Application of Microservice Architecture to B2B Processes

(IBM Watson Customer Engagement)

by

Arpit Jain

(2015047)

Supervisor(s):

External
Mr. Atul A. Gohad
(IBM ISL, Bangalore)

Internal
Dr. Aparajita Ojha
(PDPM IIITDM Jabalpur)

Computer Science and Engineering

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, DESIGN AND MANUFACTURING JABALPUR

(19th October 2018– 3rd November 2018)

Introduction

The International Business Machines Corporation (IBM) is an American multinational technology company headquartered in Armonk, New York, United States, with operations in over 170 countries. IBM manufactures and markets computer hardware, middleware and software, and provides hosting and consulting services in areas ranging from mainframe computers to nanotechnology.

IBM aims to bring Businesses closer and smarter than ever with the help of their state of the art enterprise software product called B2B Sterling Integrator. IBM B2B Integrator helps companies integrate complex B2B (Business to Business) / EDI (Electronic Data Exchange) processes with their partner communities. IBM aims to transform the B2B Sterling product into Microservice architecture.

Brief Overview

During the duration of my last report, I created a new service for Sterling Integrator – JSON/XML Transformer. This will allow the customers to use inbuilt XML/JSON service for their data conversion needs. Also, I worked upon the Rest API client service (Pre-Midterm) for making it more production ready as it is going to be passed to the QA (Quality Assurance) stage for final testing.

In this report, I covered the tasks of committing the changes in the product done by me to the Official GitHub Repository for the production version. The official release of Sterling Integrator V6.0.1 will be somewhere around May 2019. But the changes were to be committed by me so that the new product features can go to the QA testing cycle phase.

After this, I had to explain my product feature use-cases in a meeting with the QA team. Apart from this, I had to complete the documentation of all the features I have added for the team members to refer after I leave. Also, I am planning to write another IBM developer-works article on using SpringBoot with Business APIs.

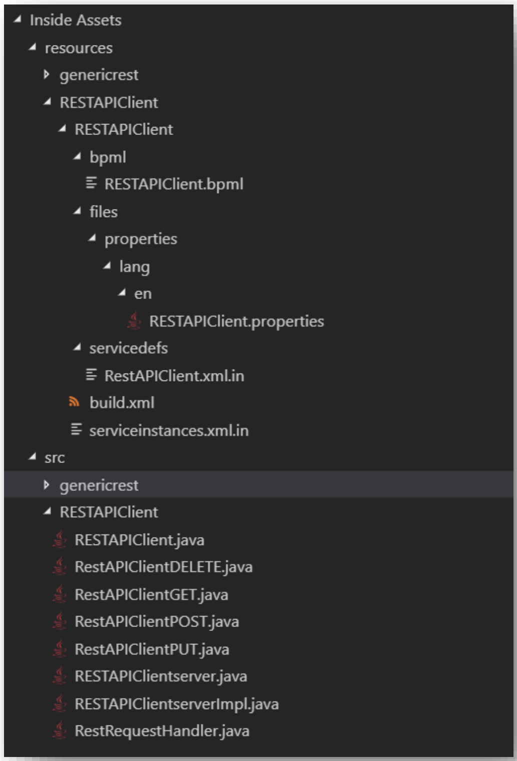
Later, we also demonstrated the POC for linking the XML-JSON-Transformer service to the REST-API-Client.

Report on the Present Investigation

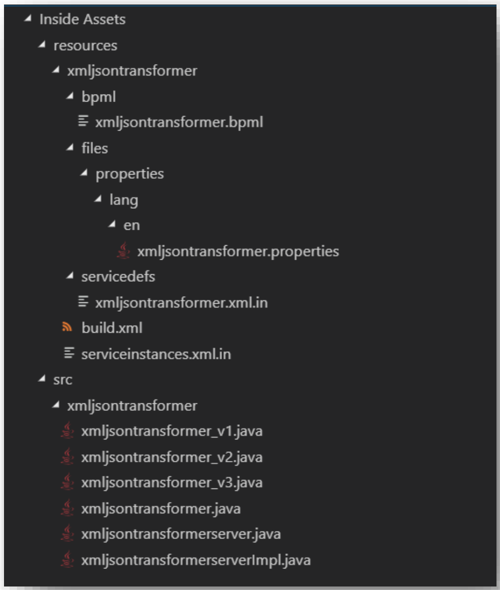
(Progress during this 15-days period)

Refinement of the existing REST API Client service and XML JSON Transformer services

