Developing Physical Solutions for InfoSphere Master Data Management Server Advanced Edition v11

MDM Workbench Development Tutorial

John Beaven – IBM, Hursley

2013

Contents

Overview

In modern businesses, the volume of information being managed by an organization is increasing whilst at the same time this information is becoming ever more critical to a businesses success.

If this information is not managed and controlled, there is the potential for companies to make vital decisions based on incorrect data or become over whelmed by the scale of the information.

Master Data Management (MDM) solutions can provide a single, trusted view of critical business data and help organizations meet growth, revenue-generation and cost-reduction goals.

The MDM Server provides such a solution with a comprehensive range of out of the box services that help businesses manage and control their information.

The MDM Workbench allows these services to be enhanced or extended to tailor the capabilities for a specific deployment.

This tutorial shows how the MDM Workbench can be used to design a model of these additional capabilities and how these can be deployed to the MDM Server.

No prior experience of either the MDM Server or MDM Workbench is assumed

The instructions in this tutorial are broken down into two types of information. Those sections marked with a 'textpad' icon contain background information that you can choose to skip if if time is pressing and refer to later:

(BACKGROUND!)

Those sections marked with a 'user' icon contain the key steps that must be followed to complete the tutorial:

(KEY STEPS!)

Machine Requirements

The requirements for running MDM V11 are as follows:

- 8GB-12GB of memory
- 50GB of dedicated disk space required for a typical installation

To set-up a machine for this tutorial, if MDM is not already installed, then it is recommended to run a 'Typical Installation'. Instructions for this can be found by following the link below:

https://www.ibm.com/developerworks/community/blogs/64033fdd-4a35-4733-852f-a39abcdf4fb3/entry/installing_mdm_workbench_v11?lang=en

System Configuration

This tutorial assumes you have completed a typical installation of MDM V11 including the software listed below:

- IBM Rational Application Developer for WebSphere Software Version 8.5.1
- IBM WebSphere Application Server Version 8.5.0.2
- IBM InfoSphere Master Data Management Workbench Version 11
- InfoSphere MDM Standard Edition or Advanced Edition Version 11
- IBM DB2 Version 10.1

For a typical installation the default user ids and passwords are as follows:

- DB2 User ID: **db2admin**
- DB2 Password: **db3Admin**
- WebSphere User ID: **mdmadmin**
- WebSphere Password: **mdmadmin**

Changes in Version 11 (BACKGROUND!)

What's this section about? *Discusses some changes since the last version of the workbench. If you have not used the workbench before you can skip this section.*

One of the most apparent changes when using the latest version of the workbench is that you no longer need to use the DEST utility to prepare the development environment. Unlike previous releases where an MDM Enterprise Archive (EAR) was unpacked into the workspace, version 11 uses an Enterprise Bundle Archive (EBA) stored outside the workspace to provide the basis for development. The necessary dependencies on the EBA are established automatically when the workbench is installed. Modules developed in the workspace are combined into a Composite Bundle Archive (CBA) which can be deployed alongside the main server EBA to customize MDM.

Some up-front Terminology (BACKGROUND!)

What's this section about? *Provides an introduction to some of the core components that make up a solution. Skip this section if required and refer to it as necessary.*

Entity Type – A physical entity in the MDM data model. As supplied the MDM Server supports a range of different Entity Types such as Person, Contract or Group. Additional Entity Types may be created in the Workbench to extend the data model.

Attribute – A field defined for an Entity Type or Entity Type Extension

Entity Type Extension – An extension to an out of the box Entity Type. Typically this mechanism is used to add additional attributes required by a solution that are not present in the out of the box data model. For example, if a Contract needed an additional contract classification field, an Entity Type Extension could be defined to add this. This should not be confused with a subtype, which is a separate concept. Creating an Entity Type Extension of Contract allows additional fields to be added to requests and responses that include instances of a Contract. The extension however is not considered to be a separate type in its own right. Additional fields are included in a generic "extended block" that's common to all Entity Type definitions.

