
- **Templates** These are the rendered view with information from the controller and model. These can be a single file (like index.html) or multiple views in one page using "partials".
- **Routing** It is concept of switching views.
- **Model View Whatever** MVC is a design pattern for dividing an application into different parts (called Model, View and Controller), each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel). The Angular JS team refers it humorously as Model View Whatever.
- **Deep Linking** Deep linking allows you to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state.
- **Dependency Injection** AngularJS has a built-in dependency injection subsystem that helps the developer by making the application easier to develop, understand, and test.

Versions of AngularJS Supported by CAST

This section highlights the AngularJS versions supported by CAST AIP 8.2.x.

- Angular JS 1.3 Extension Supports 1.0 to 1.6 Version of Angular JS
- Angular JS 1.2 Extension (Beta) Supports 1.0 to 1.6 Version of Angular JS
- Angular JS 1.1 Extension Supports 1.0 to 1.5 Version of Angular JS

File Types expected for CAST Analysis

This section highlights all the file types that can be expected to be delivered if the application has implemented this framework.

- HTML
- JS
- JSP
- ASPX, etc.,

AngularJS

AngularJS is what HTML would have been, had it been designed for applications. HTML is a great declarative language for static documents. It does not contain much in the way of creating applications, and as a result building web applications is an exercise in *what do I have to do to trick the browser into doing what I want?*

The impedance mismatch between dynamic applications and static documents is often solved with:

- **a library** - a collection of functions which are useful when writing web apps. Your code is in charge and it calls into the library when it sees fit. E.g., `jQuery`.
- **frameworks** - a particular implementation of a web application, where your code fills in the details. The framework is in charge and it calls into your code when it needs something app specific. E.g., `durandal`, `ember`, etc.

How to identify the implementation of AngularJS:

This section details the approach for identifying the presence of AngularJS in the code delivered.

- ng tags in the script
- part of the HTML contains the AngularJS app. This done by adding the ng-app attribute to the root HTML element of the AngularJS app. You can either add it to html element or body element as shown below
  <body ng-app = "myapp"> </body>
- **View**

  The view is this part
   <h2>Welcome `helloTo.title` to the world of Tutorialspoint!</h2>
  <ng-controller tells AngularJS what controller to use with this view. helloTo.title tells AngularJS to write the "model" value named helloTo.title to the HTML at this location.
- **Controller**

  The controller part is
  <script> angular.module("myapp", .controller("HelloController", function($scope) { $scope.helloTo = {}; $scope.helloTo.title = "AngularJS"; }); < /script>
  This code registers a controller function named HelloController in the angular module named myapp. The controller function is registered in angular via the angular.module(...).controller(...) function call.
  The $scope parameter passed to the controller function is the model. The controller function adds a helloTo JavaScript object, and in that object it adds a title field.
- **Execution**

  When the page is loaded in the browser, following things happen

  - HTML document is loaded into the browser, and evaluated by the browser. AngularJS JavaScript file is loaded, the angular global object is created. Next, JavaScript which registers controller functions is executed.
  - Next AngularJS scans through the HTML to look for AngularJS apps and views. Once view is located, it connects that view to the corresponding controller function.
  - Next, AngularJS executes the controller functions. It then renders the views with data from the model populated by the controller. The page is now ready.

AngularJS Description:

This section gives a brief overview of AngularJS.

AngularJS takes another approach. It attempts to minimize the impedance mismatch between document centric HTML and what an application needs by creating new HTML constructs. AngularJS teaches the browser new syntax through a construct we call *directives*.

AngularJS Components
The AngularJS framework can be divided into following three major parts

- **ng-app** This directive defines and links an AngularJS application to HTML.
- **ng-model** This directive binds the values of AngularJS application data to HTML input controls.
- **ng-bind** This directive binds the AngularJS Application data to HTML tags.

How to configure AngularJS in CAST AIP:

This section describes all the CAST configuration steps to be followed in order to configure AngularJS based application.

