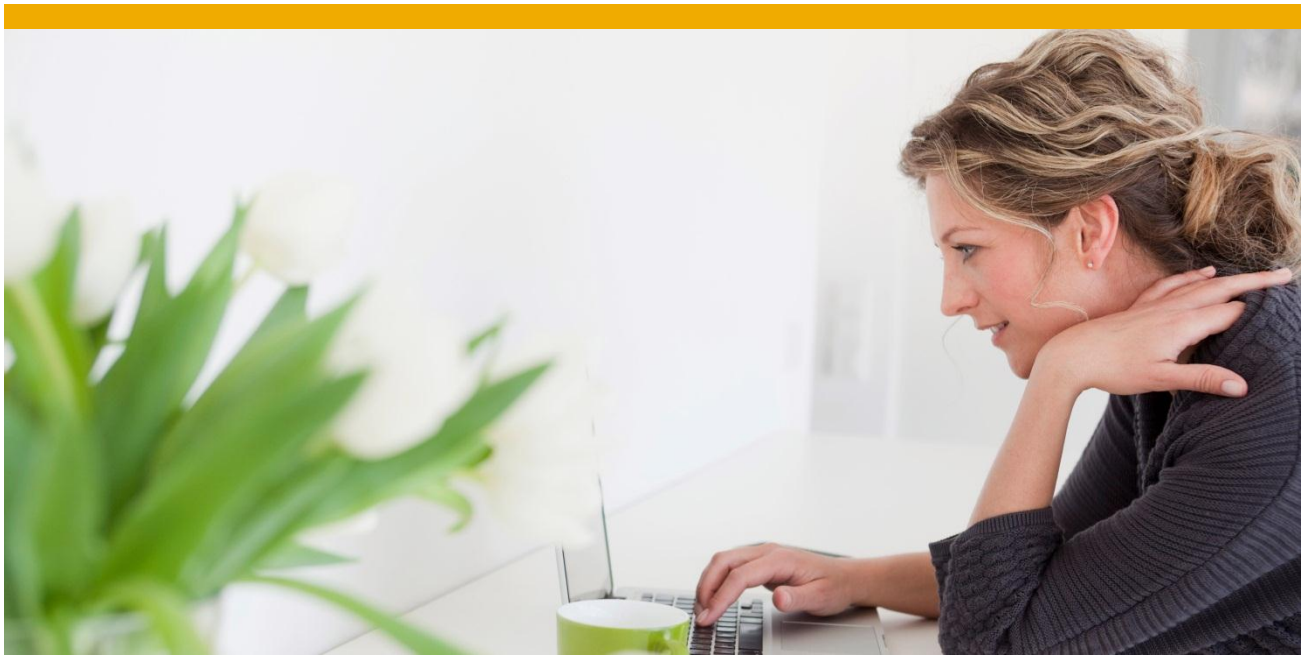


LAB2b: Create Mobiliser Web Services

Implement CRUD Services

TABLE OF CONTENTS

AGENDA

1. Extend the existing service Contract
2. Extend the service API
3. Implement the business logic and converter
4. Implement the endpoint methods
5. Test the service methods

BEFORE YOU START…

Please start the Mobiliser 5.1 Lab Virtual machine.

The Login with user is "mobiliser" and password "sybase".

CREATE THE CONTRACT AND EXTEND THE API

The customized table will be accessed by this new CRUD services. In order to keep it simple, we only implement the methods create and get/retrieve. The update method can be implemented optionally. For simplicity we extend the existing custom service.

1. Start eclipse
2. Navigate to project com.sybase365.mobiliser.custom.project.services.contract/src/main/resources/…/xsd/ and open the custom **beans-1-0.xsd** file
3. Create the Department Type that contains the id and name elements. Reuse the base types (see the documentation as a reference)

   (for more information please see Mobiliser_Framework_5.1_Development_Guide.pdf section **5.3.2 Beans XSD**)

NOTE: Please pay close attention during copying and pasting the code from the PDF into eclipse workspace because sometimes Word → PDF conversion messes up the “DOUBLE QOUTE” and sometimes adds additional CR/LF. This is more common when copying XML files.