Code Table – A structure used to contain a set of String values that can be referred to by a Type Code. Conceptually similar to an enumeration in Java, these structures are used to hold well defined sets of values that don't change often. For example, we could create a Code Table to hold a set of colors ("Red", "Green", "Blue", "Orange") Once deployed each Code Table entry has both a "value" – the String representation and a “type” – essentially the numerical database key. For example, (1, Red) or (2, Green)

Type Code – A reference to an entry within a Code Table. When referencing a Code Table, either the Code Table “Type” or “Value” may be specified. If both are present, they must match.

Transaction – A service or operation that can be executed by a client on the MDM Server. For example, when adding a person to the database, users would invoke the “addPerson” transaction. XML is used to define a request which can be passed to the server over a number of different interfaces (eg a WebService call)

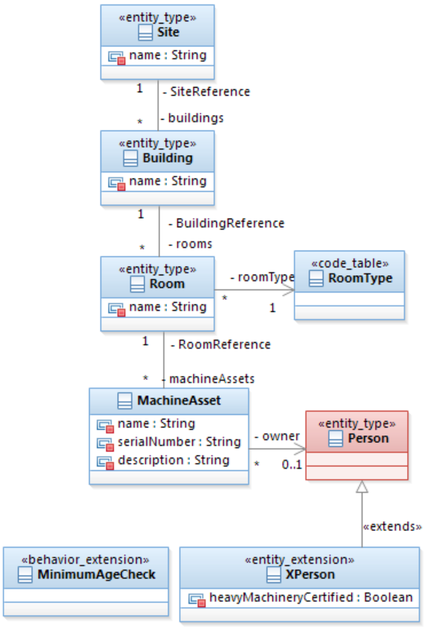
Behaviour Extension – Allows the default services provided by MDM to customized by executing code before and/or after the core service runs. For example, we could add additional validation to ensure that specific constrains are met when performing an addContract transaction.

Solution Overview (BACKGROUND!)

What's this section about? *Provides an outline of the solution that you will build in this tutorial. If time is pressing, you can skip this and simply follow the instructions in further sections below. Its recommended that you do at least review the model diagram in this section.*

For this tutorial, we have created a simplified model that consists of a range of different elements supported by the workbench. The tutorial shows how this model is defined and converted into code. It demonstrates how this code can be modified to customize the solution and how this can then be deployed to and tested on a server.

To keep things simple and allow you to focus on the mechanics involved, we minimized the amount of additional complexity that might be found in a real-world scenario. During the course of this of this tutorial we will build following model:



This model reflects a simplified asset tracking system used to manage machinery in a factory complex.

Site – A new Entity Type introduced to hold information about a factory or plant location.

Building – A new Entity Type used to hold details about a building at a Site

Room – A new Entity Type used to hold details of a specific Room within a Building

MachineAsset – A new Entity Type used to hold details of a physical machine asset within a room.

RoomType – A Code Table used to define the set of possible room classifications (“Office”, “PlantRoom”, “MeetingRoom”, “StoreRoom”)

XPerson – MDM provides an “of the box” Entity Type called Person that holds details of individuals that exist in the database. **XPerson** is an Entity Type Extension of Person that defines an extra attribute not present in the standard data model. In this case, it’s a field that indicates if a person has been certified to operate heavy machinery.

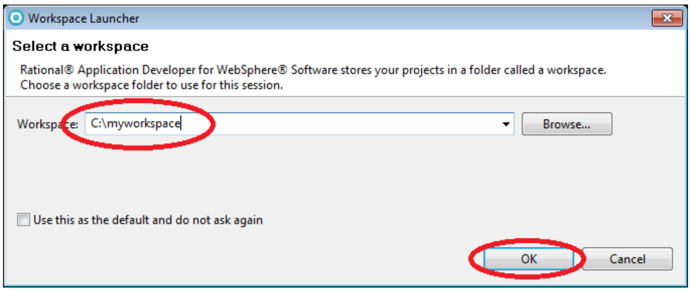
MinimumAgeCheck - Behaviour extension to ensure that all employees are over 16. Additional logic will run prior to the “addPerson” transaction to enforce this.

