DAC Plugin FRAMEWORK

Contents

[PreRequisites: 3](#_Toc43210997)

[Understanding the Process Flow: 3](#_Toc43210998)

[Initialization: 5](#_Toc43210999)

[Naming Convention: 5](#_Toc43211000)

[Comments: 6](#_Toc43211001)

[Exception Handling: 6](#_Toc43211002)

[Sample Project Structure: 7](#_Toc43211003)

Sample POSTMAN Example……………………………………………………………………………………………………………….9

PreRequisites:

* JDK-1.8, JRE-8
* Eclipse/STS
* Tomcat/Jetty Server
* Maven-3.6
* Postman

Understanding the Process Flow:

A REST API should be entered with no prior knowledge beyond the initial URI and set of standardized media types that are appropriate for the intended audience (i.e., expected to be understood by any client that might use the API).

Request:

* **REST** requires that a client make a **request** to the server in order to retrieve or modify data on the server. A **request** generally consists of: an HTTP verb, which defines what kind of operation to perform, a header, which allows the client to pass along information about the **request**, a path to a resource.
* Limit to the Request is confined to the Application.

**JSON request Example:**

{

"UserName":"newbeg2" ,

"UserId":"ashish",

"Environment":"SAP",

"EMAIL":"sahana.b.s@capgemini.com"

}

Each request is being used to create a user,reset a password ,assign a role to a user or delete a user which differs for the particular request made.

**Methods of request:**

Two common methods for the request-response between a server and client are:

* **GET**- It requests the data from a specified resource.
* **POST**- It submits the processed data to a specified resource.

Response:

* An HTTP **response** contains the status line, headers and the message/**response** body.
* The content that is returned by the **REST API** in the message body uses the JavaScript Object Notation (JSON) format.

Response of a GET API invoked- For any given HTTP GET API, if the resource is found on the server then it must return HTTP response code 200 (OK) – along with response body which is usually either XML or JSON content (due to their platform independent nature).

Response of a POST API invoked- For any given HTTP GET API, if the resource is found on the server then it must return HTTP response code 200 (OK) – along with response body which is usually either XML or JSON content (due to their platform independent nature).

Note: HTTP response code will vary depending on the API development. It can be 201 also.

Initialization:

This module will handle all the initialization of all variables and necessary configurations.

Naming Convention:

**Classes/** **Interfaces:**

* Class name starts with uppercase letter and every inner word starts with uppercase letter (camel case convention).
* Ex: (Nouns) String, StringBuffer etc
* Class name should follow- <Should define your usecase>

**Methods:**

* Class name starts with Lowercase letter and every inner word starts with uppercase letter (mixed case convention).
* Ex: (verbs) charAt(), post() etc.
* Method name should follow- post<functionality >
* Eg. [http://localhost:8080/usermanagement/userInfo-->where](http://localhost:8080/usermanagement/userInfo--%3ewhere) functionality is the userInfo.

**Variables:**

* Class name starts with Lowercase letter and every inner word starts with uppercase letter (mixed case convention).
* Ex: firstName, mobileNumber etc.

**Package:**

* Package name is always must written in lowercase letters.
* Package Name: com.capgemini.dac.automate.<ProjectName>

**Constants**:

* While declaring constants all the words should be in uppercase letters
* Ex: MOBILE\_NUMBER, HIGH\_PRIORITY

**Note**: These coding standards are mandatory for predefined library and optional for user defined library but as a Java developer good to follow for user defined also.

Comments:

* Comments are used to write detailed description for logic to understand easily what it’s doing. Those are non-executable code.
* It’s important in real time because today we are developing the application but it’s maintained by some other so to understand logics.
* Ex: syntax: //description
* Syntax: /\*\*

                \*Statement        \*/

* Comments have to be provided for each class level and method level.
* Header(Copyright details should be specified) before package

Exception Handling:

An exception can occur for many different reasons.

* A user has entered an invalid data.
* A file that needs to be opened cannot be found.
* A network connection has been lost in the middle of communications or the JVM has run out of memory.

