



HCL Volt MX

Foundry でのカスタムコード (Processors)

Student Guide



HCL Software Academy for HCL Digital Solutions

Creating a new generation of experts

もくじ

Foundry でプリプロセッサを実装する	3
前提条件	3
JavaScript を使用して Foundry でプリプロセッサを実装および構成する	3
JavaScript を使用した Foundry のポストプロセッサの実装と設定	7
Java を使用した Foundry でのプリプロセッサの実装と設定	10
Java を使用した Foundry のポストプロセッサの実装と設定	17
Java プロセッサを Foundry に統合する	21

Foundry でプリプロセッサを実装する

このレッスンでは、プロセッサを実装するために Foundry でカスタムコードを記述することを紹介します。このレッスンでは、以下について学習します。

- JavaScript を使用して Foundry でプリプロセッサを実装および設定する方法
- JavaScript を使用して Foundry でポストプロセッサを実装および構成する方法
- Java を使用して Foundry でプリプロセッサを実装し、設定する方法
- Java を使用して Foundry でポストプロセッサを実装および設定する方法

このドキュメントでは、このレッスンのハンズオン部分の詳細な手順について説明します。

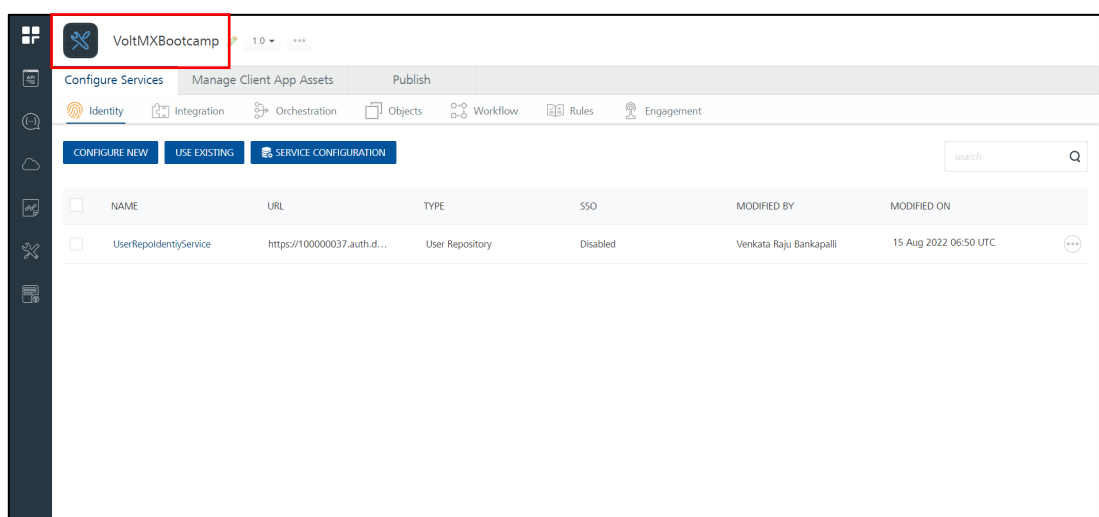
前提条件

- 前のレッスンでハンズオンの手順を完了していること

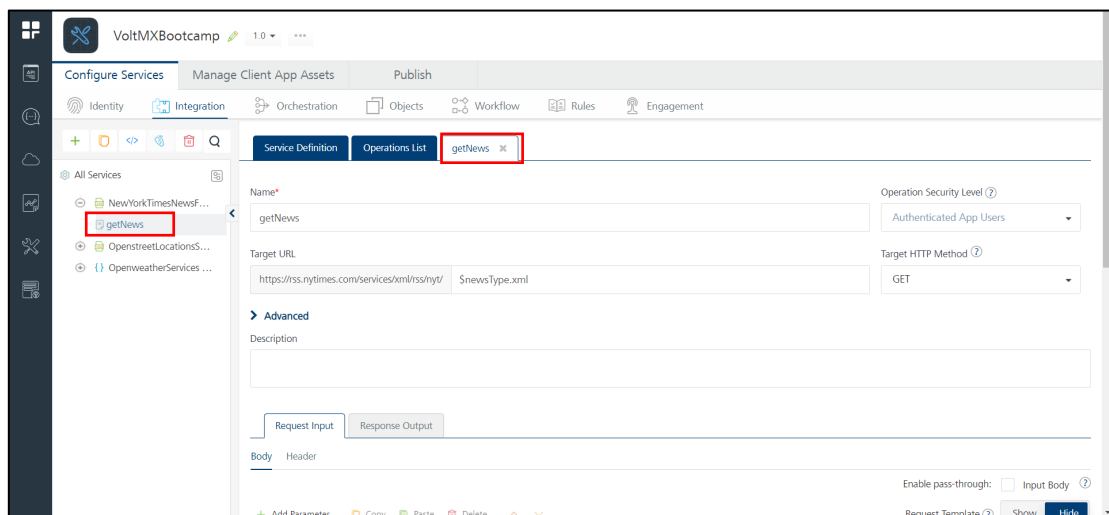
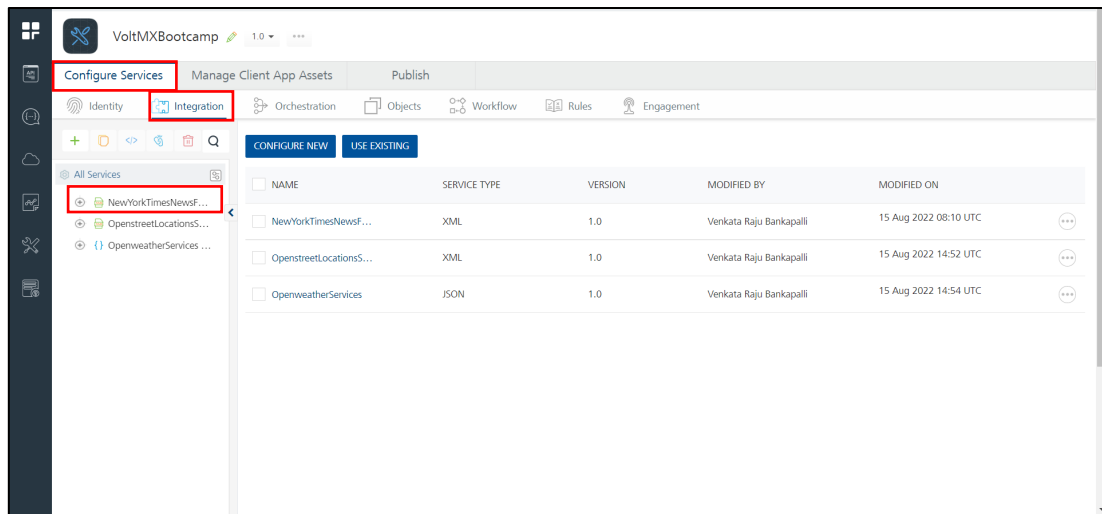
JavaScript を使用して Foundry でプリプロセッサを実装および構成する

注意事項:

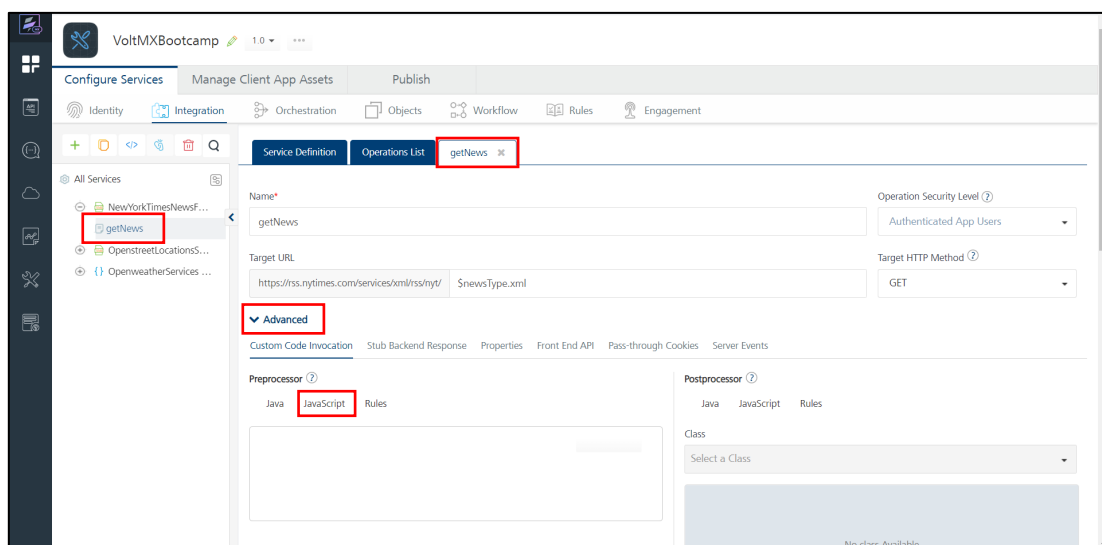
- 前のレッスンで作業していた **Foundry** プロジェクトの上で、以下に説明するステップを引き続き実行します。
- Foundry コンソールにログインする
- Foundry アプリ **VoltMXBootcamp** を開く



- **Configure Services > Integration > NewYorkTimesNewsFeed > getNews** からニュース Integration サービスの操作を開く。



- **Advanced > Custom Code Invocation** を展開します。
- **Preprocessor** セクションで **JavaScript** を選択します。



- 以下のコードスニペットを追加します。

```
function processInput(){
  var strNewsType = request.getParameter('newsType');

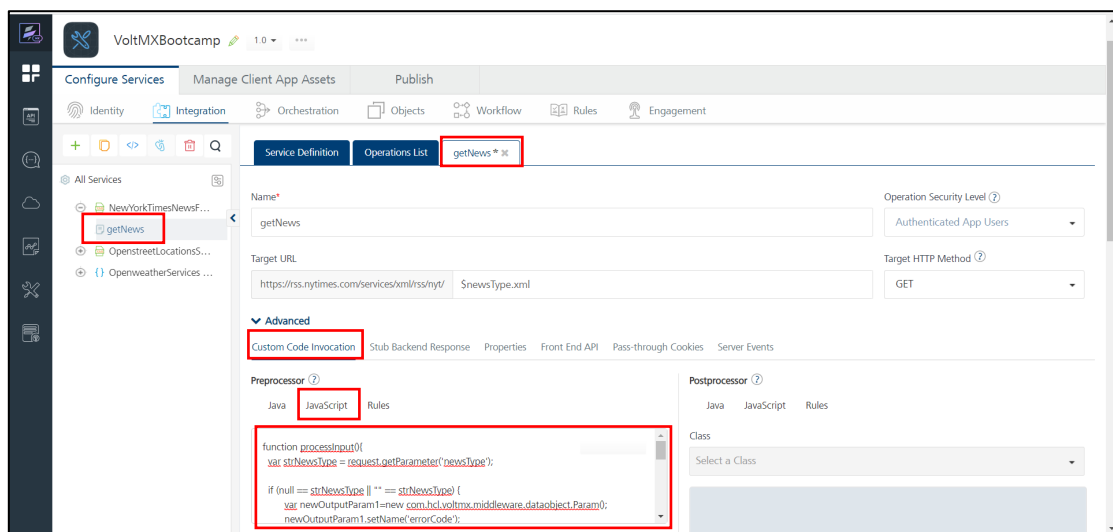
  if (null == strNewsType || "" == strNewsType) {
    var newOutputParam1=new com.hcl.voltmx.middleware.dataobject.Param();
    newOutputParam1.setName('errorCode');
    newOutputParam1.setValue('ERR001');
    result.setParam(newOutputParam1);

    var newOutputParam2=new com.hcl.voltmx.middleware.dataobject.Param();
    newOutputParam2.setName('errorMessage');
    newOutputParam2.setValue('Required input parameter is not found.');
```

