Creating Your First Program

In this section, will discuss the steps for creating a new "Hello World" ApiCode. We want to demonstrate how to create one ApiCode implementation from scratch. In this ApiCode 'class', we will be able to demonstrate the operations that we need to implement as a part of the ApiCode implementation.

The HelloWorld implementation class will take the input data as a list of HelloParameter objects. It will print it out by prefixing "Hello" to it. The Read operation on it will return an array of pre-defined HelloParameters with their data.

1. Open your TestAppCode project that you downloaded as a part of the QuickStart section.

2. Define your new HelloWorld AppCode implementation class in the source folder. Make sure that you are selecting the interface as AppCode while defining the class:

You should see a new class created with the default method signatures:

The following are basic functionalities of the three methods:

1. Open

This method should open the connection with the 3rd party system. It should initiate the session as mentioned earlier. In our case, it is 'Hello World'. And, we are not integrating with any third party system. As a result, we will just initialize the session key and print it out to the console and to the log.

```
@Override
public void open() throws RemoteException {
 // TODO Auto-generated method stub
 sessionKey = UUID.randomUUID().toString();
 logger.log(Level.INFO, "Hello World AppCode Session opened:" +
sessionKey);

System.out.println("Hello World AppCode Session opened:" + sessionKey);

 }
```

2. setContext

This method should initialize the context for the AppCode Implementation. It normally contains the initialization parameters that are required to initialize custom implementation.
In our 'HellWorld', we can just log a trace stating setContext is called in the console and in the log.

```
@Override
public void setContext(Context arg0) {
 // TODO Auto-generated method stub
 logger.log(Level.INFO, "Hello World AppCode Set Context called");

 System.out.println("Hello World AppCode Set Context called");

 }
```

3. Close

This method should close and terminate the session and invalidate it. In our case we can invalidate the session-key and log the call.

```
@Override
public void close() throws RemoteException {
 // TODO Auto-generated method stub
 // TODO Auto-generated method stub
 logger.log(Level.INFO, "Hello World AppCode close called:" + sessionKey);

 System.out.println("Hello World AppCode close called:" + sessionKey );

 sessionKey = null;

 }
```

Before we start the implementation of the "read" and "write" methods, we need to define the data transfer class ("HelloParameter") for carrying data out of the AppCode and the Filter class ("HelloFilter").

```
public static class HelloParameter{
 public String id;
 public String name;
 }

public static class HelloFilter{
 public String key;

 }
```

Define and initialize the initParameters to be used as HelloParameters for the sample read operation.

```
private List<HelloParameter> initParams ;

private void init() {

 initParams = new ArrayList<HelloWorldAppCode.HelloParameter>();

 HelloParameter initparam1 = new HelloParameter();
 initparam1.id = "1";
 initparam1.name = "Scott";

 HelloParameter initparam2 = new HelloParameter();
 initparam2.id = "2";
 initparam2.name = "Tiger";

 HelloParameter initparam3 = new HelloParameter();
 initparam3.id = "3";
 initparam3.name = "Matt";

 initParams.add(initparam1);
 initParams.add(initparam2);
 initParams.add(initparam3);

}
```

We need to ensure the init is called in the open method so that the parameters are initialized when the AppCode is instantiated and initialized.

```
@Override
public void open() throws RemoteException {
 // TODO Auto-generated method stub
 init();
 sessionKey = UUID.randomUUID().toString();
 logger.log(Level.INFO, "Hello World AppCode Session opened:" +
sessionKey);
 System.out.println("Hello World AppCode Session opened:" + sessionKey);

}
```

Now it is time to add the read and write operations.

d. ReadHello

In this read method, we match the filter passed with the initialized parameters in the array list. Then, we return back the parameter that match the key. Next, we set the value based on the sessionID initialized by prepending it with "Hello" and appending the name: "Hello:"<name>":"<sessionKey>.

```
public List<HelloParameter> readHello(HelloFilter filter){

 logger.info("Received Read operation with filter:" + filter.key + ":with
session" + sessionKey);

 System.out.println("Received Read operation with filter:" + filter.key +
":with session" + sessionKey);

 List<HelloParameter> returnList = new
ArrayList<HelloWorldSvc.HelloParameter>();

 for(HelloParameter initParam : initParams) {

  if(filter.key.equalsIgnoreCase(initParam.name)) {

   //Now set the value to "Hello":name:sessionId
   initParam.value = "Hello:"+initParam.name+":"+sessionKey;

   System.out.println("Matched Key:" + initParam.name + ":value:" +
initParam.value );

   logger.info("Matched Key:" + initParam.name + ":value:" +
initParam.value );

   returnList.add(initParam);
  }
 }

 return returnList;
}
```