Server manager Configuration:

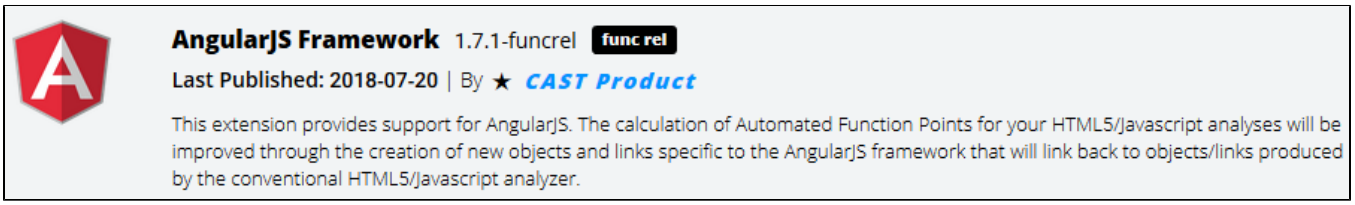
This section describes the Server Manager Configuration Steps.

Once after successful CAST Extend - AngularJS installation, Enable the extension through "Server Manager".

Pre Analysis Configuration:

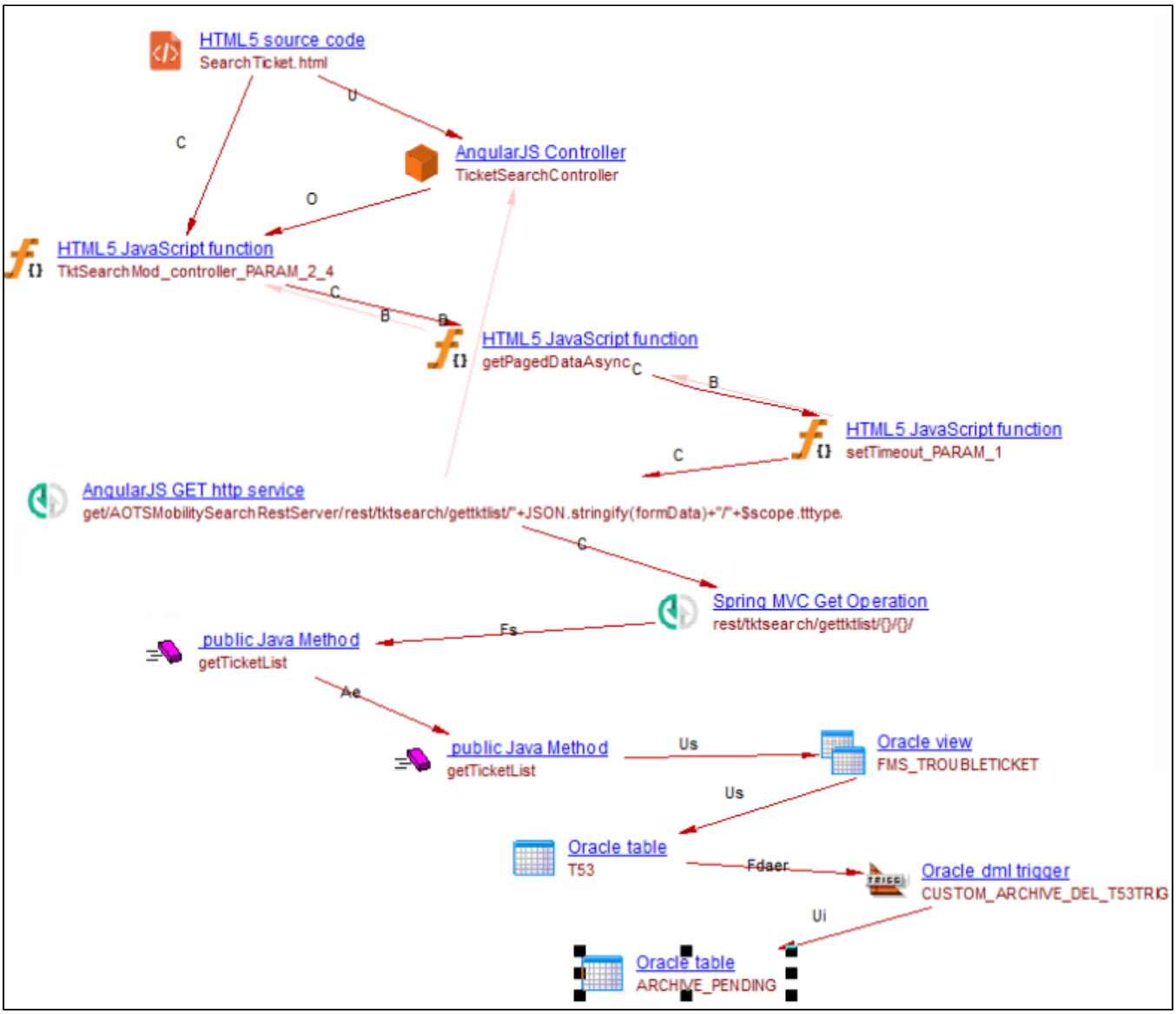
This section describes the "Pre Analysis Configuration" Steps.