Step 1 - Setting up the Workspace (KEY STEPS!)

What’s this section about? *The workspace will be prepared for development. You must follow these steps to complete the tutorial.*

- Start RAD or RSA by running the application from the Desktop or Start menu

- When the “**Workspace Launcher**” dialog appears, select a <u>new</u> folder to contain the projects developed in this tutorial and click **OK**



Workspace Launcher

Select a workspace

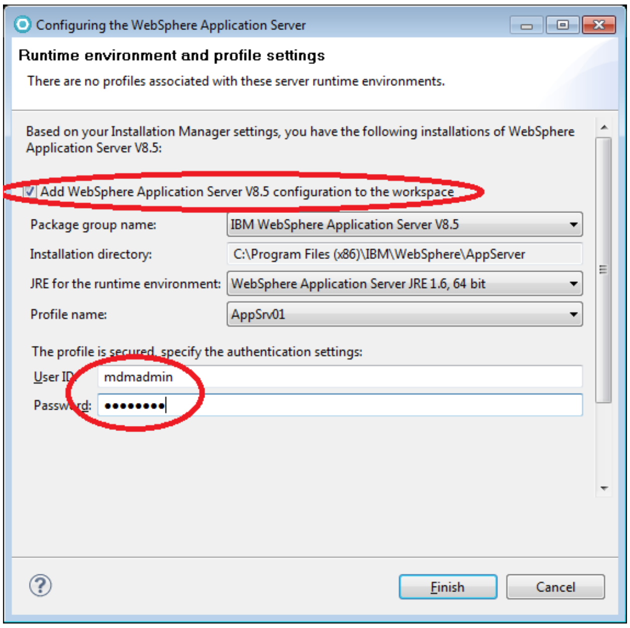
Rational® Application Developer for WebSphere® Software stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\myworkspace

☐ Use this as the default and do not ask again

Each time a brand new workspace is opened, the **Configuring the WebSphere Application Server** dialog will appear. The next steps show how to use this to create a connection to the WebSphere server configured on the local machine:

- In the **Configuring the WebSphere Application Server** dialog, ensure the "**Add WebSphere Application Server V8.5 configuration to the workspace**" option is checked.
- **Specify the user id and password, for a typical installation the defaults are "mdmadmin" and password "mdmadmin" under the authentication settings and click Finish**



Configuring the WebSphere Application Server

Runtime environment and profile settings

There are no profiles associated with these server runtime environments.

Based on your Installation Manager settings, you have the following installations of WebSphere Application Server V8.5:

☑ Add WebSphere Application Server V8.5 configuration to the workspace

Package group name: IBM WebSphere Application Server V8.5

Installation directory: C:\Program Files (x86)\IBM\WebSphere\AppServer

JRE for the runtime environment: WebSphere Application Server JRE 1.6, 64 bit

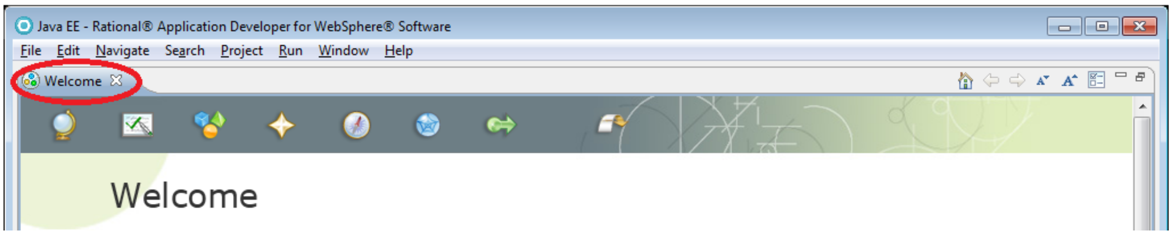
Profile name: AppSrv01

The profile is secured, specify the authentication settings:

User ID: mdmadmin

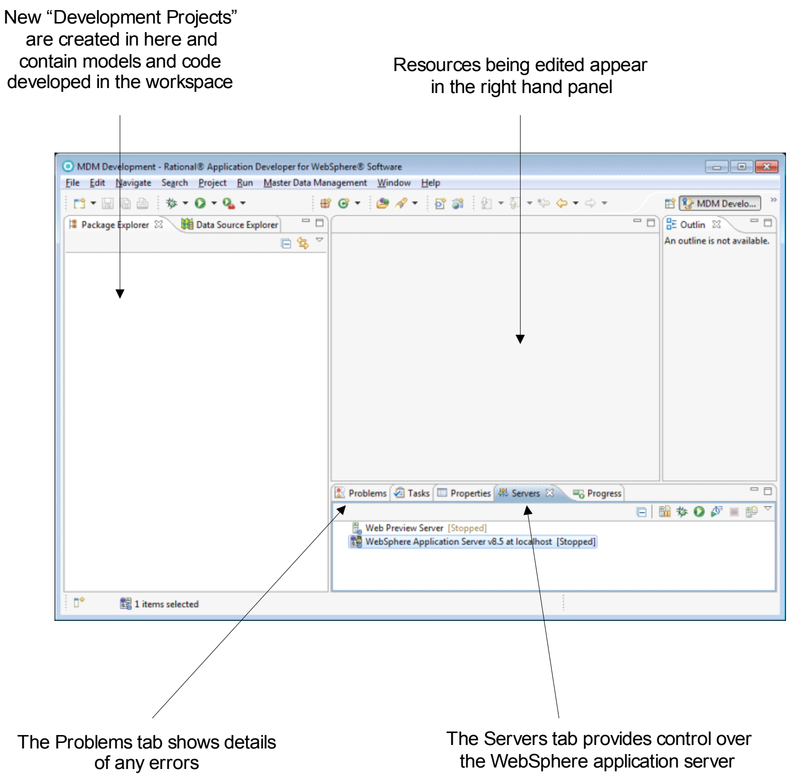
Password: ••••••••

- Once the workspace opens, close the Welcome screen by clicking on the cross on the **Welcome** tab.



Java EE - Rational® Application Developer for WebSphere® Software

You should now see an empty workspace as shown below:



New “Development Projects” are created in here and contain models and code developed in the workspace

Resources being edited appear in the right hand panel

The Problems tab shows details of any errors

The Servers tab provides control over the WebSphere application server

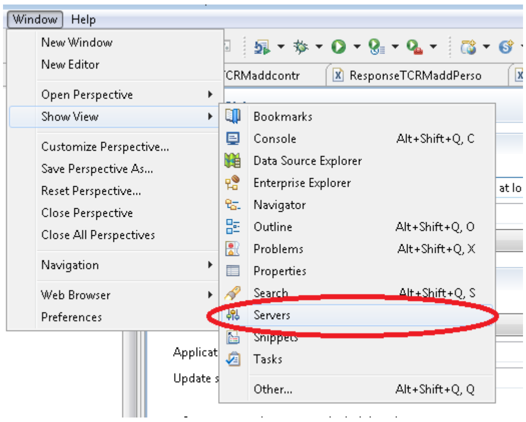
MDM Development - Rational® Application Developer for WebSphere® Software
File Edit Navigate Search Project Run Master Data Management Window Help
MDM Develo...
Package Explorer
Data Source Explorer
Outlin
An outline is not available.
Problems
Tasks
Properties
Servers
Progress
Web Preview Server [Stopped]
WebSphere Application Server v8.5 at localhost [Stopped]
1 items selected

Step 2 – Configuring the Server Connection (KEY STEPS!)