"(…)" is not a valid XML content – this symbol is used for brevity in presenting the XML file content. Having "(…)" in your actual xml configuration file would cause XML parsing errors.

```
(...)

<xs:complexType name="Department">
   <xs:annotation>
      <xs:documentation>The Department XSD complex type</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="name" type="base:strMediumNonEmpty" maxOccurs="1" minOccurs="1">
         </xs:element>
         <xs:element name="id" type="base:idLong" maxOccurs="1" minOccurs="0">
         </xs:element>
      </xs:sequence>
</xs:complexType>

(...)
```

4. Copying/Pasting above xml snippet into beans-1-0.xsd would drop the indentation – please press “Ctrl+A” to select the contents and then press “Ctrl+I” for indentation of XML contents.
5. Save the file **beans-1-0.xsd.**

6. Now open the custom **requests-1-0.xsd** file and add the request and response types and request and response elements for the createDepartment and getDepartment service. Please note that getDepartment request is a selector and createDepartment a modifier. Therefore choose the correct Mobiliser request type to extend. Create services that generate a new id should return this id.

(for more information please see Mobiliser_Framework_5.1_Development_Guide.pdf section **5.3.3 Requests XSD**)

requests-1-0.xsd (Part 1 – Get Department Service)

```
(...)

<xs:complexType name="GetDepartmentRequestType">
   <xs:complexContent>
      <xs:extension base="base:MobiliserRequestType">
         <xs:sequence>
            <xs:element name="id" type="base:idLong"
               minOccurs="1" maxOccurs="1" />
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="GetDepartmentResponseType">
   <xs:complexContent>
      <xs:extension base="base:MobiliserResponseType">
         <xs:sequence>
            <xs:element name="department" type="beans:Department"
               minOccurs="0" maxOccurs="1" />
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:element name="GetDepartmentRequest">
   <xs:complexType>
      <xs:complexContent>
         <xs:extension base="GetDepartmentRequestType">
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
<xs:element name="GetDepartmentResponse">
   <xs:complexType>
      <xs:complexContent>
         <xs:extension base="GetDepartmentResponseType">
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:element>

(...)
```

requests-1-0.xsd (Part 2 – Create Department Service)

```
(...)

<xs:complexType name="CreateDepartmentRequestType">
   <xs:complexContent>
      <xs:extension base="base:TraceableRequestType">
         <xs:sequence>
            <xs:element name="department" type="beans:Department"
               minOccurs="1" maxOccurs="1" />
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:complexType name="CreateDepartmentResponseType">
   <xs:complexContent>
      <xs:extension base="base:MobiliserResponseType">
         <xs:sequence>
            <xs:element name="id" type="base:idLong"
               minOccurs="0" maxOccurs="1" />
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:element name="CreateDepartmentRequest">
   <xs:complexType>
      <xs:complexContent>
         <xs:extension base="CreateDepartmentRequestType">
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
<xs:element name="CreateDepartmentResponse">
   <xs:complexType>
      <xs:complexContent>
         <xs:extension base="CreateDepartmentResponseType">
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:element>

(...)
```

7. Copying/Pasting above xml snippet into requests-1-0.xsd would drop the indentation – please press “Ctrl+A” to select the contents and then press “Ctrl+I” for indentation of XML contents.
8. Save the file **requests-1-0.xsd.**
9. Select the bundle project and click on *Run As* → *Maven generate-sources* (right click to pop down the context menu or execute from the command line)
10. [**Optional** – if it is not part of the build path] After the maven build was successful refresh the target/xjc source folder. To make it available to the classpath (if not already), simply right click on the xjc folder and select *Build Path* → *Use as Source Folder*

IMPLEMENT THE BUSINESS LOGIC AND CONVERTER

The business logic is build out of two parts – the OSGi service interface that is exported by the business logic bundle and the implementation. The business logic does not use the contract beans, therefore a converter service is used to convert database beans to contract beans vice versa.

