Contents

[Introduction 2](#_Toc99097598)

[Microservices conversion steps 3](#_Toc99097599)

[Step 1: In the BBean module 3](#_Toc99097600)

[Create new constant variables 3](#_Toc99097601)

[Refactor PropertiesManager 3](#_Toc99097602)

[Refactor Log4J to SLF4j 3](#_Toc99097603)

[Create and setup web services endpoints 4](#_Toc99097604)

[Step 2: In BD and CC\_WS module 4](#_Toc99097605)

[Setup Spring Boot 4](#_Toc99097606)

[Refactor Log4J to SLF4j 5](#_Toc99097607)

[Refactor calling Spring ‘getBean’ via ‘applicationContext’ from Constant class 5](#_Toc99097608)

[Create and setup BD web services (only in BD module) 5](#_Toc99097609)

[Refactor DAO classes that having auto-wired SessionFactory to EntityManagerFactory (only in BD module) 6](#_Toc99097610)

[Refactor quartz job classes (only in BD module) 7](#_Toc99097611)

[Step 3: Go back to the BBean module 8](#_Toc99097612)

[Continue setup web services endpoints 8](#_Toc99097613)

[Step 4: Go back BD and CC\_WS module 9](#_Toc99097614)

[Setup and calling MuleDispatcherV2 9](#_Toc99097615)

[Step 5: In CC module 9](#_Toc99097616)

[Setup and calling MuleDispatcherV2 9](#_Toc99097617)

# Introduction

* Requirements:
	+ Minimum JDK version: JDK 11
	+ IDE: eclipse
	+ Minimum Wildfly version: 20
* MLEB microservices SVN path:

http://smes.silverlakegroup.com.my:8088/svn/mleb/Microservices/

* Convert to MLEB microservices from this SVN path:

http://smes.silverlakegroup.com.my:8088/svn/mleb/GeRewardv2MySql/

* Microservices conversion only applicable to BD and CC\_WS module

For CC (solution domain/backoffice) this module will continue build into WAR file and deploy into wildfly

* To build microservices, the cli command is:

# set JAVA\_HOME=[your installed JDK 11]

# cd mleb\_\*

# gradlew :bd\_\*:clean :bd\_\*:bootjar

For example:

# set JAVA\_HOME=C:\Program Files\AdoptOpenJDK\jdk-11.0.10.9-hotspot

# cd mleb\_vrsm

# gradlew :bd\_vrsm:clean :bd\_vrsm:bootjar

* To start the microservices, the cli command is

# set JAVA\_HOME=[your installed JDK 11]

# cd mleb\_\*

# java -jar build\libs\\*.jar

For example:

# set JAVA\_HOME=C:\Program Files\AdoptOpenJDK\jdk-11.0.10.9-hotspot

# cd mleb\_vrsm

# java -jar build\libs\bd\_vrsm-5.0.jar

# Microservices conversion steps

## Step 1: In the BBean module

### Create new constant variables

* In the module’s main Constant class, e.g. if in mleb\_vrsm then the Constant class name will be ‘VRSMConstant.java’. Add these 3 lines:

 public static boolean LOAD\_PROPERTIES\_VIA\_FILE = true;

 public static ApplicationContext applicationContext;

 public static Environment environment;

### Refactor PropertiesManager

* In the ‘PropertiesManager.java’, need add constructor for getting properties from Spring Boot environment variables, please refer to PropertiesManager.java in the ‘bbean\_iam’
* Global search the entire BBean module that are instantiating PropertiesManager class. And then refactor it by checking the \*Constant.LOAD\_PROPERTIES\_VIA\_FILE, if it is true then instantiate it with empty constructor else instantiate it with \*Constant.environment constructor

### Refactor Log4J to SLF4j

* Global search the entire BBean module for text “import org.apache.log4j.Logger;” and replace it with “import org.slf4j.\*;”

### Create and setup web services endpoints

* Go to ‘mleb\_am\bbean\_am\src\com\silverlake\ifb\remote’ folder and then copy these 6 files:

AbstractRestTemplate.java

RestWebServiceDispatcher.java

CommonFunctionCodeEnum.java

IAMFunctionCodeEnum.java

MAMFunctionCodeEnum.java

MlebCoreFunctionCodeEnum.java

And then paste it into package ‘com.silverlake.ifb.remote’

* Global search the entire BBean module for ‘HttpInvokerDispatcher’ and replace it with ‘RestWebServiceDispatcher’