Install AngularJS Extension from CAST Extend - AngularJS. Manage it using Server Manager. Note that there is no need to create any exclusive CMS Job as mandated in other extensions.



Enlighten

This section gives an overview of the Enlighten diagram.



CMS Configuration:

This section describes the CMS Configuration which is required to set for AngularJS.

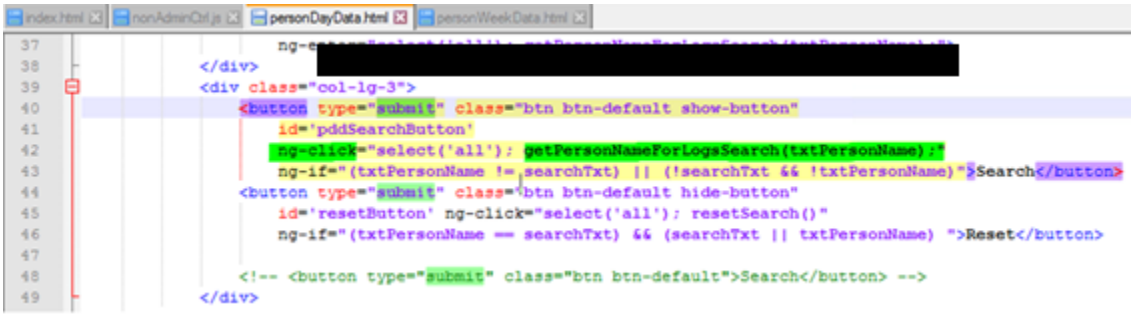
Name: URLCallInJS
Pattern: var[ ]+url[ ]*\=[ ]*SPEConstants[A-Za-z0-9_\.]+\.([A-Za-z0-9_]+)\;
Replacement: \1__URL
Type Of link: Match

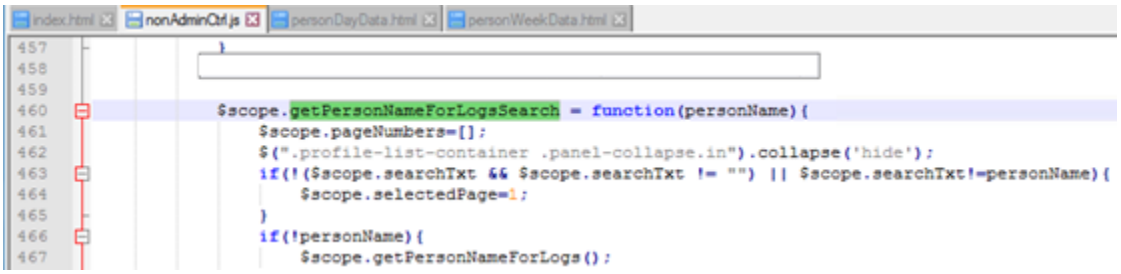
Name: SPEConstantValues
Pattern: webAPI[A-Za-z0-9_\.]+\.([A-Za-z0-9_]+)\=[A-Za-z0-9_\+ \"]+V([A-Za-z0-9_]+)\"
Replacement: \1__\2__METHOD
Type Of link: Match