1. Navigate to com.sybase365.mobiliser.custom.project.businesslogic.api and create the **IDepartmentLogic.java** interface in the business logic api bundle. The new interface should be added to the following package:

`com.sybase365.mobiliser.custom.project.businesslogic`

2. Add the two new service methods to the interface.
    a. The creator method expects a department model bean and returns the id.
    b. The selector expects the id and returns the model bean

```
package com.sybase365.mobiliser.custom.project.businesslogic;

import com.sybase365.mobiliser.custom.project.persistence.model.Department;
import com.sybase365.mobiliser.money.businesslogic.util.EntityNotFoundException;

public interface IDepartmentLogic {

    long createDepartment(final Department department, final long callerId);

    Department getDepartment(final long id, final long callerId)
            throws EntityNotFoundException;
}
```

3. Please apply java code formatter for greater code readability – press “Ctrl+A” to select all the code in Department.java and then press “Ctrl+Shift+F” for formatting the contents of the file.
4. To organize all the imports, please press “Ctrl-Shift-O” – please make sure that imports are correct.
5. Save the file IDepartmentLogic.java.

(for more information please see Mobiliser_Framework_5.1_Development_Guide.pdf section **5.6.2 Service Business Logic**)

6. Now implement the department interface in the com.sybase365.mobiliser.custom.project.businesslogic.impl bundle project as **DepartmentLogicImpl.java**. Let it implement the `InitializingBean` interface and inject the custom DAO factory. The `afterPropertiesSet()` must verify that the injected DAO factory is not null.

7. The implementation should be added to the following package:
`com.sybase365.mobiliser.custom.project.businesslogic.impl`

```
package com.sybase365.mobiliser.custom.project.businesslogic.impl;

import org.springframework.beans.factory.InitializationBean;
import com.sybase365.mobiliser.custom.project.businesslogic.IDepartmentLogic;
import com.sybase365.mobiliser.custom.project.persistence.dao.factory.api.DaoFactory;
import com.sybase365.mobiliser.custom.project.persistence.model.Department;
import com.sybase365.mobiliser.money.businesslogic.util.EntityNotFoundException;

public class DepartmentLogicImpl implements IDepartmentLogic, InitializingBean {
    private DaoFactory daoFactory;

    @Override
    public void afterPropertiesSet() throws Exception {
       if (this.daoFactory == null) {
           throw new IllegalStateException("daoFactory is required");
       }
    }

    @Override
    public long createDepartment(Department department, long callerId) {
       getDaoFactory().getDepartmentDao().save(department,
              Long.valueOf(callerId));
       return department.getId().longValue();
    }

    @Override
    public Department getDepartment(long id, long callerId)
           throws EntityNotFoundException {
       Department department = getDaoFactory().getDepartmentDao().getById(Long.valueOf(id));
       if (department == null) {
           throw new EntityNotFoundException();
       }
       return department;
    }
    public DaoFactory getDaoFactory() {
       return this.daoFactory;
    }
    public void setDaoFactory(DaoFactory daoFactory) {
       this.daoFactory = daoFactory;
    }
}
```

Note: Please see we have imported `com.sybase365.mobiliser.custom.project.persistence.dao.factory.api.DaoFactory` because the new table would only be available through the custom DaoFactory not from the Core (Money) DaoFactory.

(for more information please see Mobiliser_Framework_5.1_Development_Guide.pdf section **5.6.2 Service Business Logic**)

8. Please apply java code formatter for greater code readability – press “Ctrl+A” to select all the code in Department.java and then press “Ctrl+Shift+F” for formatting the contents of the file.
9. To organize all the imports, please press “Ctrl+Shift+O” – please make sure that imports are correct.
10. Save the file `DepartmentLogicImpl.java`
11. Now let’s create bean Converter – this utility class will play an important role in translating Contract bean into Model bean. Endpoint would use this class to translate the Contract bean into Model and pass information to BusinessLogic service.