## Step 2: In BD and CC\_WS module

### Setup Spring Boot

* Go to ‘mleb\_iam\bd\_iam\src\com\silverlake\ifb’ and the copy ‘BdIamApplication.java’ and ‘ServletInitializer.java’ and paste into package ‘com.silverlake.ifb’
* Remember to rename the ‘BdIamApplication’ to the proper class name e.g. if in mleb\_vrsm then change it to ‘BdVrsmApplication.java’. After renaming this class, you need to fix ‘ServletInitializer.java’
* Copy the config package from ‘mleb\_am\bd\_am\src\main\java\com\silverlake\ifb’ and paste into ‘mleb\_\*\bd\_\*\src\main\java\com\silverlake\ifb’
* If the current BD don’t have quartz scheduler, the do this:
	+ Open class ‘BdIamApplication’ and remove the line with ‘@EnableScheduling’
	+ Delete the class ‘SpringQrtzScheduler’
	+ Delete the package ‘com.silverlake.ifb.config.quartz’

### Refactor Log4J to SLF4j

* Global search the entire BD/CC\_WS module for text “import org.apache.log4j.Logger;” and replace it with “import org.slf4j.\*;”
* Global search the entire BD/CC\_WS module that are instantiating PropertiesManager class. And then refactor it by instantiate it with \*Constant.environment constructor

### Refactor calling Spring ‘getBean’ via ‘applicationContext’ from Constant class

* Search the entire BD/CC\_WS module for keyword ‘.getBean(‘ and then each search result change the calling method to ‘\*Constant.applicationContext’, e.g. if in mleb\_vrsm, the changed result should be like ‘VRSMConstant.application.getBean(’

### Create and setup BD web services (only in BD module)

* Go to ‘mleb\_am\bd\_am\src\com\silverlake\ifb\module’ folder and then copy this file:

AbstractBdModule.java

Into package ‘com.silverlake.ifb.module’

* In the ‘com.silverlake.ifb.module’, all the classes need to add ‘extends AbstractBdModule’. (If the class is currently ‘extends AppContext’ then just overwrite it)
* Create new package com.silverlake.ifb.web.rest and then create new class called ‘BdSolutionController.java’
* Go to ‘mleb\_am\bd\_am\src\com\silverlake\ifb\web\rest’ folder and then copy this file:

AbstractRestController.java

Into package ‘com.silverlake.ifb.web.rest’

* In the ‘BdSolutionController’ class add ‘extends AbstractRestController’. And then add the request mapping path, e.g. if in mleb\_vrsm then the request mapping path will be ‘/api/v1/bdvrsmsolution’. And then add post mapping methods for all the module classes. For example you can refer to

‘mleb\_am\bd\_am\src\main\java\com\silverlake\ifb\web\rest\BdAMSolution.java

### Refactor DAO classes that having auto-wired SessionFactory to EntityManagerFactory (only in BD module)

* In the package ‘com.silverlake.ifb.dao’, search for method signature:

public void setSessionFactory(SessionFactory sessionFactory) {

 hibernateTemplate = new HibernateTemplate(sessionFactory);

}

And then change it to this:

public void setSessionFactory(EntityManagerFactory emf) {

 SessionFactory sessionFactory = emf.unwrap(SessionFactory.class);

 hibernateTemplate = new HibernateTemplate(sessionFactory);

 }

* In the package ‘com.silverlake.ifb.dao’, search for in the mehod ‘setSessionFactory’ for ‘dialect’ variable, and if found then change that method to this:

public void setSessionFactory(EntityManagerFactory emf) {

 SessionFactory sessionFactory = emf.unwrap(SessionFactory.class);

 hibernateTemplate = new HibernateTemplate(sessionFactory);

 String s\_dialect = (String) emf.getProperties().getOrDefault("hibernate.dialect", "");

 try {

 dialect = (Dialect) Class.forName(s\_dialect).getConstructor().newInstance();

 } catch (Exception e) {

 e.printStackTrace();

 }

 }

* In the package ‘com.silverlake.ifb.dao’, search for ‘@Resource(name = "sessionFactory")’, and if found:
	+ Remove the ‘@Resource(name = "sessionFactory")’
	+ Add this method:

@Autowired

public void setSessionFactory(EntityManagerFactory emf) {

 this.sessionFactory = emf.unwrap(SessionFactory.class);

 }

### Refactor quartz job classes (only in BD module)

* All quartz job do this:
	+ Remove the ‘implements Job’ and replace with ‘extends QuartzJobBean’. Fix the error by change the override method name to ‘executeInternal’
	+ In the ‘executeInternal’, add this line ‘SpringBeanAutowiringSupport.processInjectionBasedOnCurrentContext(this);’ at the top of that method body
	+ All the line that calling Spring managed objects via ‘getBeans’ method should be remarked or removed. And replace with auto-wiring annotation. For example,