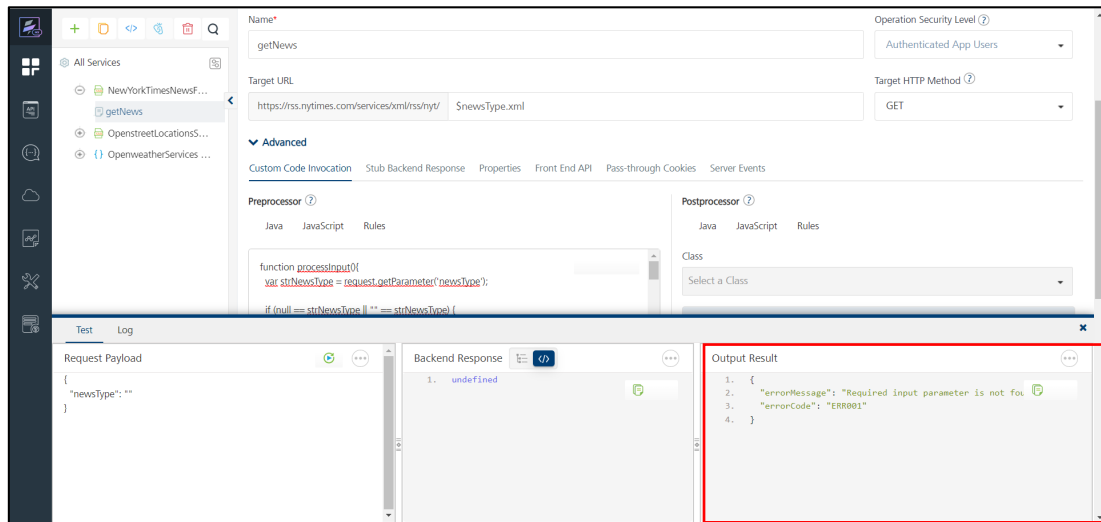
result.setParam(newOutputParam2);

```
//request.addRequestParam_('HeaderName', 'HeaderValue');

    return false;
  } else {
    return true;
  }
}
processInput();
```

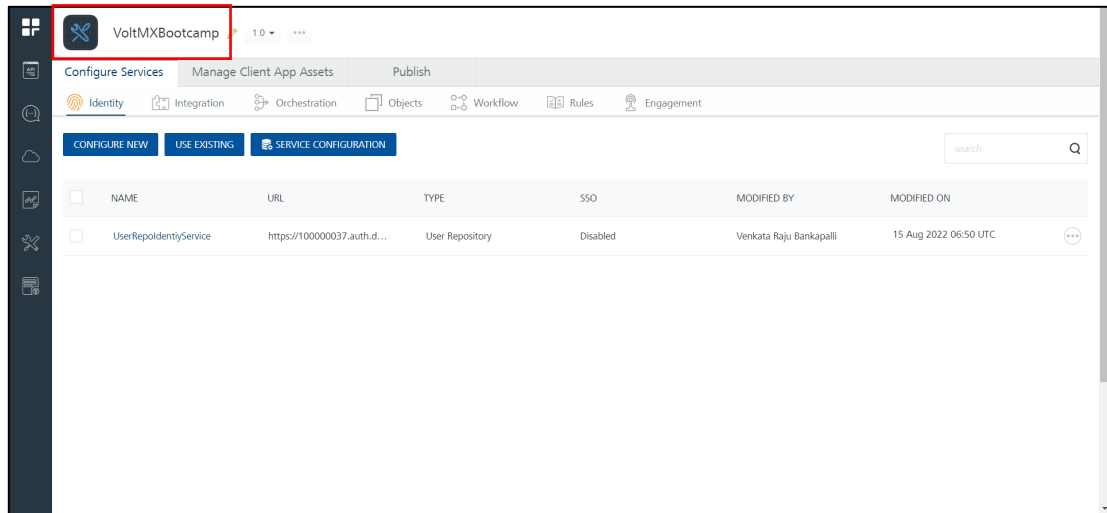


- **newsType** リクエスト入力の **TEST VALUE** を削除し、**SAVE AND FETCH RESPONSE** をクリックします。
- **Output Result** を確認します。
- プリプロセッサから返されたエラー応答、および Foundry がニュースサービスへのサービス呼び出しを行わなかったことがわかります。

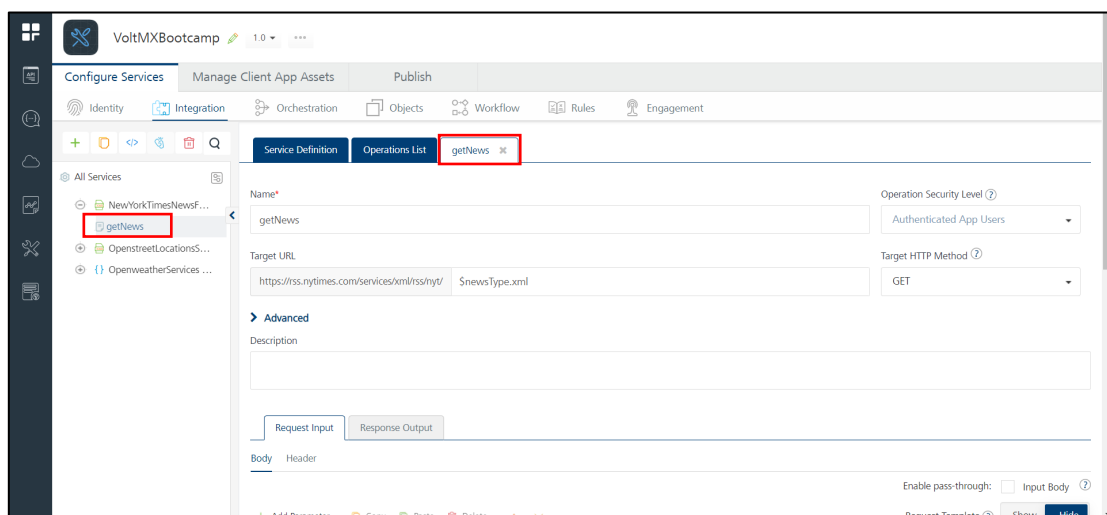
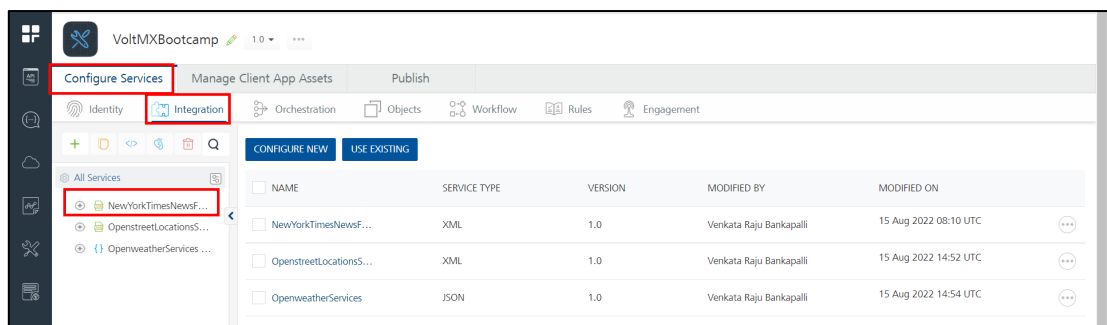


JavaScript を使用した Foundry のポストプロセッサの実装と設定

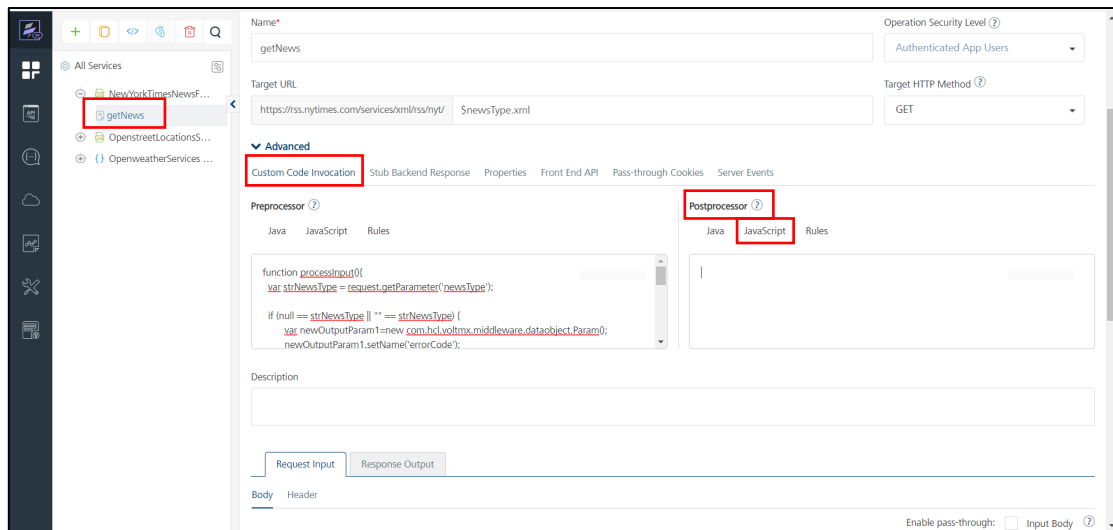
- Foundry のコンソールにログインします。
- Foundry アプリ **VoltMXBootcamp** を開きます。



- **Configure Services > Integration > NewYorkTimesNewsFeed > getNews** からニュース Integration サービスの操作を開きます。



- **Advanced > Custom Code Invocation** を展開します。
- **Postprocessor** セクションで、**JavaScript** を選択します。