12. Navigate to com.sybase365.mobiliser.custom.project.businesslogic.api bundle project and create a converter interface named **IDepartmentConverter.java** (converter package) with the methods `fromContract();` and `toContract();`
13. **IDepartmentConverter.java** should be created in following package:
`com.sybase365.mobiliser.custom.project.converter`

```
package com.sybase365.mobiliser.custom.project.converter;

import com.sybase365.mobiliser.custom.project.persistence.model.Department;

public interface IDepartmentConverter {
    Department
fromContract(com.sybase365.mobiliser.custom.project.services.contract.v1_0.beans.Department
contract);

    com.sybase365.mobiliser.custom.project.services.contract.v1_0.beans.Department
toContract(Department db);

}
```

14. Please apply java code formatter for greater code readability – press “Ctrl+A” to select all the code in Department.java and then press “Ctrl+Shift+F” for formatting the contents of the file.
15. To organize all the imports, please press “Ctrl+Shift+O” – please make sure that imports are correct.
16. Save the file IDepartmentConverter.java

17. Now implement the department converter interface in the com.sybase365.mobiliser.custom.project.businesslogic.impl bundle as **DepartmentConverter.java**.
18. **DepartmentConverter.java** should be created in following package:
`com.sybase365.mobiliser.custom.project.converter.impl`

(for more information please see Mobiliser_Framework_5.1_Development_Guide.pdf section **5.6.1 Converter**)

```
package com.sybase365.mobiliser.custom.project.converter.impl;

import org.springframework.beans.factory.InitializationBean;
import com.sybase365.mobiliser.custom.project.converter.IDepartmentConverter;
import com.sybase365.mobiliser.custom.project.persistence.dao.factory.api.DaoFactory;
import com.sybase365.mobiliser.custom.project.persistence.model.Department;

public class DepartmentConverter implements IDepartmentConverter,
       InitializingBean {
    private DaoFactory daoFactory;

    @Override
    public void afterPropertiesSet() throws Exception {
       if (this.daoFactory == null) {
           throw new IllegalStateException("daoFactory is required");
       }
    }

    @Override
    public Department
fromContract(com.sybase365.mobiliser.custom.project.services.contract.v1_0.beans.Department
contract) {

       Department db = getDaoFactory().getDepartmentDao()
              .newInstance();
       db.setName(contract.getName());
       return db;
    }

    @Override
    public com.sybase365.mobiliser.custom.project.services.contract.v1_0.beans.Department
toContract(Department db) {

       com.sybase365.mobiliser.custom.project.services.contract.v1_0.beans.Department
contract = new
com.sybase365.mobiliser.custom.project.services.contract.v1_0.beans.Department();
       contract.setId(db.getId());
       contract.setName(db.getName());
       return contract;
    }

    public DaoFactory getDaoFactory() {
       return this.daoFactory;
    }

    public void setDaoFactory(DaoFactory daoFactory) {
       this.daoFactory = daoFactory;
    }
}
```

19. Please apply java code formatter for greater code readability – press “Ctrl+A” to select all the code in Department.java and then press “Ctrl+Shift+F” for formatting the contents of the file.
20. To organize all the imports, please press “Ctrl+Shift+O” – please make sure that imports are correct.
21. Save the file DepartmentConverter.java

22. Now let's move on to Spring Bean and OSGI configuration.
23. Navigate to src/main/resources package under com.sybase365.mobiliser.custom.project.businesslogic.impl bundle project in eclipse.
24. Open the **bundle-context.xml** configuration and add the beans for the implementing classes (converter and business logic). The business logic must also inject the DAO factory.

```
(...)

<bean id="departmentLogic"
class="com.sybase365.mobiliser.custom.project.businesslogic.impl.DepartmentLogicImpl" >
    <property name="daoFactory" ref="daoFactory" />
</bean>

<bean id="departmentConverter"
class="com.sybase365.mobiliser.custom.project.converter.impl.DepartmentConverter" >
    <property name="daoFactory" ref="daoFactory" />
</bean>

(...)
```