*Remarked this line: //CrossChannelService crossChannelService = (CrossChannelService) ctx.getBean("crossChannelService");*

*Add this line at the class body:*

*@Autowired*

*private CrossChannelService crossChannelService;*

* In the quartz.properties, remark all the property key starts with ‘org.quartz.plugin.jobInitializer’, for example

*#org.quartz.plugin.jobInitializer.class = org.quartz.plugins.xml.XMLSchedulingDataProcessorPlugin*

*#org.quartz.plugin.jobInitializer.fileNames = quartz.xml*

*#org.quartz.plugin.jobInitializer.failOnFileNotFound = true*

*#org.quartz.plugin.jobInitializer.scanInterval = 120*

## Step 3: Go back to the BBean module

### Continue setup web services endpoints

* Go to ‘mleb\_am\bbean\_am\src\com\silverlake\ifb\remote’ folder and then copy this file:

MicroserviceUrlEnum.java

* Open the ‘MicroserviceUrlEnum’ class, and then remove all the BD AM enumerated endpoints. And replace with current BD module enumerated endpoints. For the format and logics how to create enumerated endpoints, please refer to ‘MicroserviceUrlEnum.java’ in ‘mleb\_am/bbean/src/main/ java/com/silverlake/ifb/remote’
* After finish setting up the enumerated BD module endpoints, now setup mapping of the Function Code to BD module enumerated endpoints.

First, in the ‘com.silverlake.ifb.remote’ package, duplicate the ‘IAMFunctionCodeEnum’ class and then rename the duplicated class to ‘\*FunctionCodeEnum’, e.g. if current BD is ‘bd\_vrsm’ then that class name would be ‘VrsmFunctionCodeEnum’.

Secondly, remove all the enumerated function codes and replace the current BD’s function codes as new enumerated value. Make sure the enumerated parameter is mapped to the correct endpoint in the ‘MicroserviceUrlEnum’

Lastly, in the ‘\*FunctionCodeEnum’ class, rename the method name ‘getBdAMFullEndpoint’ to ‘getBd\*FullEndpoint’, e.g. if current BD is ‘bd\_vrsm’ then that class name would be ‘getBdVrsmFullEndpoint’.

## Step 4: Go back BD and CC\_WS module

### Setup and calling MuleDispatcherV2

* Copy ‘MuleDispatcherV2.java’ from ‘mleb\_am\bd\_am\src\main\java\com\silverlake\ifb\dispatcher\’ and paste into package ‘com.silverlake.ifb.dispacher’
* After that open the ‘MuleDispatcherV2.java’ find the ‘AMFunctionCodeEnum’ and change it to current BD’s function code enumerated class, e.g. if current BD is ‘bd\_vrsm’ then the current BD’s function code enumerated class is ‘VrsmFunctionCoeEnum’
* Global search the current BD/CC\_WS for ‘MuleDispatcher’ and if found replace it with ‘MuleDispatcherV2’
* Global search the current BD/CC\_WS for ‘sendToRemoteDispatcher’, and do the following logic:
	+ If the function code is from current ‘\*FunctionCodeEnum’, then ‘sendToRemoteDispatcher’ calling method second parameter should ‘null’
	+ If the function code is not in current ‘\*FunctionCodeEnum’, then ‘sendToRemoteDispatcher’ calling method second parameter should be from another Function Code enum class. If that function code not found then need to add it.

## Step 5: In CC module

### Setup and calling MuleDispatcherV2

* Copy ‘MuleDispatcherV2.java’ from ‘mleb\_am\cc\_am\src\main\java\com\silverlake\ifb\dispatcher\’ and paste into package ‘com.silverlake.ifb.dispacher’
* After that open the ‘MuleDispatcherV2.java’ find the ‘AMFunctionCodeEnum’ and change it to current BD’s function code enumerated class, e.g. if current BD is ‘bd\_vrsm’ then the current BD’s function code enumerated class is ‘VrsmFunctionCoeEnum’
* Global search the current BD/CC\_WS for ‘MuleDispatcher’ and if found replace it with ‘MuleDispatcherV2’
* Global search the current BD/CC\_WS for ‘sendToRemoteDispatcher’, and do the following logic:
	+ If the function code is from current ‘\*FunctionCodeEnum’, then ‘sendToRemoteDispatcher’ calling method second parameter should ‘null’

If the function code is not in current ‘\*FunctionCodeEnum’, then ‘sendToRemoteDispatcher’ calling method second parameter should be from another Function Code enum class