- 以下のコードスニペットを追加します。

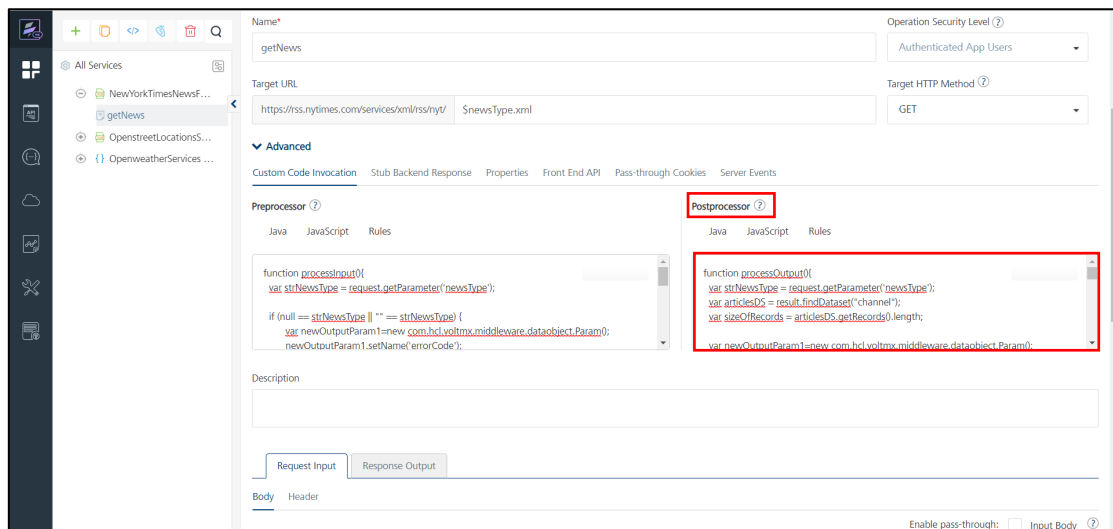
```
function processOutput(){
  var strNewsType = request.getParameter('newsType');
  var articlesDS = result.findDataset("channel");
  var sizeOfRecords = articlesDS.getRecords().length;

  var newOutputParam1=new com.hcl.voltmx.middleware.dataobject.Param();
  newOutputParam1.setName('newsType');
  newOutputParam1.setValue(strNewsType);
  result.setParam(newOutputParam1);

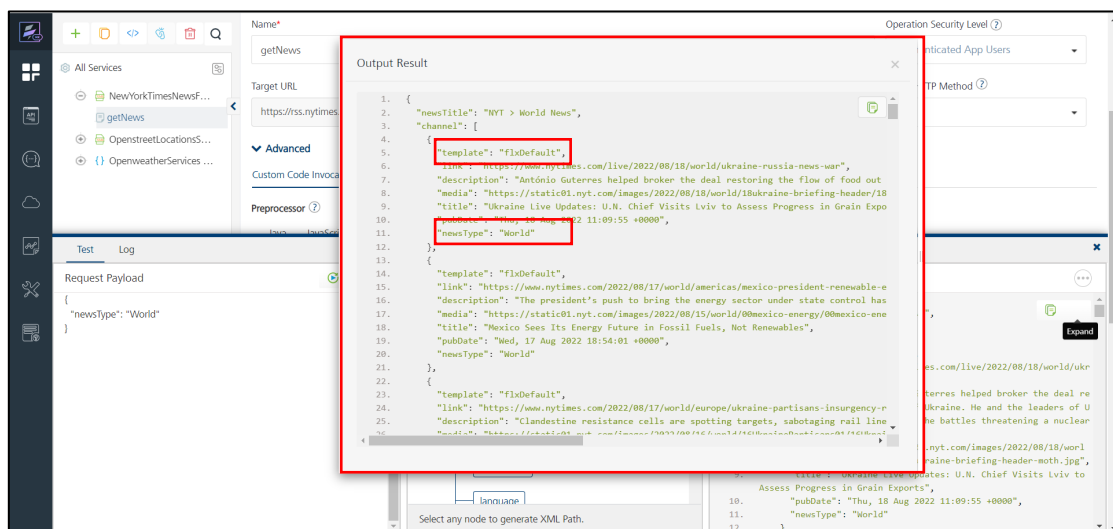
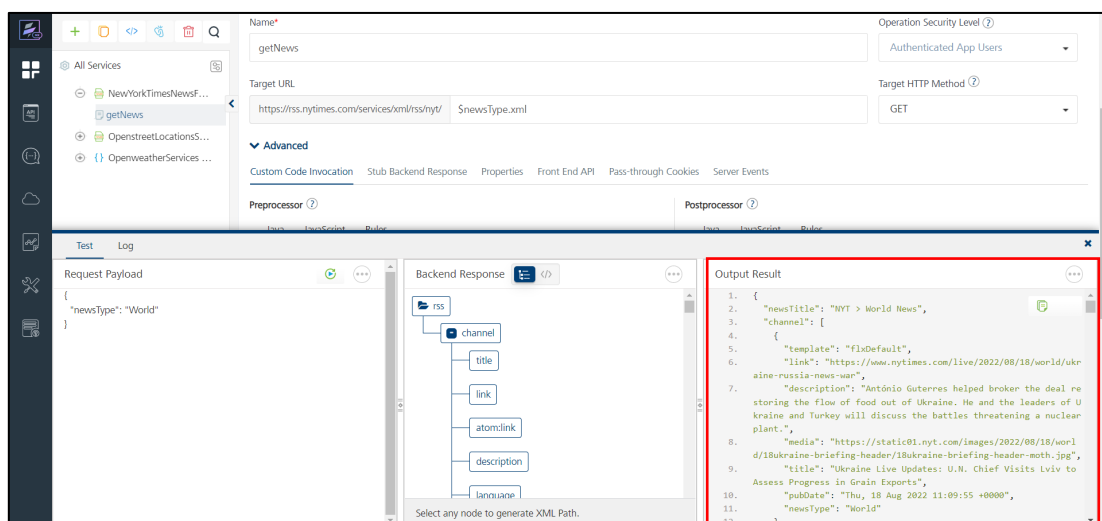
  var newOutputParam2= new com.hcl.voltmx.middleware.dataobject.Param();
  newOutputParam2.setName('articleCount');
  newOutputParam2.setValue(sizeOfRecords);
  result.setParam(newOutputParam2);

  for (i = 0; i < sizeOfRecords; i++) {
    var datasetRecord = articlesDS.getRecord(i);
    var varParam = new com.hcl.voltmx.middleware.dataobject.Param();
    varParam.setName('newsType');
    varParam.setValue(request.getParameter('newsType'));
    datasetRecord.setParam(varParam);

    //var varParamTemplate = new com.hcl.voltmx.middleware.dataobject.Param();
    //varParamTemplate.setName('template');
    //varParamTemplate.setValue('flxDefault');
    //varParamTemplate.setValue('flx' + strNewsType);
    //if (strNewsType === "Sports")
    //{
    //  varParamTemplate.setValue('flxSports');
    //}
    //else if (strNewsType === "World")
    //{
    //  varParamTemplate.setValue('flxWorld');
    //}
    //else
    //{
    //  varParamTemplate.setValue('flxDefault');
    //}
    //datasetRecord.setParam(varParamTemplate);
  }
}
processOutput();
```

- **SAVE AND FETCH RESPONSE** をクリックします。
- **Output Result** を確認します。
- **newType** と **template** パラメータが **channel** の各レコードに追加され、**newType** パラメータもレスポンスに直接パラメータとして追加されていることがわかります。

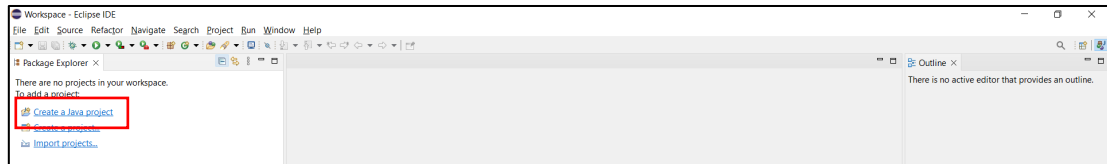


Java を使用した Foundry でのプリプロセッサの実装と設定

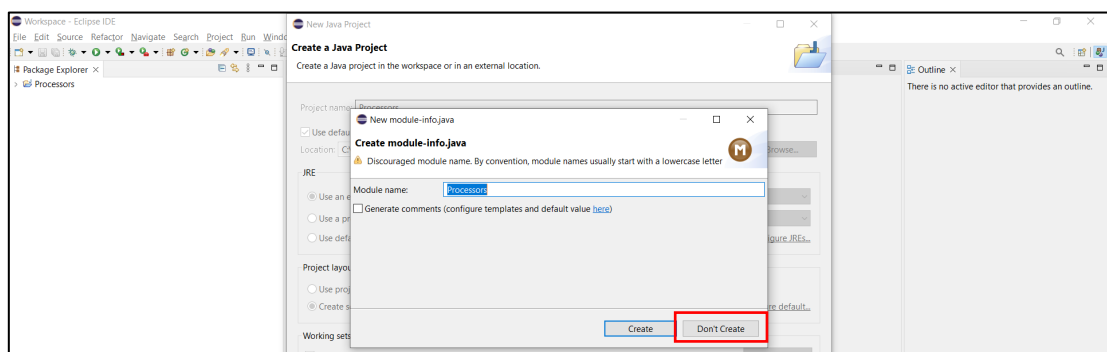
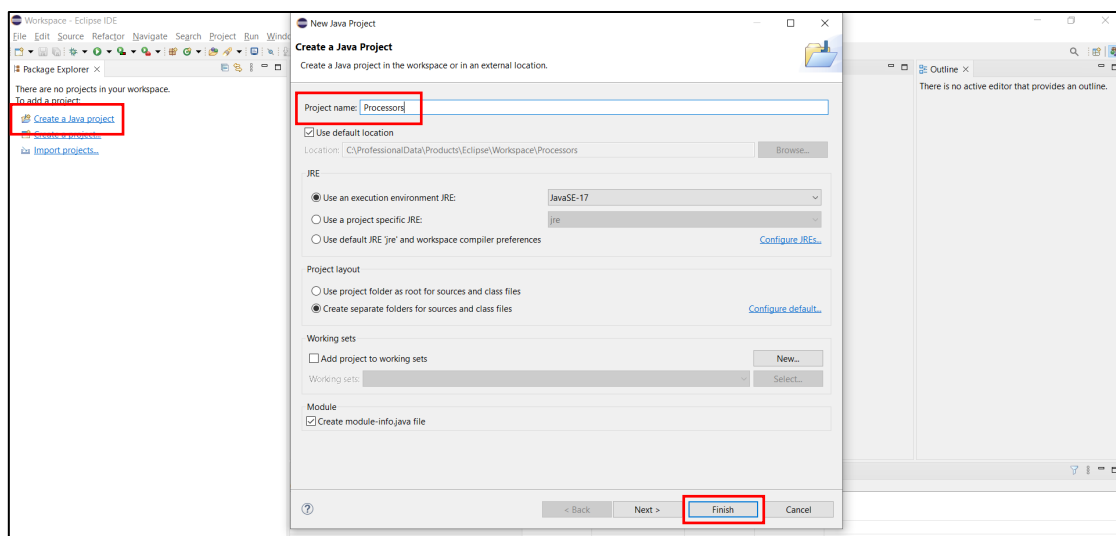
- 簡単な Java プロジェクトを作成できる任意の IDE を開いてください。