25. Copying/Pasting above xml would drop the indentation – please press "Ctrl+A" to select the contents and then press "Ctrl+I" for indentation of XML contents.
26. Save the file **bundle-context.xml.**

27. Open the OSGi **bundle-context-osgi.xml** configuration and add the converter and business logic interface to the service exports

```
(...)

<osgi:service
interface="com.sybase365.mobiliser.custom.project.businesslogic.IDepartmentLogic"
ref="departmentLogic" />

<osgi:service
interface="com.sybase365.mobiliser.custom.project.converter.IDepartmentConverter"
ref="departmentConverter" />

(...)
```

28. Copying/Pasting above xml would drop the indentation – please press "Ctrl+A" to select the contents
29. and then press "Ctrl+I" for indentation of XML contents.
30. Save the file **bundle-context-osgi.xml.**

CREATE THE ENDPOINT IMPLEMENTATION

Now the endpoint implementation can use the business logic and converter. Both must be injected via spring configuration and the service is referenced in the OSGi configuration.

1. Create an endpoint interface named **IDepartmentEndpoint.java** to expose the services to the client under the *com.sybase365.mobiliser.custom.project.service.contract* bundle
2. IDepartmentEndpoint.java should be created in the following package:
`com.sybase365.mobiliser.custom.project.services.contract.api`

(for more information please see Mobiliser_Framework_5.1_Development_Guide.pdf section **5.4 Endpoint**)

```
package com.sybase365.mobiliser.custom.project.services.contract.api;

import javax.annotation.security.RolesAllowed;
import com.sybase365.mobiliser.custom.project.services.contract.v1_0.CreateDepartmentRequest;
import com.sybase365.mobiliser.custom.project.services.contract.v1_0.CreateDepartmentResponse;
import com.sybase365.mobiliser.custom.project.services.contract.v1_0.GetDepartmentRequest;
import com.sybase365.mobiliser.custom.project.services.contract.v1_0.GetDepartmentResponse;

public interface IDepartmentEndpoint {

    @RolesAllowed(value = "WS_CREATE_DEPARTMENT")
    CreateDepartmentResponse createDepartment(
            final CreateDepartmentRequest request);

    @RolesAllowed(value = "WS_GET_DEPARTMENT")
    GetDepartmentResponse getDepartment(final GetDepartmentRequest request);
}
```

3. Please apply java code formatter for greater code readability – press “Ctrl+A” to select all the code in Department.java and then press “Ctrl+Shift+F” for formatting the contents of the file.
4. To organize all the imports, please press “Ctrl+Shift+O” – please make sure that imports are correct.
5. Save the file DepartmentConverter.java

6. Navigate to *com.sybase365.mobiliser.custom.project.services.endpoint* and create the **DepartmentEndpoint.java** class implementing the endpoint interface and override the `getDepartment();` and `createDepartment();` method. The return type is [Create|Get]DepartmentResponse and the parameter is [Create|Get]DepartmentRequest. Don’t forget to add the injection check to the `afterPropertiesSet()` method