Existing services which were created had to be more production ready, so we organized the code structure to make it more understandable for the future maintenance. Also, we made improvements to the file structure organization for making it ready to be committed to the GitHub repository. After this, we had to explain this code to the future maintainers so that they can handle the bugs and issues in these product features during the QA cycles.



```
Inside Assets
  resources
    genericrest
    RESTAPIClient
      RESTAPIClient
        bpml
          RESTAPIClient.bpml
        files
          properties
            lang
              en
                RESTAPIClient.properties
        servicedefs
          RestAPIClient.xml.in
        build.xml
        serviceinstances.xml.in
  src
    genericrest
    RESTAPIClient
      RESTAPIClient.java
      RestAPIClientDELETE.java
      RestAPIClientGET.java
      RestAPIClientPOST.java
      RestAPIClientPUT.java
      RESTAPIClientserver.java
      RESTAPIClientserverImpl.java
      RestRequestHandler.java
```



```
Inside Assets
  resources
    xmljsontransformer
      bpml
        xmljsontransformer.bpml
      files
        properties
          lang
            en
              xmljsontransformer.properties
      servicedefs
        xmljsontransformer.xml.in
      build.xml
      serviceinstances.xml.in
  src
    xmljsontransformer
      xmljsontransformer_v1.java
      xmljsontransformer_v2.java
      xmljsontransformer_v3.java
      xmljsontransformer.java
      xmljsontransformerserver.java
      xmljsontransformerserverImpl.java
```

Adding Reference to Third Party Jars in Sterling Integrator

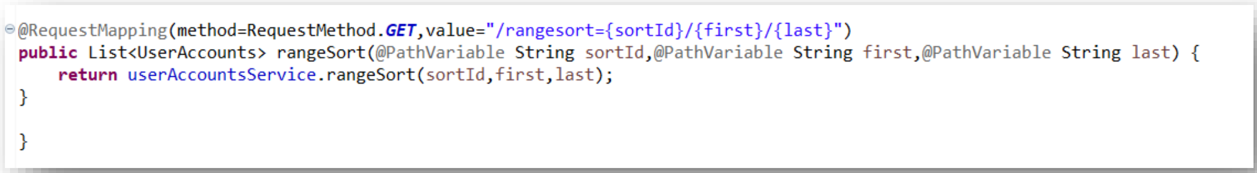
Initially, we were building the classes by adding references to the Thirdparty Jars in dynamic classpath for the product. That is a Hardcoded way of specifying the location of the Jars. One disadvantage of this methodology is that for this, we need to handle the maintenance of the future upgrades(versions) of jars as well. So, it becomes a bit complicated on our end. So as to avoid the complicacy, we added the Jars as a File-Set in the Base source clump. After this, we added the reference entries for these Jars in Build.xml file. So now the jars are being referenced and classes being compiled.

Following were the external third-party open-source JAR files required for the two services –

- Json-simple.jar
- Hamcrest-Core.jar
- Junit.jar
- Org.json.jar

SpringBoot UserAccounts Microservice improvements

In the past few days, some features were also added to UserAccounts Microservice. Now, it can take Sort and Range parameters simultaneously from the users and perform sorting on the fetched range of user accounts. On fetching all user accounts instances, it is being restricted to some 1000 useraccounts for performance benefits. Initially, it was fetching all the useraccounts irrespective of size. This became a bottleneck in the case of heavily populated Database.



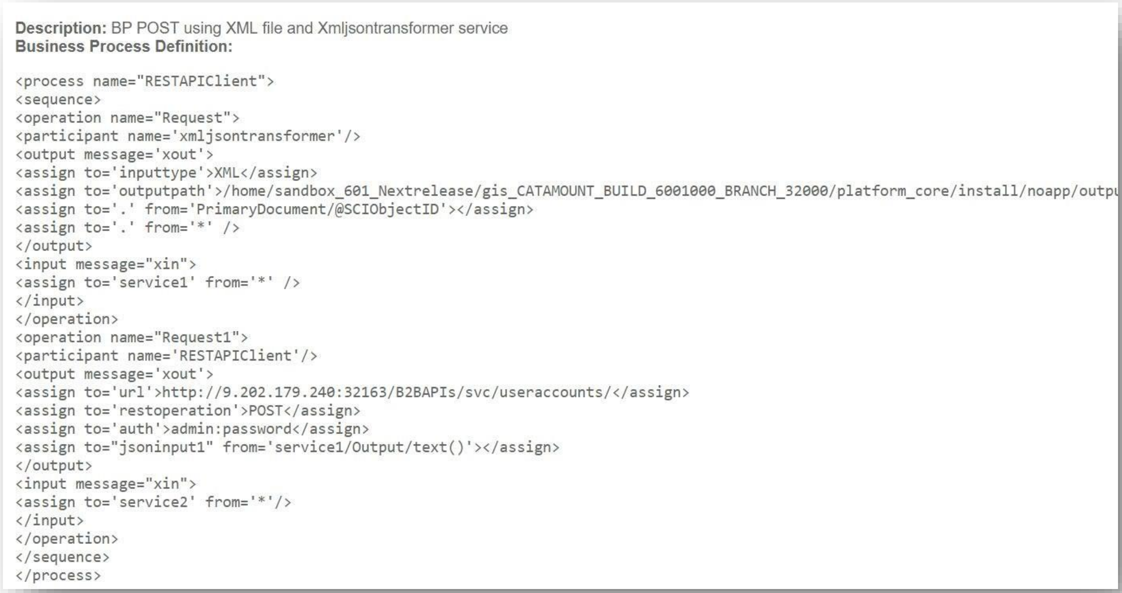
```
@RequestMapping(method=RequestMethod.GET,value="/rangesort={sortId}/{first}/{last}")
public List<UserAccounts> rangeSort(@PathVariable String sortId,@PathVariable String first,@PathVariable String last) {
    return userAccountsService.rangeSort(sortId,first,last);
}

}
```