注意：このチュートリアルのスクリリーンショットをキャプチャするために Eclipse IDE を使用しました。

- Eclipse IDE を開きます。

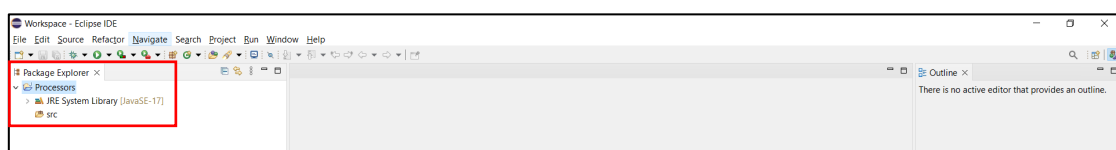


- Processors という名前で新しい Java Project を作成します。

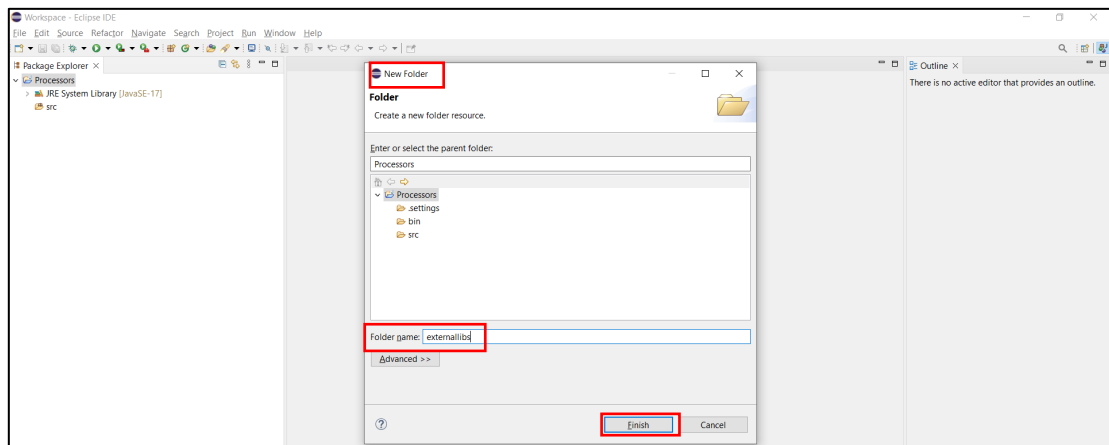


注：New module-info.java のウィンドウが表示されたら、Don't Create をクリックしてください。

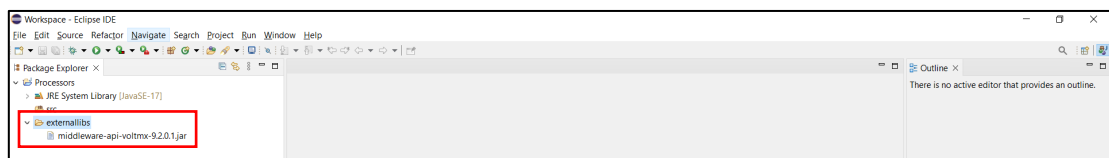
- 新しい Java プロジェクトが作成されたことが確認できます。



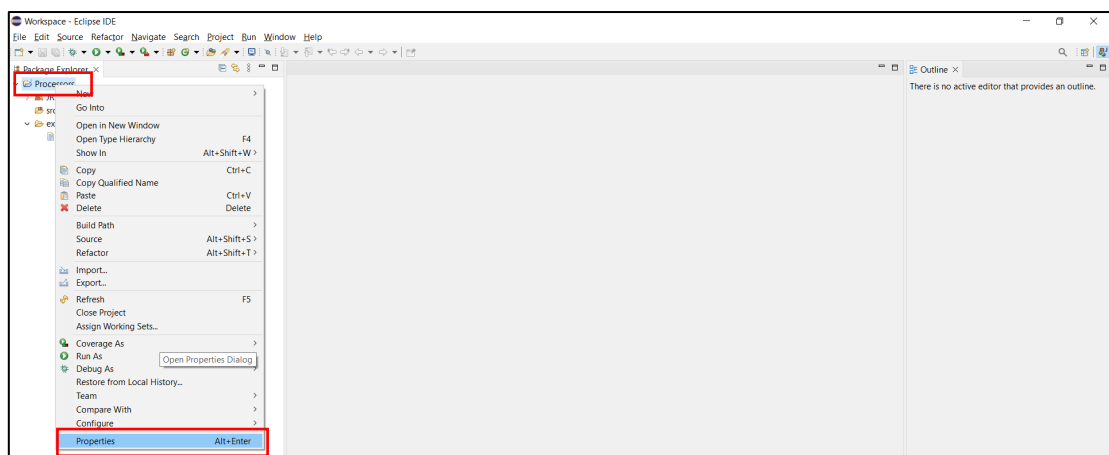
- このレッスンの添付ファイルから、zip ファイル **middleware-api-voltmx.zip** をダウンロードします。
- middleware-api-voltmx.zip** を解凍します。**middleware-api-voltmx-9.2.0.1.jar** という jar ファイルがあります。
- Java プロジェクトに **externallibs** フォルダを作成します。



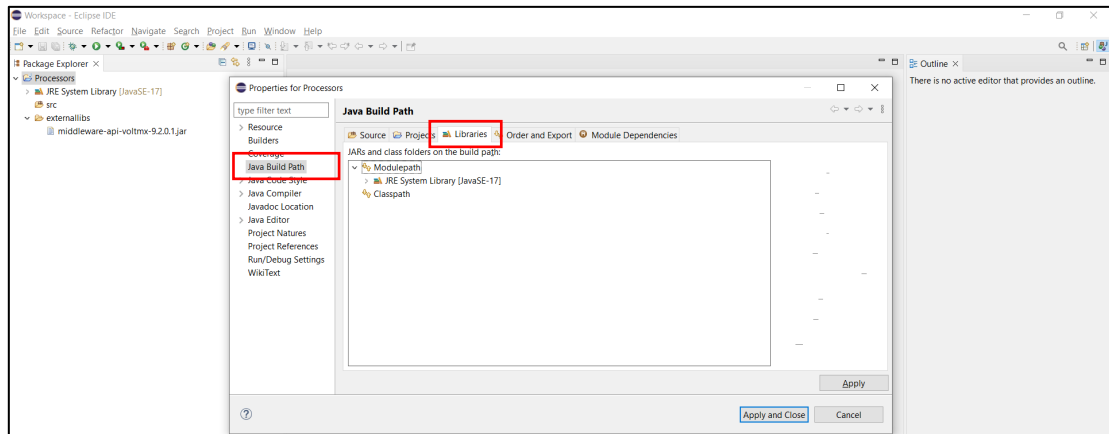
- jar ファイル **middleware-api-voltmx-9.2.0.1.jar** をこの **externallibs** フォルダにコピーします。



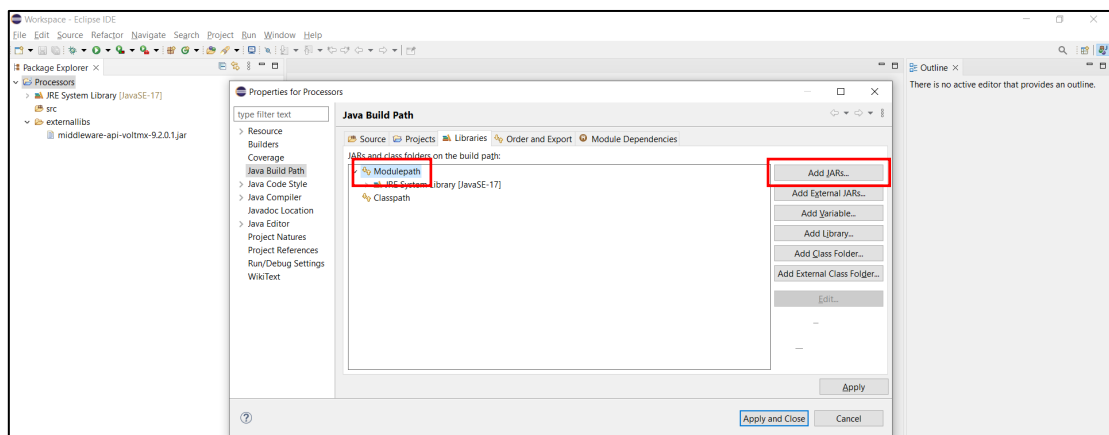
- プロジェクトのビルドパスに **middleware-api-voltmx-9.2.0.1.jar** ファイルを追加します。
- プロジェクトを右クリックし、プロジェクトのプロパティにアクセスします。



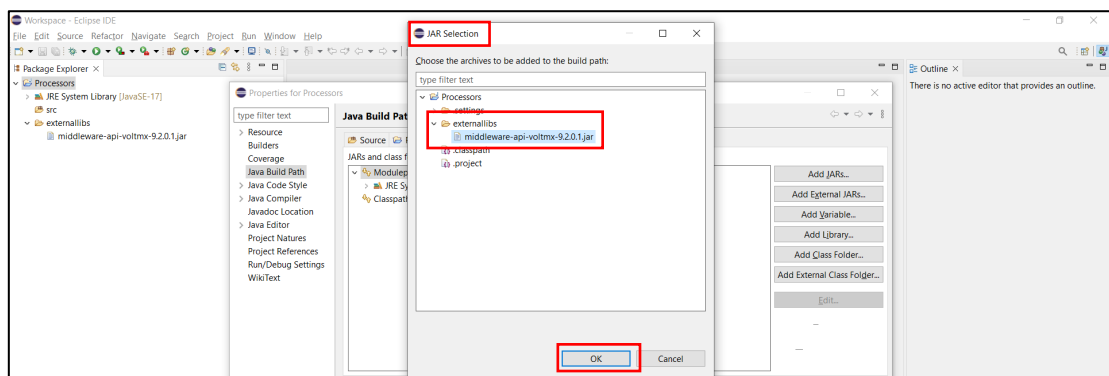
- **Java Build Path** をクリックし、**Libraries** タブを選択します



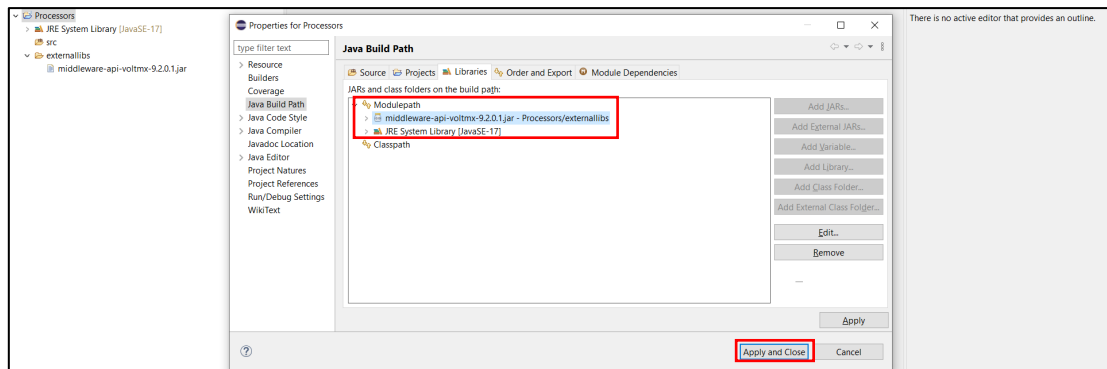
- **Modulepath** をクリックし、**Add JARs...** をクリックします。



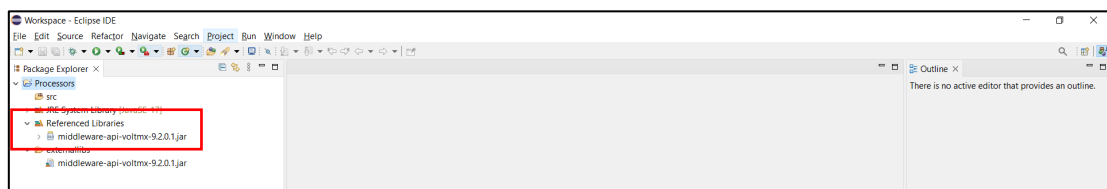
- JAR Selection ウィンドウから、**externallibs** フォルダにある **middleware-api-voltxmx-9.2.0.1.jar** を選択して OK をクリックします。



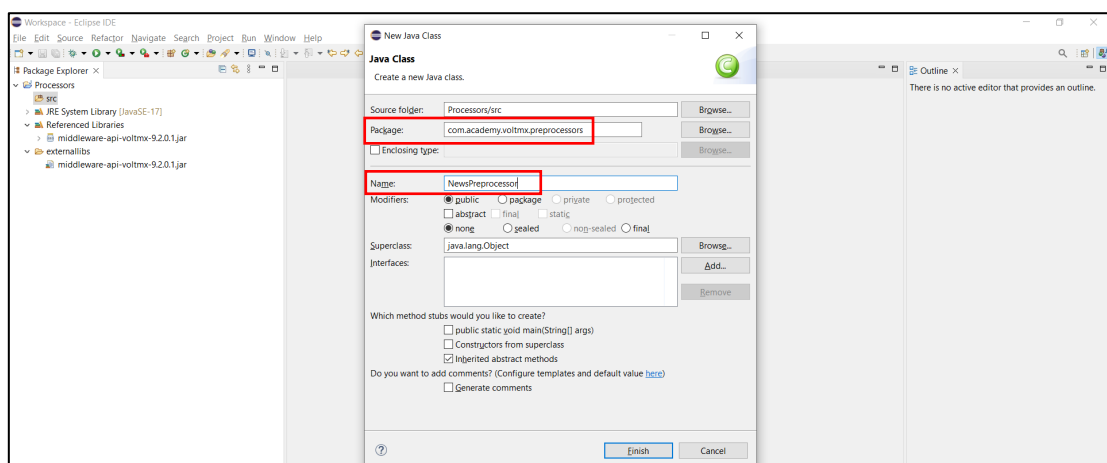
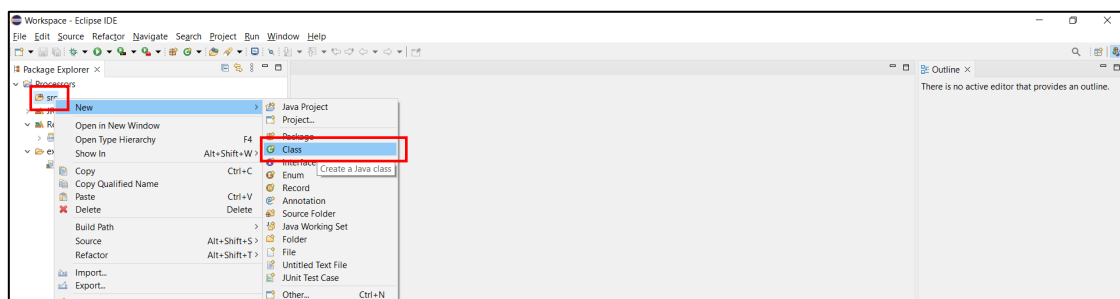
- **modulepath** の下に **middleware-api-voltmx-9.2.0.1.jar** が追加されていることが確認できます。
Apply and Close をクリックします。