7. DepartmentEndpoint.java should be created in the following package:
`com.sybase365.mobiliser.custom.project.services.endpoint`

```
package com.sybase365.mobiliser.custom.project.services.endpoint;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.util.Assert;
import com.sybase365.mobiliser.custom.project.businesslogic.IDepartmentLogic;
import com.sybase365.mobiliser.custom.project.converter.IDepartmentConverter;
import com.sybase365.mobiliser.custom.project.services.contract.api.IDepartmentEndpoint;
import com.sybase365.mobiliser.custom.project.services.contract.v1_0.CreateDepartmentRequest;
import
com.sybase365.mobiliser.custom.project.services.contract.v1_0.CreateDepartmentResponse;
import com.sybase365.mobiliser.custom.project.services.contract.v1_0.GetDepartmentRequest;
import com.sybase365.mobiliser.custom.project.services.contract.v1_0.GetDepartmentResponse;
import com.sybase365.mobiliser.framework.gateway.security.api.ICallerUtils;

public class DepartmentEndpoint implements IDepartmentEndpoint,
       InitializingBean {
    private static final Logger LOG = LoggerFactory
.getLogger(DepartmentEndpoint.class);
    private ICallerUtils callerUtils;
    private IDepartmentLogic departmentLogic;
    private IDepartmentConverter departmentConverter;

    @Override
    public void afterPropertiesSet() throws Exception {
       Assert.notNull(this.callerUtils, "callerUtils is required");
       Assert.notNull(this.departmentLogic, "departmentLogic is required");
       Assert.notNull(this.departmentConverter, "departmentConverter is required");
    }
    @Override
    public CreateDepartmentResponse createDepartment(
           CreateDepartmentRequest request) {
       LOG.debug("#createDepartment() starts");
       final CreateDepartmentResponse response = new CreateDepartmentResponse();
       response.setId(Long.valueOf(this.departmentLogic.createDepartment(
              this.departmentConverter.fromContract(request.getDepartment()),
              this.callerUtils.getCallerId())));
       LOG.debug("#createDepartment created department id #{}", response
              .getId().toString());
       return response;
    }
    @Override
    public GetDepartmentResponse getDepartment(GetDepartmentRequest request) {
       LOG.debug("#getDepartment() starts");
       final GetDepartmentResponse response = new GetDepartmentResponse();
       response.setDepartment(this.departmentConverter
              .toContract(this.departmentLogic.getDepartment(request.getId(),
                     this.callerUtils.getCallerId())));
       LOG.debug("#getDepartment returned department id #{}", response
              .getDepartment().getId().toString());
       return response;
    }
    public void setDepartmentConverter(IDepartmentConverter departmentConverter) {
       this.departmentConverter = departmentConverter;
    }
    public void setDepartmentLogic(IDepartmentLogic departmentLogic) {
       this.departmentLogic = departmentLogic;
    }
    public void setCallerUtils(ICallerUtils callerUtils) {
       this.callerUtils = callerUtils;
    }
}
```

8. Please apply java code formatter for greater code readability – press “Ctrl+A” to select all the code in Department.java and then press “Ctrl+Shift+F” for formatting the contents of the file.
9. To organize all the imports, please press “Ctrl+Shift+O” – please make sure that imports are correct.
10. Save the file DepartmentEndpoint.java

NOW MOVE ON TO SPRING WIRING TASK

11. Open the **bundle-context-endpoint.xml** configuration in *com.sybase365.mobiliser.custom.project.service.endpoint* bundle and add both OSGi service references to the spring injection of the endpoint implementing class

```
(...)

<bean id="departmentImplDelegate"
class="com.sybase365.mobiliser.custom.project.services.endpoint.DepartmentEndpoint" >
    <property name="callerUtils" ref="callerUtils" />
    <property name="departmentLogic" ref="departmentLogic" />
    <property name="departmentConverter" ref="departmentConverter" />
</bean>

(...)
```

12. Copying/Pasting above xml would drop the indentation – please press “Ctrl+A” to select the contents and then press “Ctrl+I” for indentation of XML contents.
13. Save the file **bundle-context-endpoint.xml.**