e. WriteHello

In the write method we take the input list and print the value based on the logic explained above. After iterating through the list, we return back an array of SaveResults with the newID that contains the appended sessionKey.
You can view the code below

```
public SaveResult[] writeHello(List<HelloParameter> inputParams){

 List<SaveResult> saveResults = new ArrayList<SaveResult>();

 logger.info("Received write operation with session" + sessionKey);

 System.out.println("Received write operation with session" + sessionKey);

 for(HelloParameter inputParam : inputParams) {

  System.out.println("Received input parameter:id:" + inputParam.id +
":name:" + inputParam.name);
  logger.info("Received input parameter:id:" + inputParam.id + ":name:" +
inputParam.name);

  inputParam.value = "Hello:"+inputParam.name+":"+sessionKey;
  System.out.println("Writing value:" + inputParam.value );
  logger.info("Writing value:" + inputParam.value);

  SaveResult result = new SaveResult();
  result.id = inputParam.id;
  result.newid = inputParam.id+":"+sessionKey;
  result.success = true;

  saveResults.add(result);

 }

 return saveResults.toArray(new SaveResult[] {});
}
```

Now, we can view the entire class below:

```
package myappcode;

import java.rmi.RemoteException;
import java.util.ArrayList;
import java.util.UUID;
import java.util.logging.Level;
import java.util.logging.Logger;

import com.appmashups.appcode.AppCode;
import com.appmashups.appcode.Context;
import com.appmashups.appcode.SaveResult;

import java.util.List;

public class HelloWorldSvc implements AppCode {

 public static class HelloParameter{
  public String id;
```

```
  public String name;
  public String value;
 }

 public static class HelloFilter{
  public String key;
}

 private List<HelloParameter> initParams ;

 private String sessionKey;

 static Logger logger = Logger.getLogger(HelloWorldSvc.class.getName());

 @Override
 public void close() throws RemoteException {
  // TODO Auto-generated method stub
  // TODO Auto-generated method stub
  logger.log(Level.INFO, "Hello World AppCode close called:" + sessionKey);
  System.out.println("Hello World AppCode close called:" + sessionKey );
  sessionKey = null;

 }

 @Override
 public void open() throws RemoteException {
  // TODO Auto-generated method stub
  init();
  sessionKey = UUID.randomUUID().toString();
  logger.log(Level.INFO, "Hello World AppCode Session opened:" +
sessionKey);
  System.out.println("Hello World AppCode Session opened:" + sessionKey);
 }

 @Override
 public void setContext(Context arg0) {
  // TODO Auto-generated method stub
  logger.log(Level.INFO, "Hello World AppCode Set Context called");
  System.out.println("Hello World AppCode Set Context called");

 }

 private void init() {

  initParams = new ArrayList<HelloWorldSvc.HelloParameter>();

  HelloParameter initparam1 = new HelloParameter();
  initparam1.id = "1";
  initparam1.name = "Scott";

  HelloParameter initparam2 = new HelloParameter();
  initparam2.id = "2";
  initparam2.name = "Tiger";
```

```
  HelloParameter initparam3 = new HelloParameter();
  initparam3.id = "3";
  initparam3.name = "Matt";

  initParams.add(initparam1);
  initParams.add(initparam2);
  initParams.add(initparam3);

 }

 public List<HelloParameter> readHello(HelloFilter filter){

  logger.info("Received Read operation with filter:" + filter.key + ":with
session" + sessionKey);

  System.out.println("Received Read operation with filter:" + filter.key +
":with session" + sessionKey);

  List<HelloParameter> returnList = new
 ArrayList<HelloWorldSvc.HelloParameter>();

  for(HelloParameter initParam : initParams) {

   if(filter.key.equalsIgnoreCase(initParam.name)) {

    //Now set the value to "Hello":name:sessionId
    initParam.value = "Hello:"+initParam.name+":"+sessionKey;

    System.out.println("Matched Key:" + initParam.name + ":value:" +
initParam.value );

    logger.info("Matched Key:" + initParam.name + ":value:" +
initParam.value );

    returnList.add(initParam);
   }
  }

  return returnList;

 }

 public SaveResult[] writeHello(List<HelloParameter> inputParams){

  List<SaveResult> saveResults = new ArrayList<SaveResult>();

  logger.info("Received write operation with session" + sessionKey);

  System.out.println("Received write operation with session" + sessionKey);

  for(HelloParameter inputParam : inputParams) {
```

```
   System.out.println("Received input parameter:id:" + inputParam.id +
":name:" + inputParam.name);
   logger.info("Received input parameter:id:" + inputParam.id + ":name:" +
inputParam.name);

   inputParam.value = "Hello:"+inputParam.name+":"+sessionKey;

   System.out.println("Writing value:" + inputParam.value );

   logger.info("Writing value:" + inputParam.value);

   SaveResult result = new SaveResult();
   result.id = inputParam.id;
   result.newid = inputParam.id+":"+sessionKey;
   result.success = true;

   saveResults.add(result);

   }

   return saveResults.toArray(new SaveResult[] {});
  }
```

```
}
```

```
@Test
public void testReadHello() {

 try {

  HelloWorldSvc svc = new HelloWorldSvc();

  svc.setContext(null);

  svc.open();

  HelloFilter filter = new HelloFilter();
  filter.key = "Scott";

  svc.readHello(filter);

  svc.close();

  } catch (Exception e) {
   e.printStackTrace();

  }

 }
```