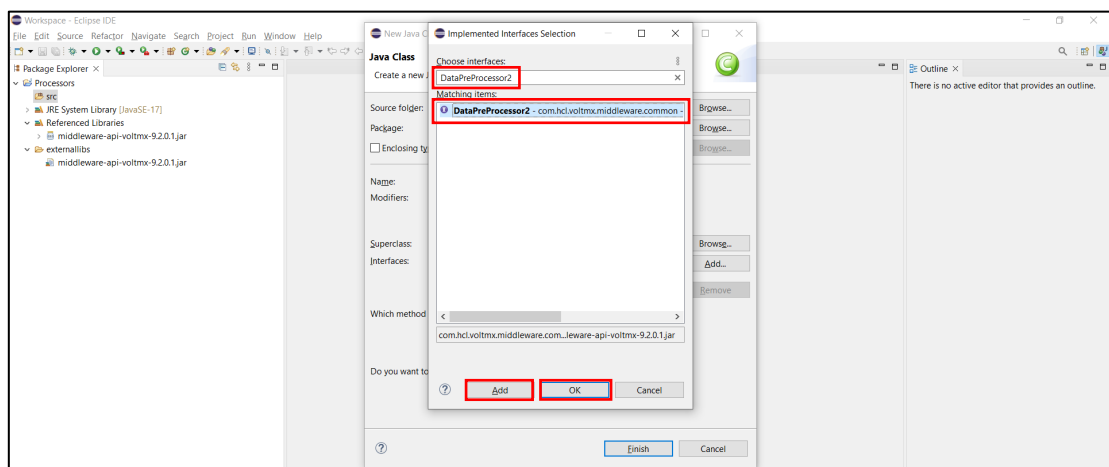
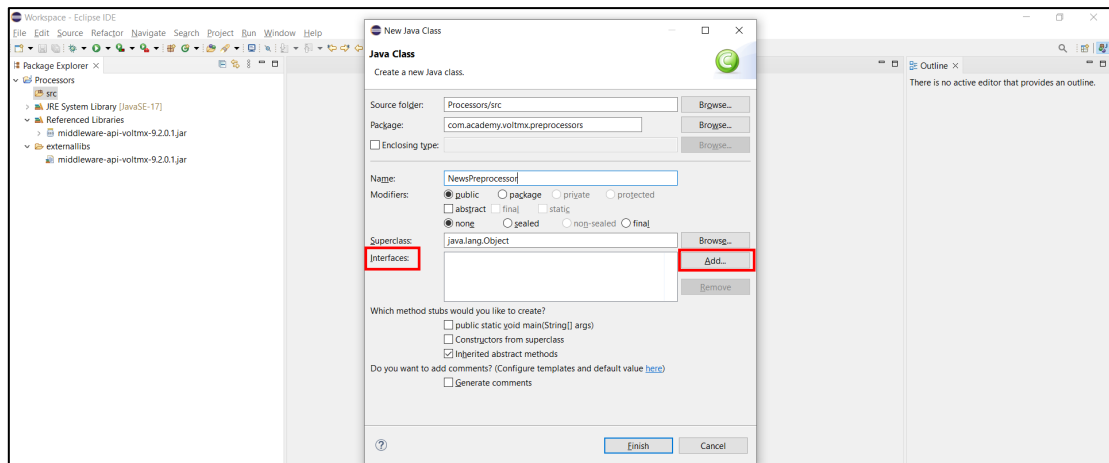
- プロジェクトの **Referenced Libraries** に **middleware-api-voltmx-9.2.0.1.jar** という JAR が表示されます。



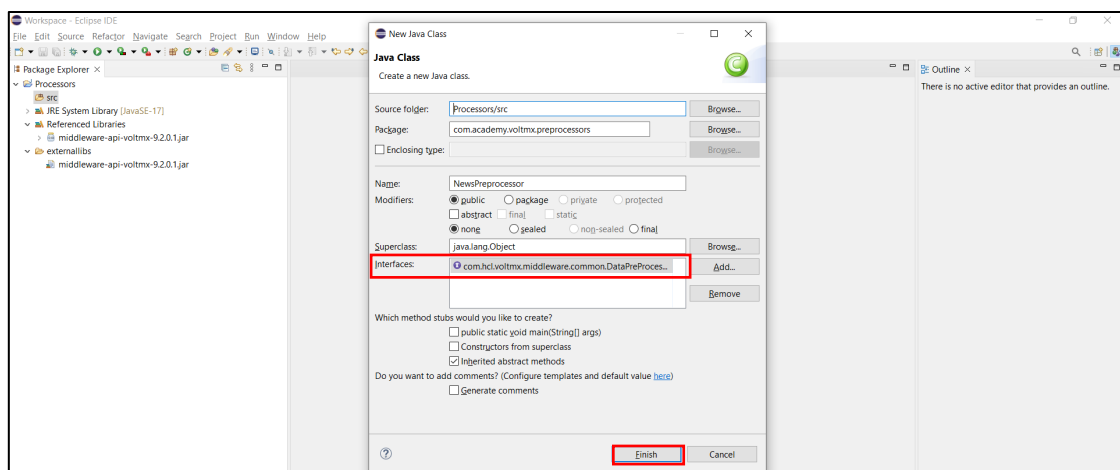
- **Name** を **NewsPreprocessor**、**Package name** を **com.academy.voltmx.preprocessors** として、新しい Java クラスを作成します。
- プロジェクト内の **src** フォルダを右クリックし、**New > Class** を選択します。



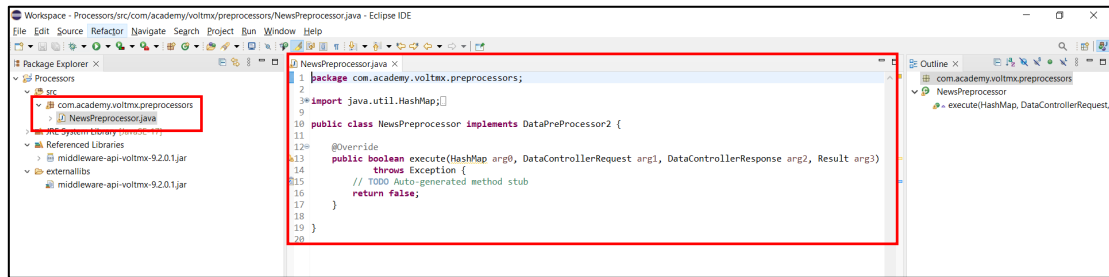
- インターフェイス に対して 追加 オプションをクリックし、**DataPreProcessor2** を検索し、それを選択して **Add** をクリックし、**OK** をクリックします。



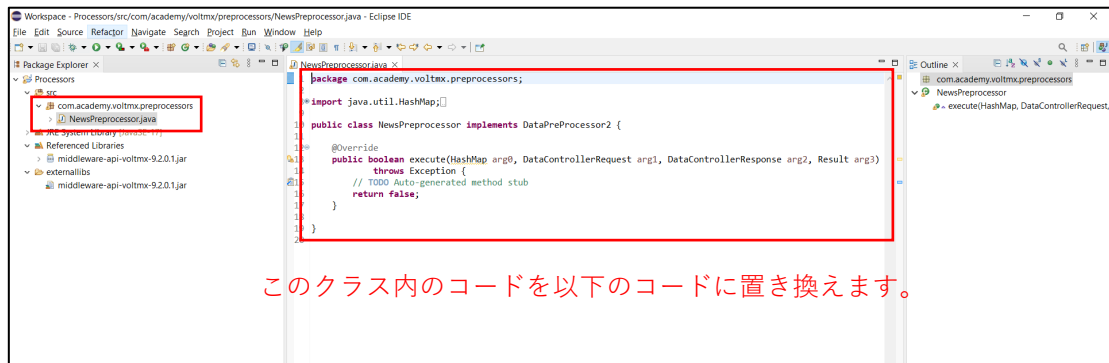
- **DataPreprocessor2** インターフェイスが **Interfaces** セクションに追加されていることが確認できます。 **Finish** をクリックします。



- プロジェクトに新しいクラスが作成されることが確認できます。



- このクラスのコードを以下のコードスニペットに置き換えます。



```
package com.academy.voltmx.preprocessors;
```

```
import java.util.HashMap;
```

```
import com.hcl.voltmx.middleware.common.DataPreProcessor2;
```

```
import com.hcl.voltmx.middleware.controller.DataControllerRequest;
```

```
import com.hcl.voltmx.middleware.controller.DataControllerResponse;
```

```
import com.hcl.voltmx.middleware.dataobject.Param;
```

```
import com.hcl.voltmx.middleware.dataobject.Result;
```

```
public class NewsPreprocessor implements DataPreProcessor2 {
```

```
    @Override
```

```
    public boolean execute(HashMap inputParameters, DataControllerRequest dataControllerRequest,
        DataControllerResponse dataControllerResponse, Result result) throws Exception {
```

```
        String strNewsType = (String) inputParameters.get("newsType");
```

```
        boolean continueExecution = false;
```

```
        if (null != strNewsType && !"".equalsIgnoreCase(strNewsType.trim())) {
            continueExecution = true;
```

```
        } else {
```

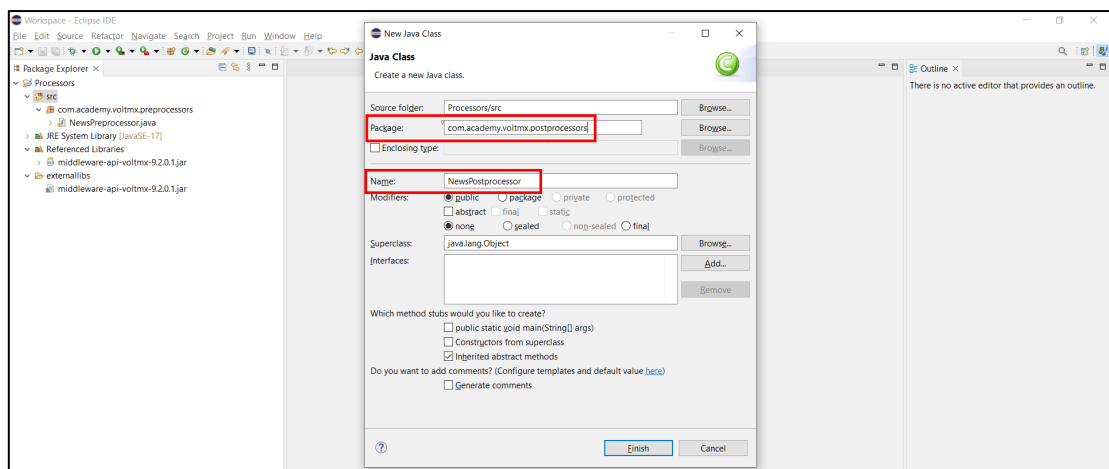
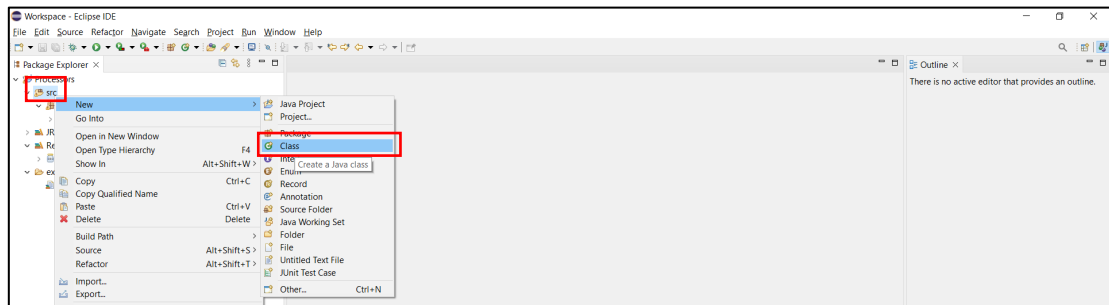
```
            Param newOutputParam1 = new Param();
            newOutputParam1.setName("errorCode");
            newOutputParam1.setValue("ERR001");
            result.addParam(newOutputParam1);
```

```
            Param newOutputParam2 = new Param();
            newOutputParam2.setName("errorMessage");
            newOutputParam2.setValue("Required input parameter is not found.");
            result.addParam(newOutputParam2);
```