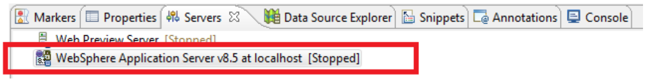
What’s this section about? *The WebSphere server connection definition will be updated to control how code is deployed from the workspace to the server*

The server connection created by the “**Configuring the WebSphere Application Server**” dialog above needs some fine tuning. We need to change the settings so that binary code developed in the workspace is copied to the WAS server rather than being linked to it. The distinction may seem subtle, but the effect is to de-couple code developed in the workspace with the code deployed on the server. The server deployment will remain valid and functional even if the workspace is deleted (if it was linked to a workspace that was deleted, this would cause problems). The following steps show how to do this:

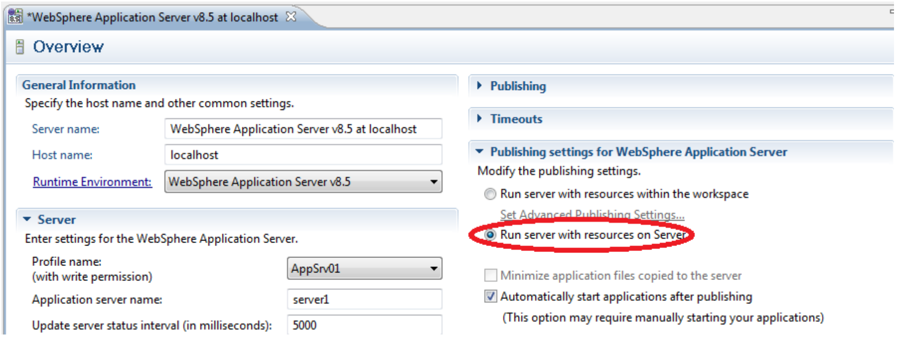
- Open the **Servers** view from the RSA main menu **Window -> Show View -> Servers**



- Double click on the WebSphere Application Server v8.5 in **Servers** view to open the configuration settings for the server connection.


Markers
Properties
Servers
Data Source Explorer
Snippets
Annotations
Console
Web Preview Server [Stopped]
WebSphere Application Server v8.5 at localhost [Stopped]

- Under **Publishing settings for WebSphere Application Server**, ensure the **Run server with resources on Server** option is selected.


*WebSphere Application Server v8.5 at localhost

Overview

General Information

Specify the host name and other common settings.

Server name: WebSphere Application Server v8.5 at localhost

Host name: localhost

Runtime Environment: WebSphere Application Server v8.5

Server

Enter settings for the WebSphere Application Server.

Profile name: (with write permission) AppSrv01

Application server name: server1

Update server status interval (in milliseconds): 5000

Publishing

Timeouts

Publishing settings for WebSphere Application Server

Modify the publishing settings.

○ Run server with resources within the workspace

Set Advanced Publishing Settings...

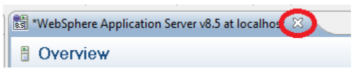
● Run server with resources on Server

□ Minimize application files copied to the server

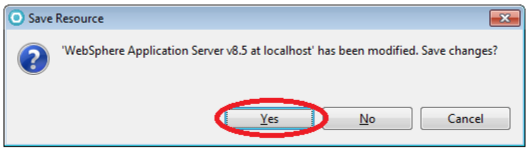
☑ Automatically start applications after publishing

(This option may require manually starting your applications)

- Close the WebSphere Application Server v8.5 configuration



- Save the configuration settings when prompted by clicking **Yes**


Save Resource

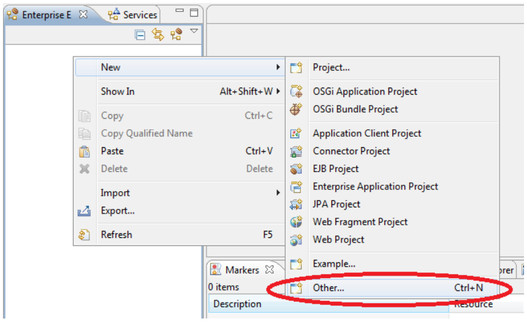
'WebSphere Application Server v8.5 at localhost' has been modified. Save changes?