14. Now open the OSGi **bundle-context-osgi.xml** configuration and add the businesslogic and converter interfaces to the service import.

```
(...)

<!-- OSGI Import -->
<osgi:reference id="departmentLogic"
interface="com.sybase365.mobiliser.custom.project.businesslogic.IDepartmentLogic" />

<osgi:reference id="departmentConverter"
interface="com.sybase365.mobiliser.custom.project.converter.IDepartmentConverter" />

(...)

<mobconfig:mobiliser-service id="departmentImpl"
       endpoint-ref="departmentImplDelegate" security-advisor="securityAdvisor"
              txn-advice="txAdvice" autodetect-interfaces="true" />

(...)

<!-- OSGI Exports -->
<osgi:service ref="departmentEndpoint"
interface="com.sybase365.mobiliser.framework.gateway.api.IEndpointInformation">
       <osgi:service-properties>
              <beans:entry key="path" value="/custom" />
       </osgi:service-properties>
</osgi:service>

(...)
```

15. Copying/Pasting above xml would drop the indentation – please press “Ctrl+A” to select the contents and then press “Ctrl+I” for indentation of XML contents.
16. Save the file **bundle-context-osgi.xml.**

17. Now open the **bundle-context-service-config.xml** configuration and add both OSGi service references to the spring injection of the endpoint implementing class

```
(...)

<bean id="departmentEndpoint" parent="abstractEndpoint"
class="com.sybase365.mobiliser.framework.gateway.api.impl.EndpointInformationImpl">
	<property name="endpoint" ref="departmentImpl" />
	<property name="allowedMessageElements">
		<list>
			<value>(Create|Get)Department(Request|Response)</value>
		</list>
	</property>
</bean>

(...)
```

18. Copying/Pasting above xml would drop the indentation – please press “Ctrl+A” to select the contents and then press “Ctrl+I” for indentation of XML contents.
19. Save the file **bundle-context-service-config.xml.**

20. Finally to allow this endpoint to be accessed successfully by the consumer with appropriate privileges, please create SQL scripts to insert the Privileges into `MOB_UMGR_PRIVILEGES` table.
21. Navigate to *com.sybase365.mobiliser.custom.project.persistence* bundle and to /src/main/schema/scripts/common/002_DATA
22. Create a new script file **062_PROJECT_PRIVILEGES_TRAINING.sql** to insert the new privileges

```
INSERT INTO MOB_UMGR_PRIVILEGES
(ID_PRIVILEGE,STR_PRIVILEGE,DAT_CREATION,ID_CUSTOMER_CREATION)
VALUES ('WS_CREATE_DEPARTMENT','Allows to create a new department',null,0);
INSERT INTO MOB_UMGR_ROLE_PRIVILEGES (ID_ROLE, ID_PRIVILEGE, ID_CUSTOMER_CREATION)
VALUES ('SYSTEM','WS_CREATE_DEPARTMENT', 0);

INSERT INTO MOB_UMGR_PRIVILEGES
(ID_PRIVILEGE,STR_PRIVILEGE,DAT_CREATION,ID_CUSTOMER_CREATION)
VALUES ('WS_GET_DEPARTMENT','Allows to retrieve a new department',null,0);
INSERT INTO MOB_UMGR_ROLE_PRIVILEGES (ID_ROLE, ID_PRIVILEGE, ID_CUSTOMER_CREATION)
VALUES ('SYSTEM','WS_GET_DEPARTMENT', 0);
```

23. Save the file **062_PROJECT_PRIVILEGES_TRAINING.sql**

BUILD AND CREATE RUNTIME ENVIRONMENT

1. Open a terminal (Applications/Accessories/Terminal)
2. Please shutdown the Mobiliser and Web containers (if running) before moving on to the next steps:
3. Change to the mobiliser money directory
   $ cd ~/workspace/custom
4. Run following command to build the Mobiliser R5 Vanilla reference application:
   $ mvn clean install
5. Once the maven shows "**BUILD SUCCESS**", go to the following directory:
   $ cd ~/workspace/custom/dist/target
6. You should see following two files:
   $ ls -ll
   com.sybase365.mobiliser.custom.project.dist-1.2.0-SNAPSHOT-dist-ase.zip
   com.sybase365.mobiliser.custom.project.dist-1.2.0-SNAPSHOT-scriptarchive-ase.jar

   The **.zip** file is a complete Mobiliser R5 runtime environment including Apache Felix (OSGI Server) and mobiliser deployable services.
   The **.jar** file contains all database scripts used by the dbmaintain utility.