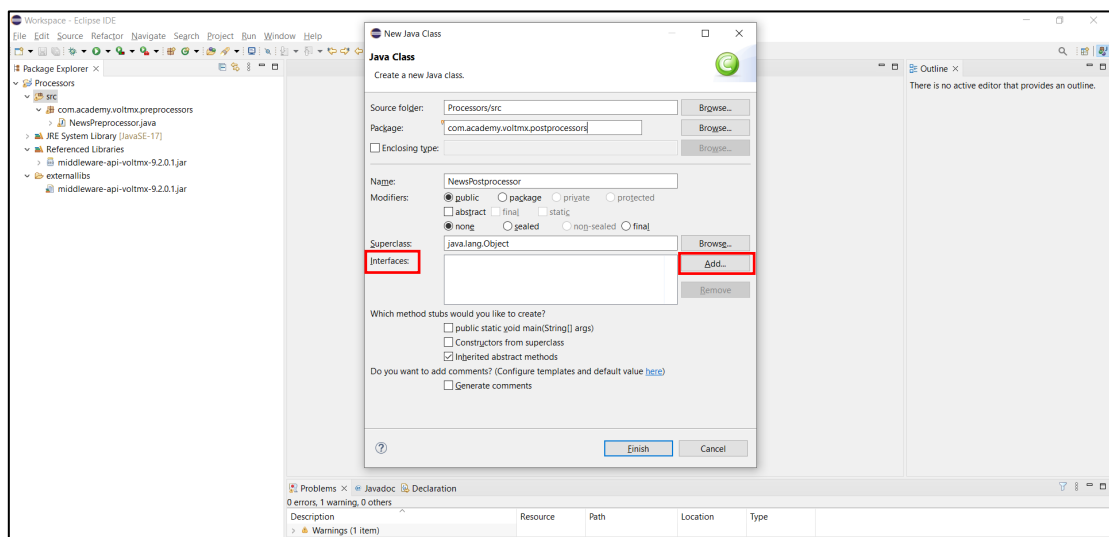
```
        // dataControllerRequest.addRequestParam_("HeaderName", "HeaderValue");
    }
    return continueExecution;
}
}
```

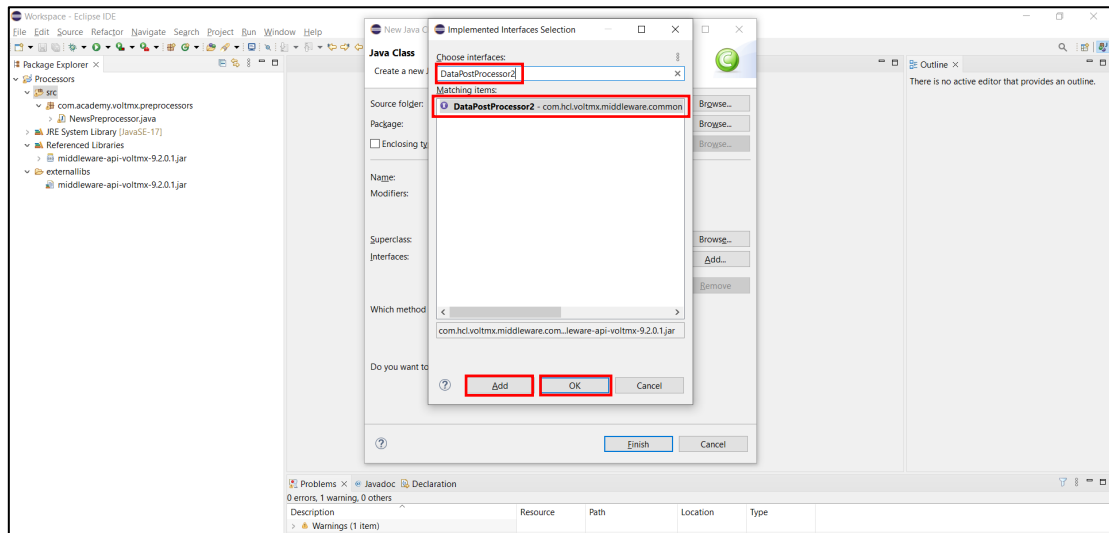

Java を使用した Foundry のポストプロセッサの実装と設定

- **Name** を **NewsPostprocessor**、**Package name** を **com.academy.voltmx.postprocessors** として新しい Java クラスを作成します。
- プロジェクト内の **src** フォルダを右クリックし、**New > Class** を選択します。

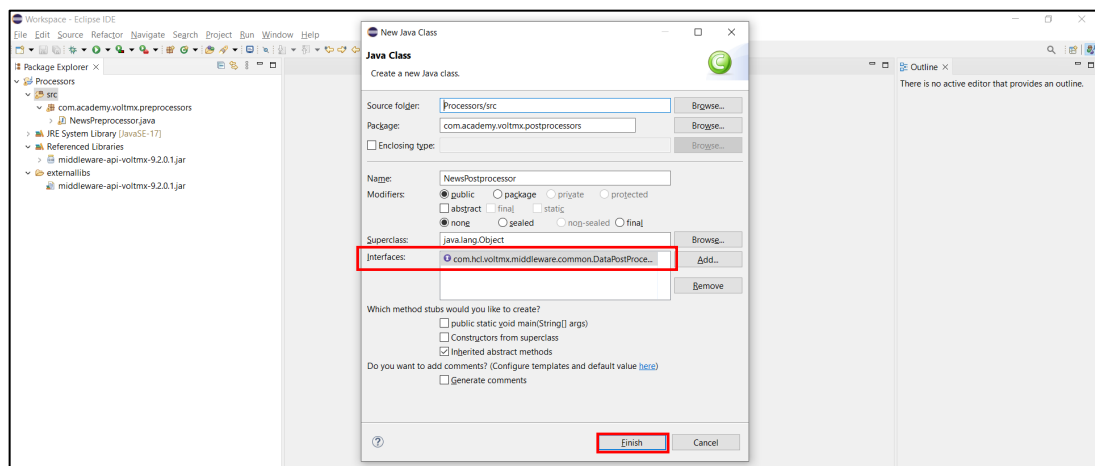


- インターフェイス に対して **Add** オプションをクリックし、**DataPostProcessor2** を検索し、それを選択して **Add** をクリックし、**OK** をクリックします。

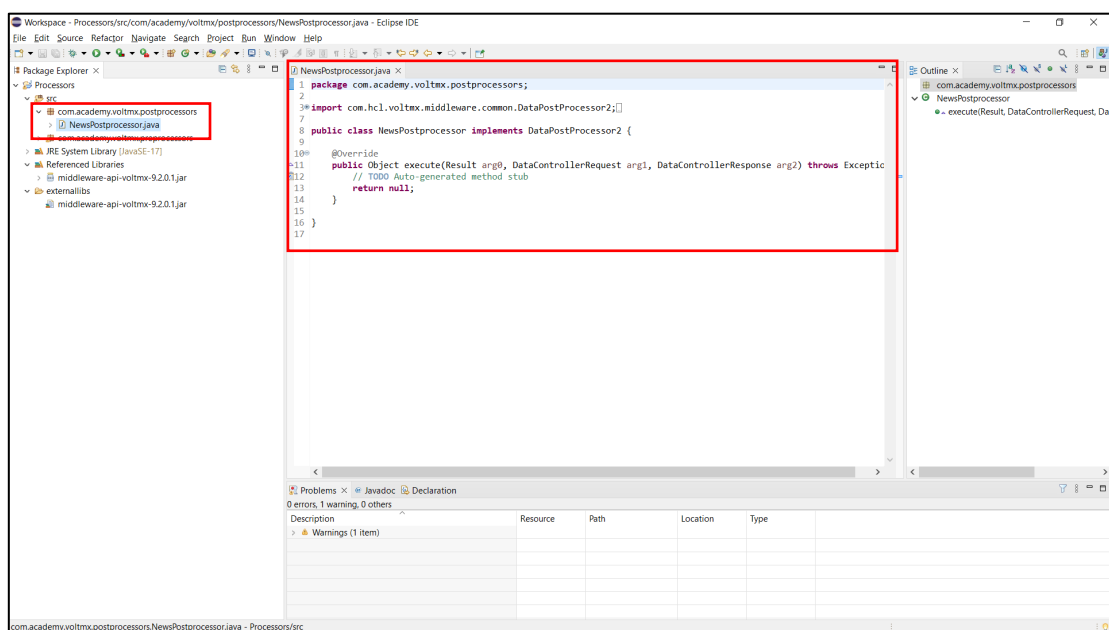




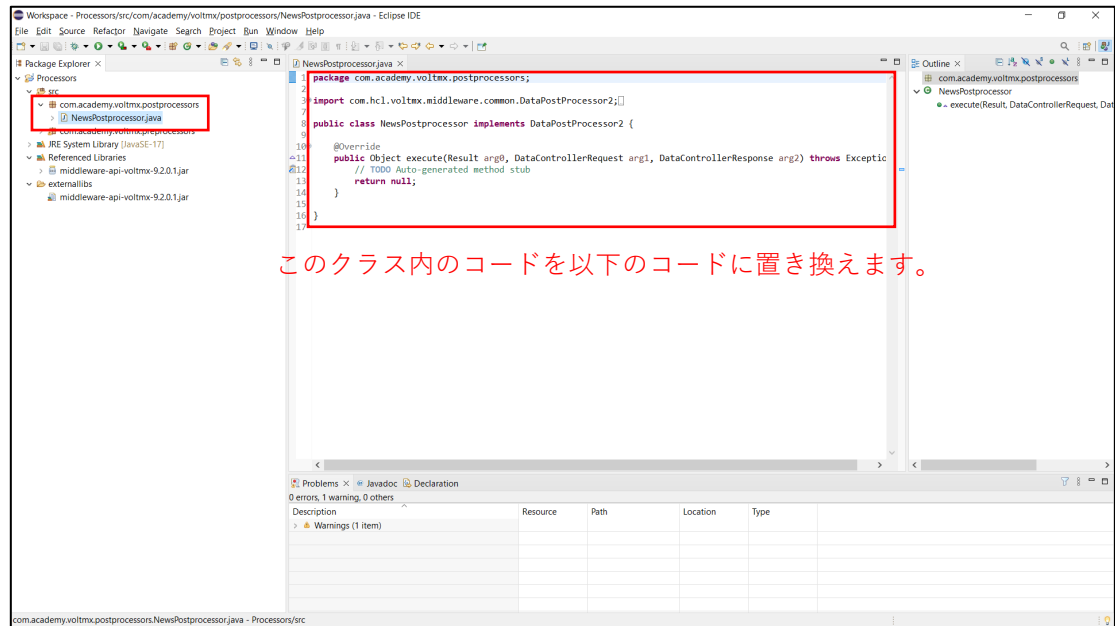
- **DataPostprocessor2** インターフェイスが **Interfaces** セクションに追加されていることが確認できます。**Finish** をクリックします。