The workspace is now configured and you are ready to start developing code!

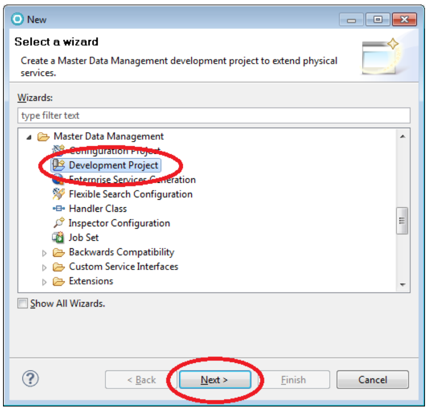
Step 3 - Creating the "Development Model" (KEY STEPS!)

What's this section about? *A new Development project containing a development model (previously referred to as a Hub Model) will be created in the workspace. You must follow these steps to complete the tutorial.*

- Open the "context menu" (right hand mouse button) in the **Package Explorer** or **Navigator** view (left hand portion of RAD window)
- Select **New** -> **Other…** from the menu

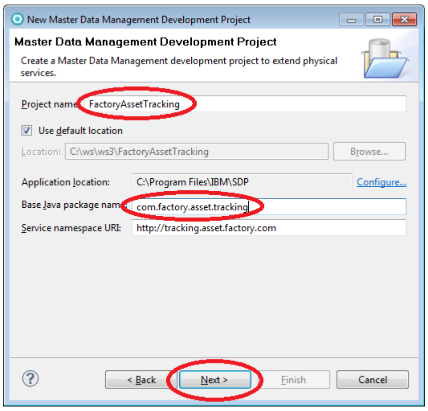


- In the **New** wizard, select **Master Data Management ->Development Project** wizard and click **Next**



Create a Master Data Management development project to extend physical services.

- In the **New Master Data Management Development Project** window enter the project name "**FactoryAssetTracking**", the **Base Java package** name "**com.factory.asset.tracking**" and click **Next**



New Master Data Management Development Project

Master Data Management Development Project

Create a Master Data Management development project to extend physical services.

Project name: FactoryAssetTracking

☑ Use default location

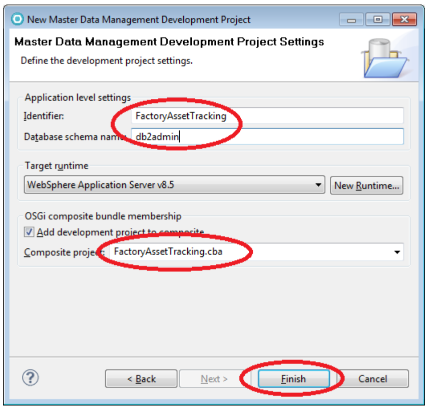
Location: C:\ws\ws3\FactoryAssetTracking

Application location: C:\Program Files\IBM\SDP

Base Java package name: com.factory.asset.tracking

Service namespace URI: http://tracking.asset.factory.com

- In the following panel enter "**FactoryAssetTracking**" for the **Identifier** (previously Hub Base Name) and **db2admin** for the **Database schema name**. Ensure that **Add development project to composite** is checked and that the **Composite project** is set to **FactoryAssetTracking.cba**. Click **Finish**



New Master Data Management Development Project

Master Data Management Development Project Settings

Define the development project settings.

Application level settings

Identifier: FactoryAssetTracking

Database schema name: db2admin

Target runtime

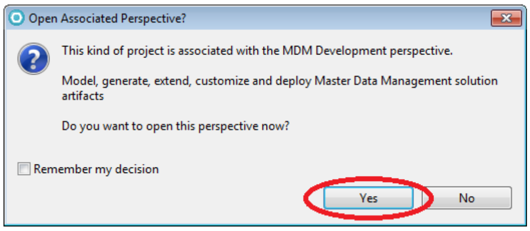
WebSphere Application Server v8.5

OSGi composite bundle membership

☑ Add development project to composite

Composite project: FactoryAssetTracking.cba

Hint: Once the project has been created you will be prompted to change to the MDM Development perspective. You should click ***Yes*** *to do this as certain menu options will not be visible otherwise.*



Open Associated Perspective?