7. Run dbmaintain: java -jar com.sybase365.mobiliser.custom.project.dist-1.2.0-SNAPSHOT-scriptarchive-ase.jar
8. Make sure that the new privileges are added to `MOB_UMGR_PRIVILEGES` using Squirrel SQL.

TEST THE SERVICE

After completing the implementation we can build the new dist package start the mobiliser server. Review the felix.out and mobiliser.log file for exceptions.
Note:

1. Unzip the newly created runtime environment in ~/workspace/custom/dist/target as follows:
   $ unzip com.sybase365.mobiliser.custom.project.dist-1.2.0-SNAPSHOT-dist-ase.zip
2. $ cd ~/workspace/custom/dist/target/com.sybase365.mobiliser.custom.project.dist-1.2.0-SNAPSHOT/money
3. Start the money container via bin/startup.sh
4. Wait until the server has started and then open the AIMS console at http://localhost:8080/system/console
5. Verify that all bundles are active
6. Verify that the business logic bundle exports the two new services
7. Try to get the WSDL: http://localhost:8080/mobiliser/custom/Custom.wsdl. Verify that the department service methods are available as designed
8. Start SoapUI and create a new project that uses the web service URL
9. In SoapUI change the request property values for "Authentication Type"="Preemptive" and provide values for username=mobiliser and password=secret
10. Try to invoke the two new web services

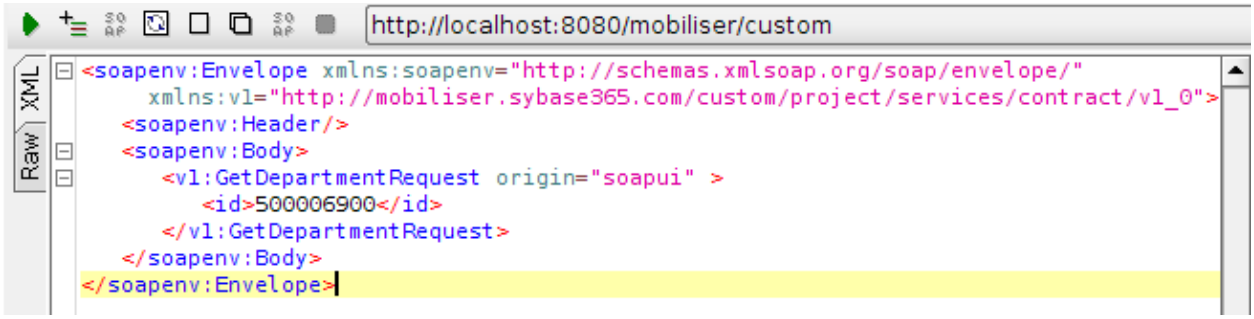
Create a New Department

The setup of SoapUI should look similar to the following:


```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
          xmlns:v1="http://mobiliser.sybase365.com/custom/project/services/contract/v1_0">
  <soapenv:Header/>
  <soapenv:Body>
    <v1:CreateDepartmentRequest origin="soapui"
    traceNo="0000000000016" repeat="false">
      <department>
        <name>test_department104</name>
      </department>
    </v1:CreateDepartmentRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Get the New Department

To “get” the newly created department a SoapUI “get” similar to the following can be used:




```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
          xmlns:v1="http://mobiliser.sybase365.com/custom/project/services/contract/v1_0">
  <soapenv:Header/>
  <soapenv:Body>
    <v1:GetDepartmentRequest origin="soapui" >
      <id>500006900</id>
    </v1:GetDepartmentRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

www.sap.com