- プロジェクトに新しいクラスが作成されることが確認できます。



- このクラス内のコードを以下のコードに置き換えます。



このクラス内のコードを以下のコードに置き換えます。

```
package com.academy.voltmx.postprocessors;
```

```
import com.hcl.voltmx.middleware.common.DataPostProcessor2;
import com.hcl.voltmx.middleware.controller.DataControllerRequest;
import com.hcl.voltmx.middleware.controller.DataControllerResponse;
import com.hcl.voltmx.middleware.dataobject.Dataset;
import com.hcl.voltmx.middleware.dataobject.Param;
import com.hcl.voltmx.middleware.dataobject.Record;
import com.hcl.voltmx.middleware.dataobject.Result;
```

```
public class NewsPostprocessor implements DataPostProcessor2 {
```

```
    @Override
```

```
    public Object execute(Result result,
                           DataControllerRequest dataControllerRequest,
                           DataControllerResponse dataControllerResponse) throws Exception {
        // TODO Auto-generated method stub
```

```
        String strNewsType = dataControllerRequest.getParameter("newsType");
        Dataset articlesDS = result.getDatasetByld("channel");
        int sizeOfRecords = articlesDS.getAllRecords().size();
```

```
        Param newOutputParam1=new Param();
        newOutputParam1.setName("newsType");
        newOutputParam1.setValue(strNewsType);
        result.addParam(newOutputParam1);
```

```
        Param newOutputParam2= new Param();
        newOutputParam2.setName("articleCount");
        newOutputParam2.setValue(Integer.toString(sizeOfRecords));
        result.addParam(newOutputParam2);
```

```
        for (int i = 0; i < sizeOfRecords; i++) {
```

```

        Record datasetRecord = articlesDS.getRecord(i);
        Param varParam = new Param();
        varParam.setName("newsType");
        varParam.setValue(strNewsType);
        datasetRecord.addParam(varParam);

        //var varParamTemplate = new com.hcl.voltmx.middleware.dataobject.Param();
        //varParamTemplate.setName('template');
        //varParamTemplate.setValue('flxDefault');
        //varParamTemplate.setValue('flx' + strNewsType);
        //if (strNewsType === "Sports")
        //{
        //    varParamTemplate.setValue('flxSports');
        //}
        //else if (strNewsType === "World")
        //{
        //    varParamTemplate.setValue('flxWorld');
        //}
        //else
        //{
        //    varParamTemplate.setValue('flxDefault');
        //}
        //datasetRecord.setParam(varParamTemplate);

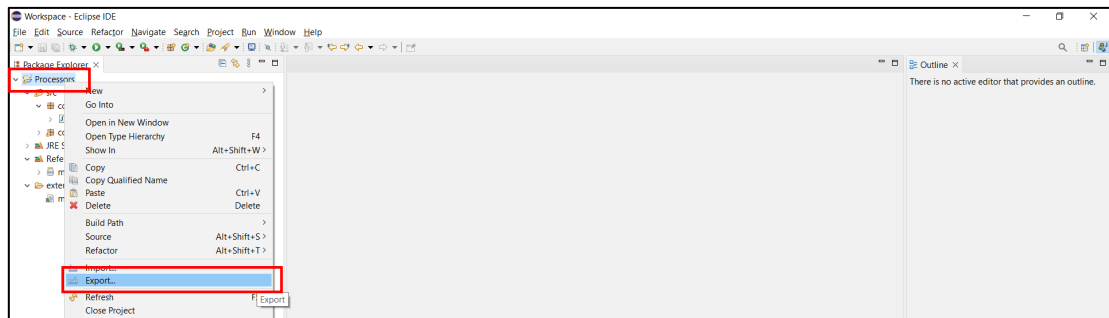
        //Here is a sample code for adding a new dataset with records into result
        //Dataset savingsAccounts = new Dataset("savingsAccounts");
        //ArrayList<Record> savAccRecords = new ArrayList<>();
        //Record bodyDSRecord = new Record();
        //Param imgBankLogo = new Param();
        //imgBankLogo.setName("imgBankLogo");
        //imgBankLogo.setValue("bankofamerica.png");
        //bodyDSRecord.addParam(imgBankLogo);
        //savAccRecords.add(bodyDSRecord);
        //savingsAccounts.addAllRecords(savAccRecords);
        //result.addDataset(savingsAccounts);

    }
    return result;
}
}

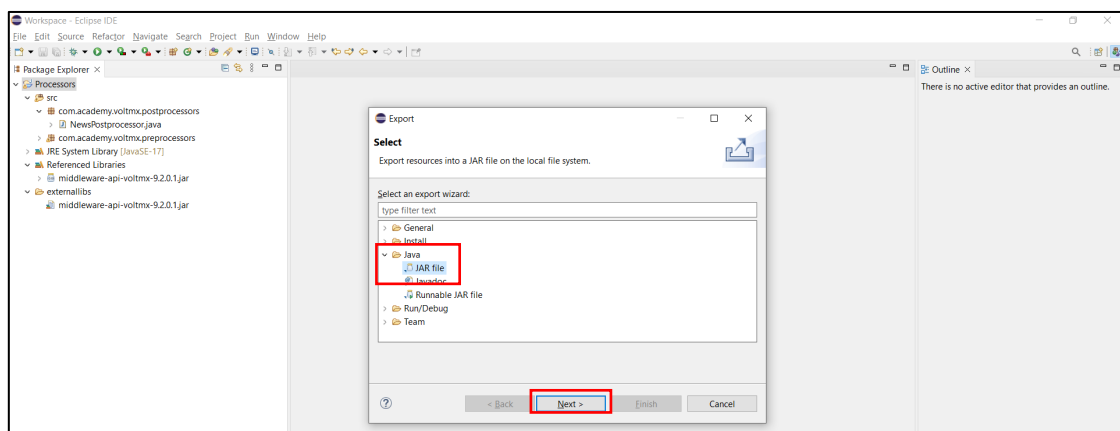
```

Java プロセッサを Foundry に統合する

- Java プロジェクトから JAR をコンパイルして生成します。
- プロジェクトで右クリックして、**Export** をクリックします。

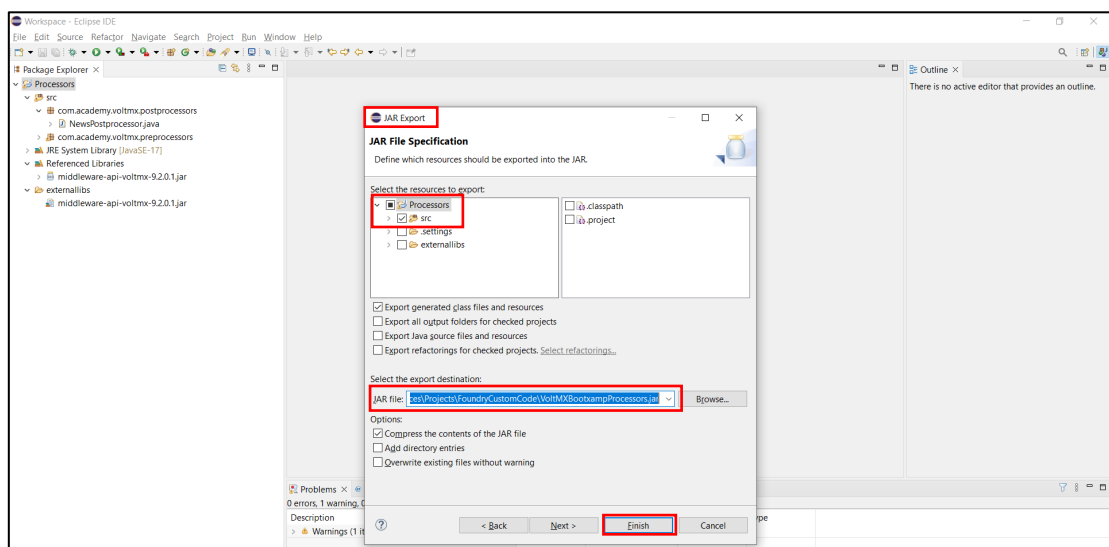


- **Export** で **JAR file** オプションを選択し、Next をクリックします。

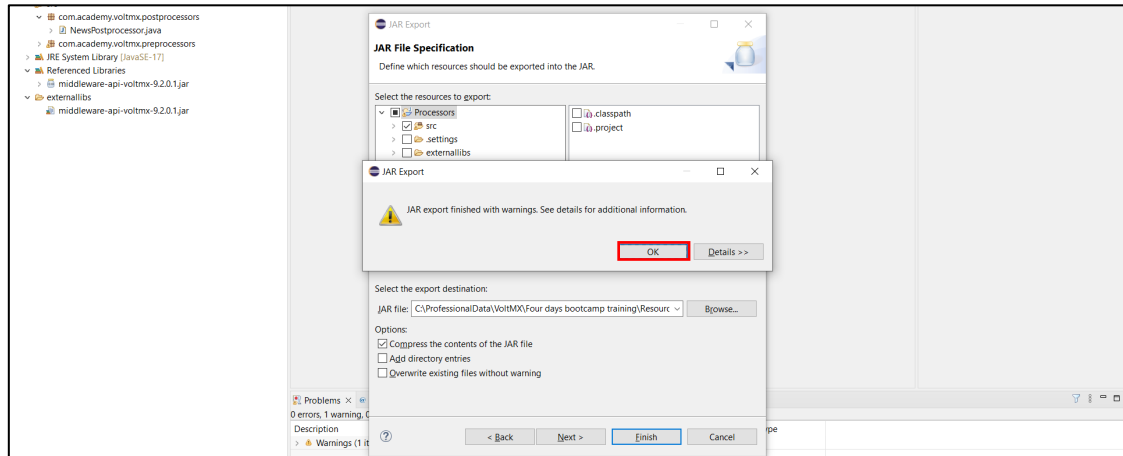


- **Select resources to export** セクションで **src** フォルダのみを選択します。JAR ファイルを生成する場所と JAR の名前を **VoltMXBootcampProcessors** として指定します。**Finish** をクリックします。

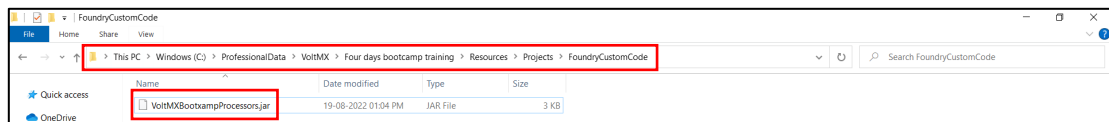
注： **Select resources to export** セクションで **externallibs** フォルダを選択しないでください。これらのライブラリはすでに Foundry の一部として存在しているためです。



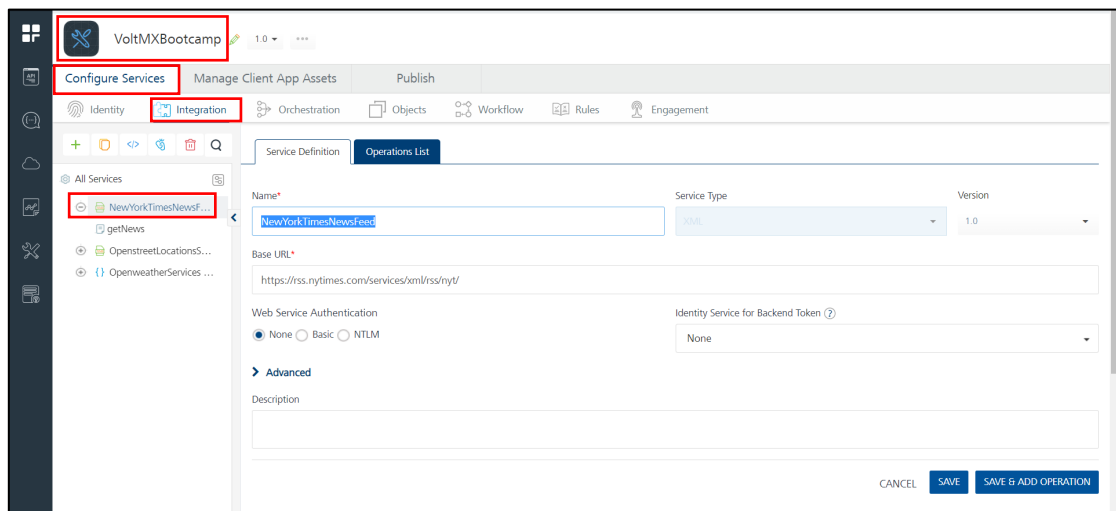
- **JAR Export** ウィンドウで **OK** をクリックします。



