**Code Walkthrough**

**URL**: <https://testtwexw7186bb8d.int.sap.hana.ondemand.com/ui/#/0>

1. When URL is accessed, it calls the UI Services (/ui) and gives login window.
2. On successful login, it calls several services (listed below) along with the loading of screen using UI5 components:
3. Project
4. Language
5. Domains
6. User

On hitting the URL, the request goes to web.xml file in server of the deployed (ui) module which contains the mapping of servlet

<servlet>

<servlet-name>CXFServlet</servlet-name>

<servlet-class>org.apache.cxf.jaxrs.servlet.CXFNonSpringJaxrsServlet</servlet-class>

<init-param>

<param-name>javax.ws.rs.Application</param-name>

<param-value>com.sap.translationservice.service.web.extern.app.WorkflowApplicationExternal</param-value>

</init-param>

</servlet>

1. Servlet CXFServlet will be called and it goes to: com.sap.translationservice.service.web.extern.app.**WorkflowApplicationExternal**
2. WorkflowApplicationExternal contains services

**public** **class** WorkflowApplicationExternal **extends** BaseApplication {

**public** WorkflowApplicationExternal() {

classes.add(LanguageService.**class**);

classes.add(DomainService.**class**);

classes.add(TexttypeService.**class**);

classes.add(Workflow.**class**);

classes.add(ProjectService.**class**);

classes.add(UserService.**class**);

classes.add(EditTranslationService.**class**);

classes.add(ReportTranslationService.**class**);

classes.add(UploadFileService.**class**);

classes.add(DownloadTranslationsService.**class**);

classes.add(UserAssistanceService.**class**);

Services:

1. Language Services(/languages):

“/languages” is configured as path in LanguageService.java.

LanguageServices.java

**public** **class** LanguageService {

**private** **static** **final** Logger ***log*** = LoggerFactory.*getLogger*(LanguageService.**class**);

**public** **static** **final** String ***XS\_SERVICE\_URL*** = "/sap/translsvc/xsServices/STH/LanguageService.xsjs";

@QueryParam("search")

**public** String searchQuery;

**public** LanguageService() {

}

@GET

@Consumes(***JSON\_MEDIA\_TYPE***)

@Produces(***JSON\_MEDIA\_TYPE***)

**public** String getLanguages() {

**if** (searchQuery != **null** && !searchQuery.isEmpty()) {

searchQuery = "search="+searchQuery;

}

String response = "";

**try** {

response = XSConnection.*getIt*(searchQuery, ***XS\_SERVICE\_URL***, WorkflowXSConnection.***DESTINATION\_NAME***);

} **catch** (XSConnectionException e) {

***log***.error("Error when reading from XSJS Language Service", e);

**throw** **new** InternalServerErrorException();//NOSONAR

}

response = response.replaceAll("\"NAME\"", "\"name\"");

response = response.replaceAll("\"ID\"", "\"id\"");

**return** response;

}

}

**XS\_SERVICE\_URL:**contains the path of LanguageServices.xsjs

**@QueryParam("search"):**It will check for query parameter

**@GET:** It specifies type of request it will support.

**@Consumes(JSON\_MEDIA\_TYPE):** It supports request parameter in JSON format

**@Produces(JSON\_MEDIA\_TYPE):** It ensures the response will be generated in JSON format.

**response=XSConnection.*getIt*(searchQuery, *XS\_SERVICE\_URL*, WorkflowXSConnection.*DESTINATION\_NAME*)**

Above code will call **getIt** method with parameters

WorkflowXSConnection.DESTINATION\_NAME=”xsbackend”

**getIt** method will call HttpConnectionHelper.call(TYPE\_GET, parameters, targetURL, destinationName) internally.

**HttpConnectionHelper.call(TYPE\_GET, parameters, targetURL, destinationName):** It will call LanguageServices.xsjs internally and get response.

response = response.replaceAll("\"NAME\"", "\"name\"");

response = response.replaceAll("\"ID\"", "\"id\"");

**return** response;

At last replaceAll operation is done and response is returned.