Linking XML-JSON-Transformer with REST-API-Client service

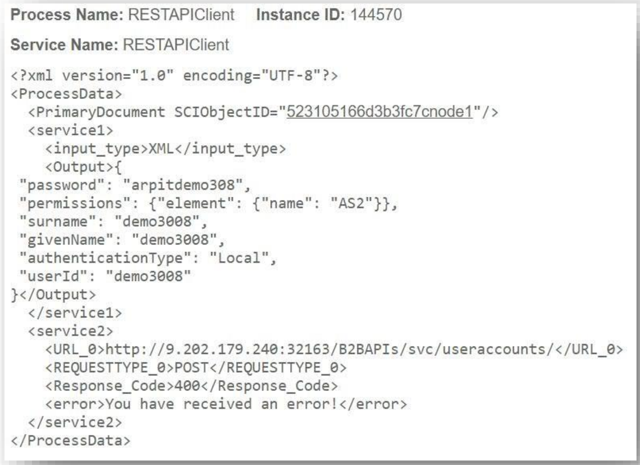
The following snippet of the Business Process shows the sequence of the two instances of separate services. The first service XML-JSON-Transformer takes a file input in XML Format and converts it into JSON format.

The second service (REST API Client) picks up this output from the first service and Posts it to form a UserAccount using the Tenx UserAccount API.



Description: BP POST using XML file and Xmljsontransformer service
Business Process Definition:

```
<process name="RESTAPIClient">
<sequence>
<operation name="Request">
<participant name='xmljsontransformer'/>
<output message='xout'>
<assign to='inputtype'>XML</assign>
<assign to='outputpath'>/home/sandbox_601_Nextrelease/gis_CATAMOUNT_BUILD_6001000_BRANCH_32000/platform_core/install/noapp/outpu
<assign to='.' from='PrimaryDocument/@SCIObjectID'></assign>
<assign to='.' from='*' />
</output>
<input message="xin">
<assign to='service1' from='*' />
</input>
</operation>
<operation name="Request1">
<participant name='RESTAPIClient'/>
<output message='xout'>
<assign to='url'>http://9.202.179.240:32163/B2BAPIs/svc/useraccounts/</assign>
<assign to='restoperation'>POST</assign>
<assign to='auth'>admin:password</assign>
<assign to="jsoninput1" from='service1/Output/text()'></assign>
</output>
<input message="xin">
<assign to='service2' from='*'/>
</input>
</operation>
</sequence>
</process>
```



Process Name: RESTAPIClient   **Instance ID:** 144570

Service Name: RESTAPIClient

```
<?xml version="1.0" encoding="UTF-8"?>
<ProcessData>
  <PrimaryDocument SCIObjectID="523105166d3b3fc7cnode1"/>
  <service1>
    <input_type>XML</input_type>
    <Output>{
 "password": "arpitdemo308",
 "permissions": {"element": {"name": "AS2"}},
 "surname": "demo3008",
 "givenName": "demo3008",
 "authenticationType": "Local",
 "userId": "demo3008"
}</Output>
  </service1>
  <service2>
    <URL_0>http://9.202.179.240:32163/B2BAPIs/svc/useraccounts/</URL_0>
    <REQUESTTYPE_0>POST</REQUESTTYPE_0>
    <Response_Code>400</Response_Code>
    <error>You have received an error!</error>
  </service2>
</ProcessData>
```

Results and Discussions

Before Midterm, I built a REST Client Adapter for the Sterling Integrator which allowed the users to place REST calls over Business APIs. Earlier, these APIs only accepted JSON as input. Using this XML JSON Transformer service as an intermediate layer, users will have the versatility to provide input in both formats. Although JSON is the accepted standard for API inputs and outputs, still the users of Sterling Integrator require this feature as Business Processes and some other important components of the product follow XML standards. In this report, I have shown the demonstration of a single workflow to call both services one after the other in order to give XML format inputs to REST API Client too.

Conclusions

During past few weeks, I was involved in creating a new service for the Sterling Integrator. This Service is called XML JSON Transformer. This service would allow the user to transform the data from XML to JSON and vice-versa. Using this service, input could be taken from the user in any format and using this service, it can be converted to the opposite format and then be supplied as input to some other service. This small but useful service is an integral part of what the users were really demanding.

Also, apart from this, I refined all the product additions I performed during the duration of my internship and I have started working upon the documentation of the same.

Final Target

My target for the next few days is to complete the documentation of all the product features I have worked on.