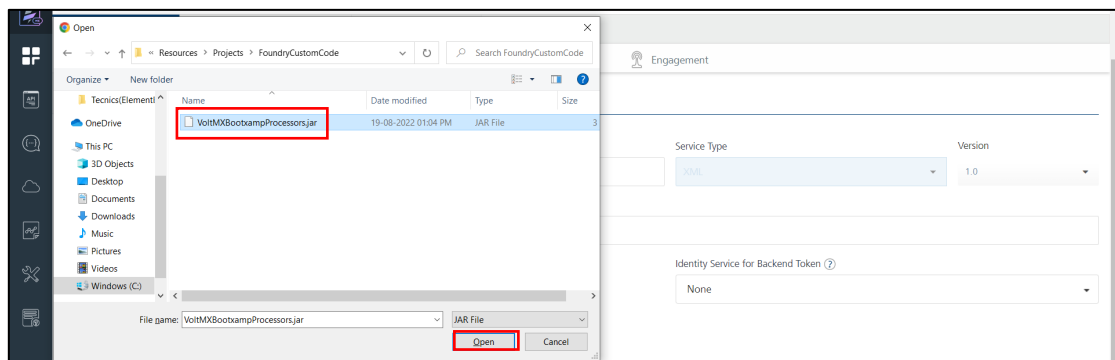
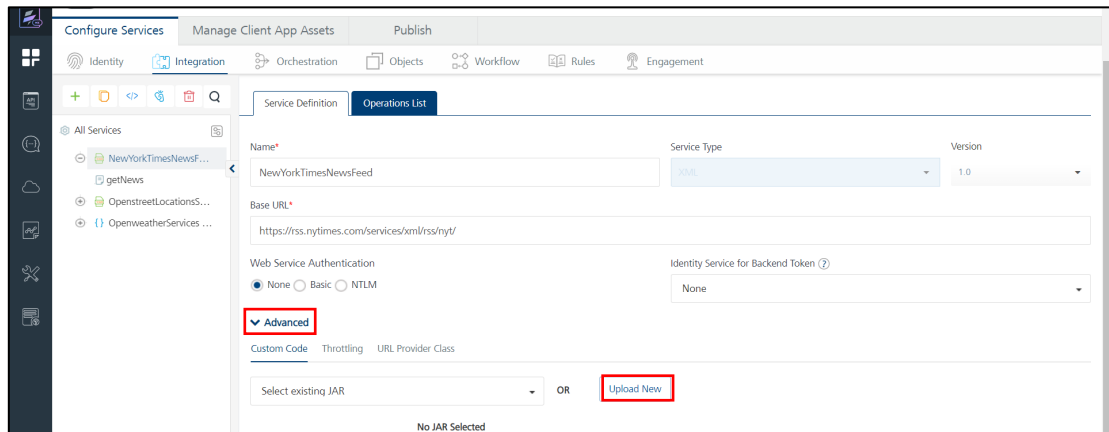
- JAR が正常に生成されたことを確認します。



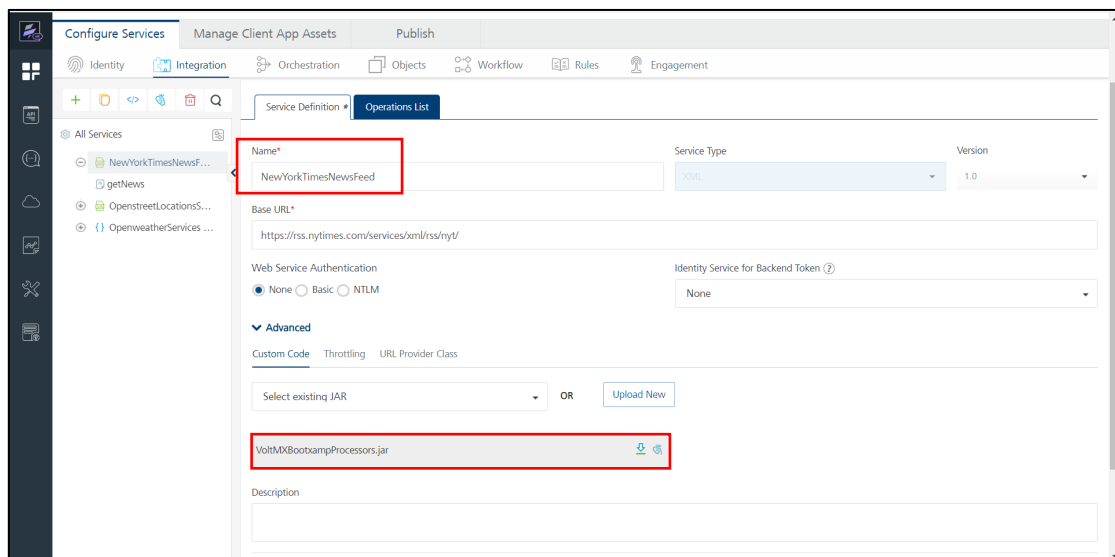
- Foundry コンソールにログインします。
- Foundry プロジェクト **VoltMXBootcamp** を開きます。
- **Configure Services > Integration > NewYorkTimesNewsFeed** に移動します。



- **Advanced** をクリックします。
- **Upload New** をクリックし、Java プロジェクトから生成した JAR ファイル **VoltMXBootcampProcessors.jar** をアップロードします。



- JAR ファイルは、**NewYorkTimesNewsFeed** の統合サービスにリンクされます。



- **SAVE** をクリックします。

Service Definition * Operations List

Name* NewYorkTimesNewsFeed Service Type JAX-RS Version 1.0

Base URL* https://rss.nytimes.com/services/xml/rss/nyt/

Web Service Authentication: None Basic NTLM

Identity Service for Backend Token: None

Advanced

Custom Code Throttling URL Provider Class

Select existing JAR OR Upload New

VoltMXBootcampProcessors.jar

Description

CANCEL **SAVE** SAVE & ADD OPERATION

- **Operation List** をクリックし、**getNew** オペレーションを開きます。

Service Definition **Operations List**

ADD OPERATION

Configured Operations

NAME	URL	MODIFIED BY	MODIFIED ON
getNews	https://rss.nytimes.com/services/xml/rss/nyt/NewsType.xml	Venkata Raju Bankapalli	19 Aug 2022 07:06 UTC

Service Definition Operations List **getNews**

Name* getNews Operation Security Level Authenticated App Users

Target URL https://rss.nytimes.com/services/xml/rss/nyt/NewsType.xml Target HTTP Method GET

Advanced

Description

Request Input Response Output

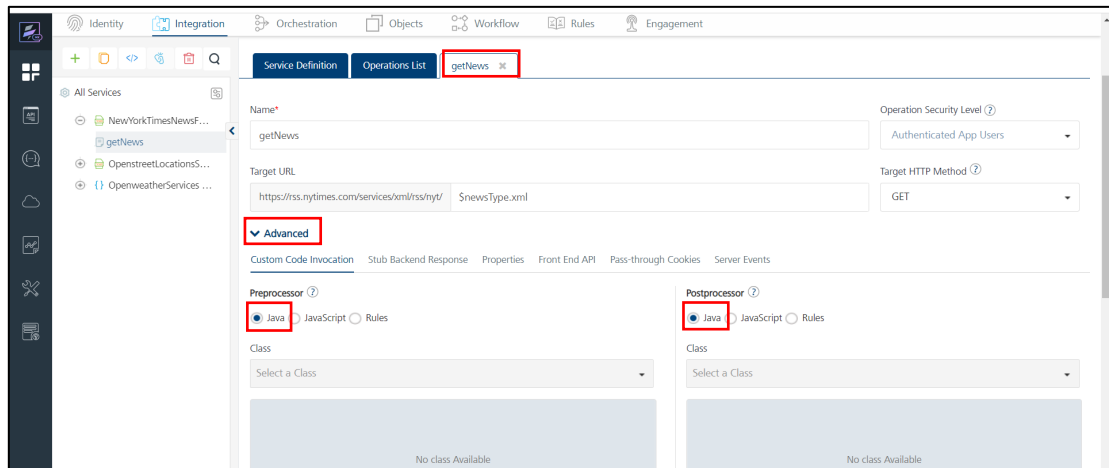
Body Header

Enable pass-through: Input Body

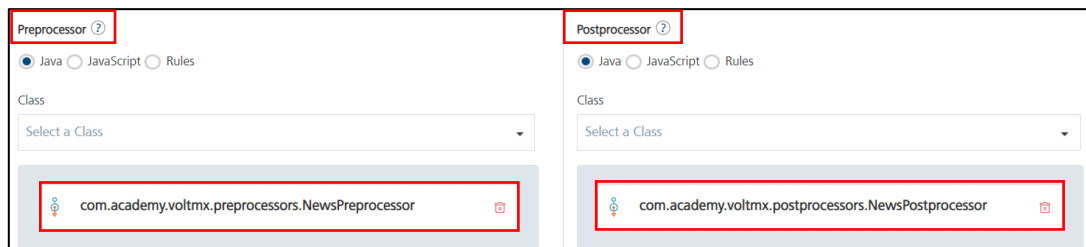
Request Template Show Hide

NAME	VALUE	TEST VALUE	DEFAULT VALUE...	DATA TYPE	ENCOD...	DESCRIPTION
newsType	request	World		string	✓	

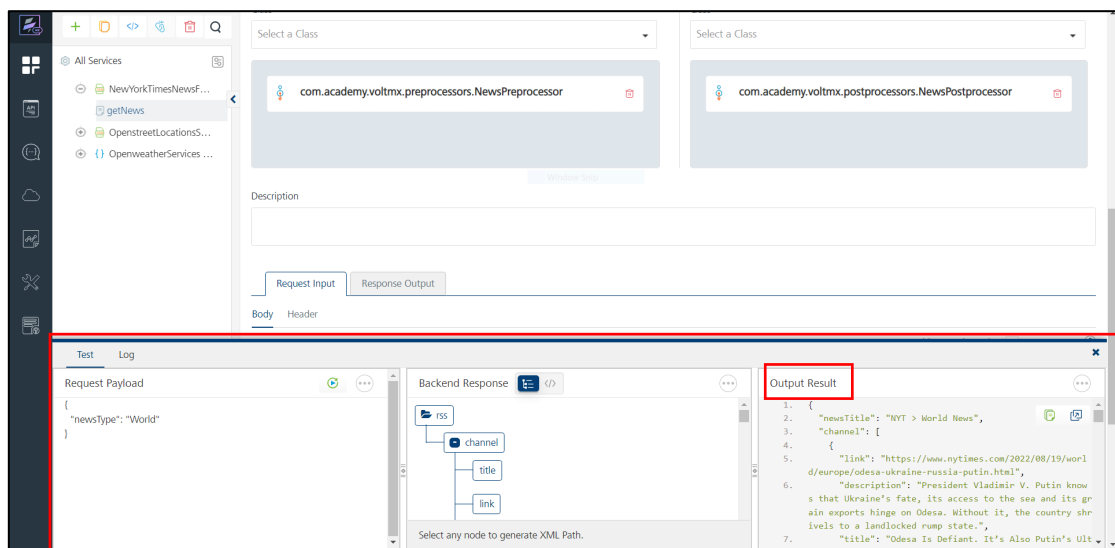
- **Advanced** オプションをクリックし、**Preprocessor** と **Postprocessor** セクションで **Java** を選択します。



- プリプロセスにクラス `com.academy.voltmx.preprocessors.NewsPreprocessor` を、ポストプロセッサにクラス `com.academy.voltmx.postprocessors.NewsPostprocessor` を選択します。



- **SAVE AND FETCH RESPONSE** をクリックします。 **Output Result** を確認します。



注意:

- トレイルクラウドを使用している場合、**Java** カスタムコードをテストすることはできません。この機能は、トレイルクラウド **Foundry** 環境では無効になっています。

おめでとうございます。このレッスンのハンズオンは完了です。