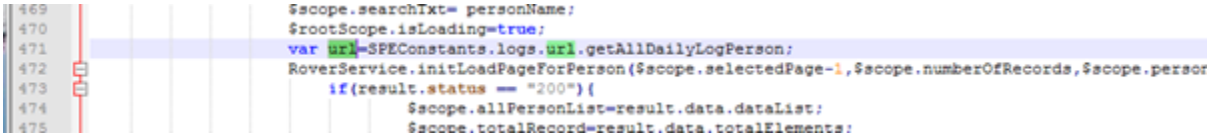
KB ModificationQueries:

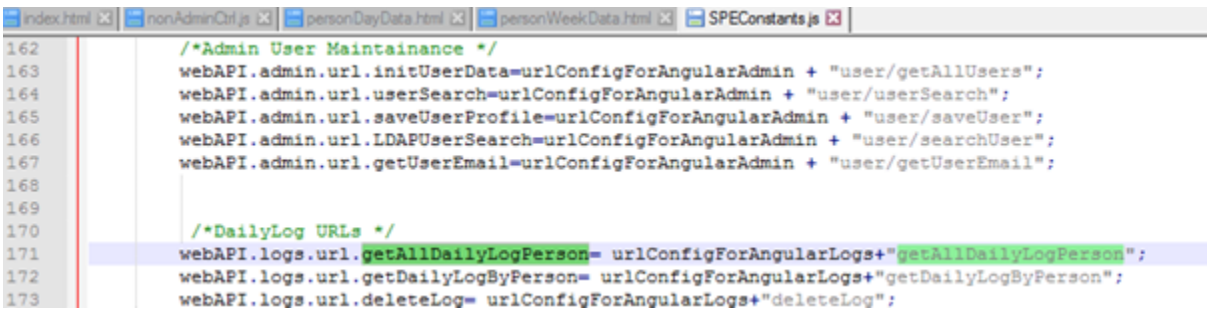
JavaScriptFunctionToJavaMethodLink:
/* Template creating new links between existing unlinked objects */
insert into §CI_LINKS (CALLER_ID, CALLED_ID, LINK_TYPE, ERROR_ID)
select distinct ctv.caller_id,c2.object_id ,'callLink',0 from
§cdt_objects c1,§cdt_objects c2,§ctv_links ctv,§cdt_objects c3,
§cdt_objects c4,§csv_object_descriptions c5
where ctv.caller_id = c1.object_id and
c1.object_type_str = 'HTML5 JavaScript function'
and c2.object_type_str = 'Java Method' and c3.object_type_str ='Search String'
and c4.object_type_str = 'Search String' and ctv.called_id = c3.object_id
and c3.object_name like '%URL' and c4.object_name like '%METHOD'
and split_part(c3.object_name,'__',1) = split_part(c4.object_name,'__',1)
and split_part(c4.object_name,'__',2) = split_part(split_part(c5.description,'/',2),"",1)
and c5.object_id = c2.object_id and
c5.desc_type= 'Annotation:' and c5.description like '%RequestMapping%'
/

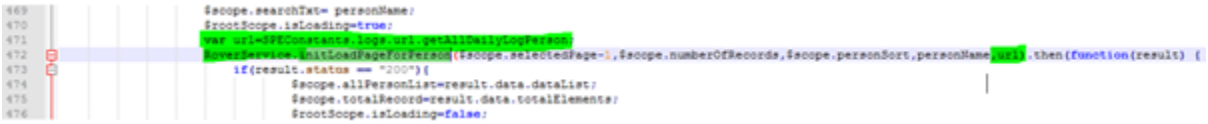
Sample code:







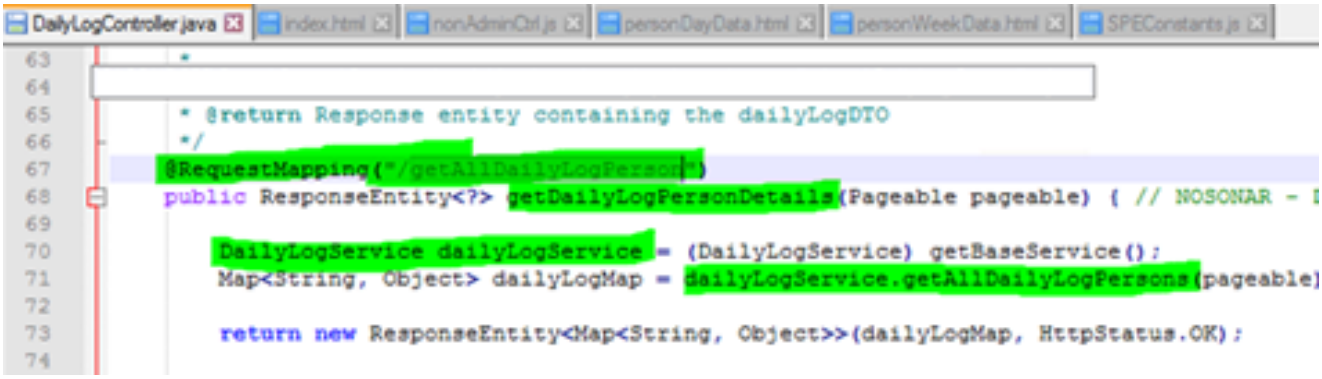




```
$scope.searchTxt= personName;
$rootScope.isLoading=true;
var url=SPEConstants.logs.url.getAllDailyLogPerson;
RoverService.initLoadPageForPerson($scope.selectedPage-1,$scope.numberOfRecords,$scope.personSort,personName,url).then(function(result) {
    if(result.status == "200"){
            $scope.allPersonList=result.data.dataList;
            $scope.totalRecord=result.data.totalElements;
            $rootScope.isLoading=false;
```

roverService.js | DailyLogRepository.java | DailyLogServiceImpl.java | DailyLogController.java | index.html | nonAdminCtrl.js

```
45                  };
46
47              },
48              initLoadPageForPerson : function(page, size,sort ,personName,url) {
49                  personName = encodeValue(personName);
50                  url = url + "/" + personName;
51                  console.log(url);
52                  return result = $http({
53                      method : 'GET',
54                      url : url + '?page=' + page + '&size=' + size + '&sort=' +sort
55                  }).success(function(data, status, headers, config) {
56                      return data;
57                  }).error(function(data, status, headers, config) {
58                      if(data.message.indexOf("Invalid token")>=0)
59                      {
60                          $window.location.href = './unauthorized.html';
61                      }
62                      else if(data.message.indexOf("Unauthorized access for role")>=0)
63                      {
64                          $window.location.href = './unauthorizedPage.html';
65                      }
66                      else
67                      {
68                          /*$window.location.href = './errorMsg.html';*/
69                      }
70                      return {
```



DailyLogController.java | index.html | nonAdminCtrl.js | personDayData.html | personWeekData.html | SPEConstants.js

```
63      *
64
65      * @return Response entity containing the dailyLogDTO
66      */
67     @RequestMapping("/getAllDailyLogPerson")
68     public ResponseEntity<?> getDailyLogPersonDetails(Pageable pageable) { // NOSONAR -
69
70         DailyLogService dailyLogService = (DailyLogService) getBaseService();
71         Map<String, Object> dailyLogMap = dailyLogService.getAllDailyLogPersons(pageable)
72
73         return new ResponseEntity<Map<String, Object>>(dailyLogMap, HttpStatus.OK);
74
```

TCC Configuration:

This section describes the TCC configuration which needs to be set.

Entry points

- HTML
- JSP
- ASPX

Limitations

If the application uses Angular JS with JSP, then the links between them needs to be created.

Reference

Reference Material	Link
Cast Sotware for AngularJS Extension	https://doc.castsoftware.com/display/TECHNOS/AngularJS
What Is AngularJS?	https://docs.angularjs.org/guide/introduction