Next, we will test the write operation. Here, we take Array of HelloParameter objects as input and the write operation prints it out.

```
@Test
public void testWriteHello() {
try
```

Unknown macro: { HelloWorldSvc svc = new HelloWorldSvc(); svc.setContext(null); svc.open();

HelloParameter param = new HelloParameter(); param.name = "Test1"; param.id = "id1"; HelloParameter param2 = new HelloParameter(); param2.name = "Test1"; param2.id = "id2"; List<HelloParameter> inputParams = new ArrayList<HelloWorldSvc.HelloParameter>(); svc.writeHello(inputParams); svc.close(); }

```
catch (Exception e)
```

Unknown macro: { e.printStackTrace(); }

```
}
```

So, if we analyze the outputs of the read and write test runs, we find that the Read operation prints out the result from the init array that matches the filter by appending the session key.

The Write operation reads the input parameter array. Then, it prints the input parameter by prepending "Hello" to the name and appending the SessionKey to the same.

Now that we have developed our HelloWorldSvc AppCode implementation, it is time to deploy it to DBSync platform. Then, we will run a test workfow.

For your reference, the complete code is given below:

```
package myappcode.test;

import static org.junit.Assert.*;

import java.util.ArrayList;
import java.util.List;

import myappcode.HelloWorldSvc;
import myappcode.HelloWorldSvc.HelloFilter;
import myappcode.HelloWorldSvc.HelloParameter;

import org.apache.tools.ant.taskdefs.Concat;
import org.junit.Test;

import com.appmashups.appcode.Context;

public class TestHelloWorldAppCode {

 //@Test
 public void testOpen() {

  try {

   HelloWorldSvc svc = new HelloWorldSvc();

   svc.setContext(null);

   svc.open();

   svc.close();

  } catch (Exception e) {
   e.printStackTrace();
  }
 }

 //@Test
 public void testReadHello() {

  try {

   HelloWorldSvc svc = new HelloWorldSvc();

   svc.setContext(null);

   svc.open();

   HelloFilter filter = new HelloFilter();
   filter.key = "Scott";

   svc.readHello(filter);

   svc.close();
```

```
 } catch (Exception e) {
  e.printStackTrace();
 }

}

 @Test
 public void testWriteHello() {
  try {

   HelloWorldSvc svc = new HelloWorldSvc();

   svc.setContext(null);

   svc.open();

   HelloParameter param = new HelloParameter();
   param.name = "Test1";
   param.id = "id1";

   HelloParameter param2 = new HelloParameter();
   param2.name = "Test1";
   param2.id = "id2";

   List<HelloParameter> inputParams = new
 ArrayList<HelloWorldSvc.HelloParameter>();
   inputParams.add(param);
   inputParams.add(param2);

   svc.writeHello(inputParams);

   svc.close();

  } catch (Exception e) {
   e.printStackTrace();
 }
```

```
}
```