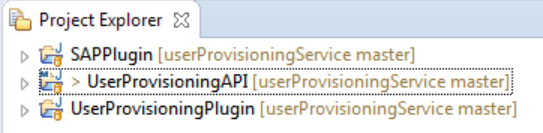
Exception occurs during the execution of the program. If the application contains exception then the program terminates abnormally and rest of the code will not executed. To overcome that and execute rest of the code we must handle the exception. Handling the exception using,

1. try-catch block

2. throws keyword

**Sample Project Structure**

The Project structure below is a Maven Web Service.Get/Post method requests are written here with the following hierarchy.



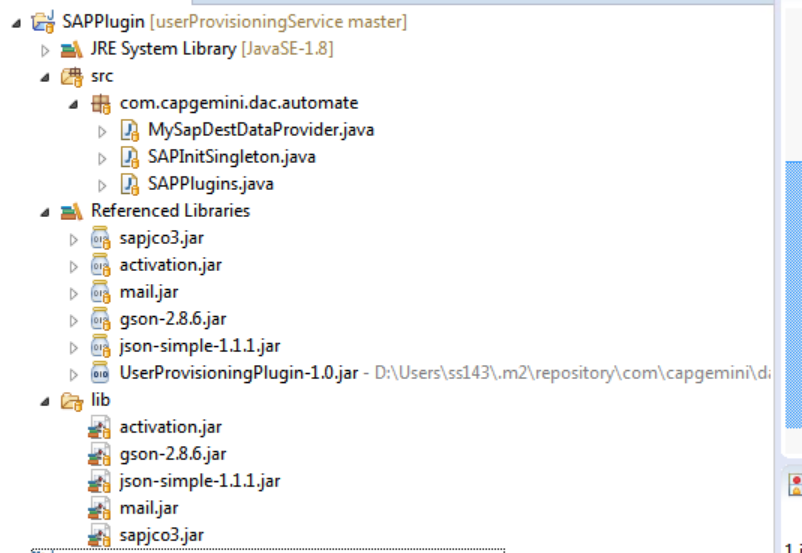
1.UserProvisioning API-will contain the request data and parameters information

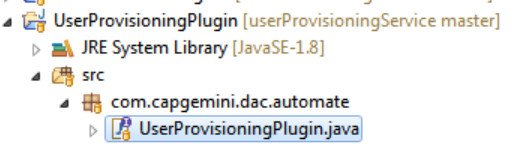
2.UerProvisioningPlugin- contains the an interface class which is used to pass parameters from API Request to Plugin.

3.Plugin-(In this case,SAPPlugin)- will contain the business logic of API which implements the following functionalities of user Provisioning:

* Create User
* Reset Password for a user
* Get the userInfo(validity period,mail id,password status)
* Assign a role to a user
* Delete the user

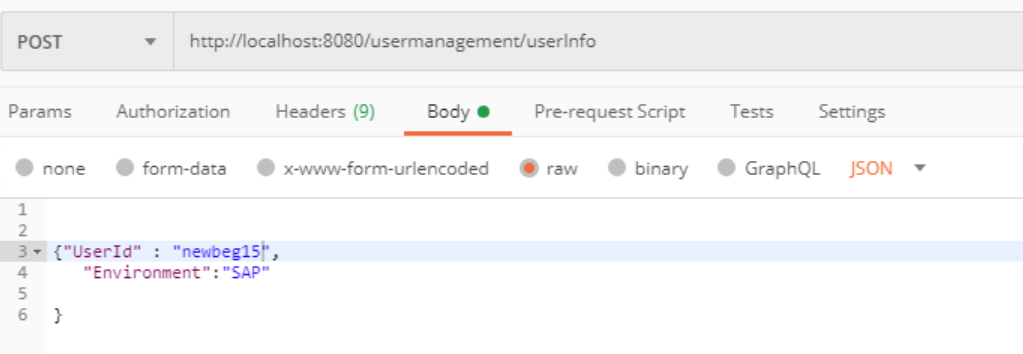
The Project structure below is a Standalone Java which contains the Plugin(ie,here is the SAPPlugin).Also it will implements the UserProvisioningPlugin ,which is an interface between the API and Standalone Java Application.This Project will contain the business logic of API .



The UserProvisioningPlugin Project contains the an interface class which is used to pass parameters from API Request to Plugin.

**Sample POSTMAN Example**

Below screenshot depicts the sample data for POST request for the user Info.



Below screenshot depicts the sample data for response