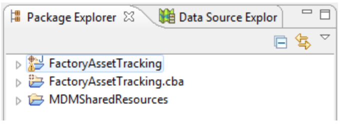
This kind of project is associated with the MDM Development perspective.

Model, generate, extend, customize and deploy Master Data Management solution artifacts

Do you want to open this perspective now?

□ Remember my decision

When the wizard completes you will see that three projects have been created:



FactoryAssetTracking

FactoryAssetTracking.cba

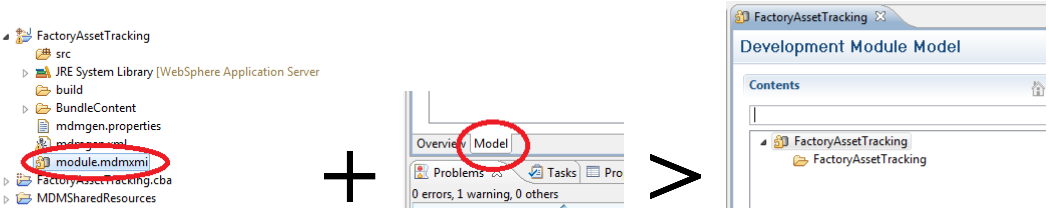
MDMSharedResources

FactoryAssetTracking – This is the "Development Project" that contains the solution model file (module.mdmxmi) and eventually generated code.

FactoryAssetTracking.cba – This is the composite bundle archive that will group together one or more development projects for subsequent deployment - the CBA is deployed alongside the MDM EBA to customize MDM. Although you can have more than one CBA, its recommended that you add further development projects to the same CBA project.

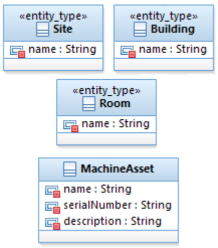
MDMSharedResources – This is similar to CustomerResources from previous versions of the MDM. It groups together content that's solution wide rather than project specific (schema, SQL etc). Multiple projects contribute content.

- Open the **module.mdmxmi** file in the **FactoryAssetTracking** project and select the **Model** tab. This will open the empty development model in the right hand panel

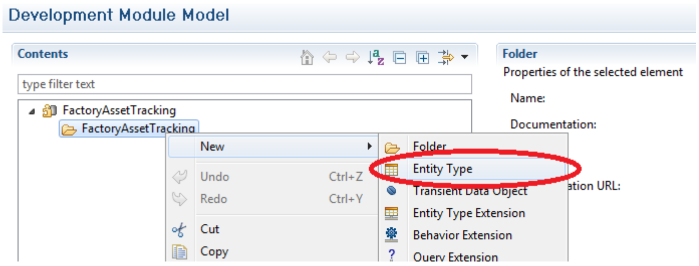


Step 4 - Creating the four new Entity Types (KEY STEPS!)

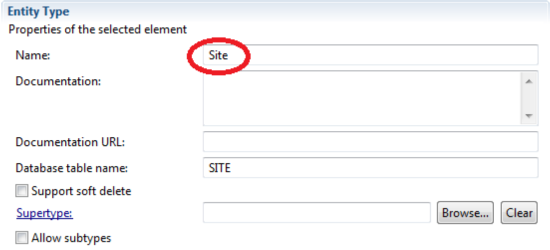
What's this section about? *In this section of the tutorial we will define the four new Entity Types that form the core of the solution. These are* ***Site****,* ***Building****,* ***Room*** *and* ***MachineAsset****. Initially these will be created without the relationships between them which will be added later. You must follow these steps to complete the tutorial.*



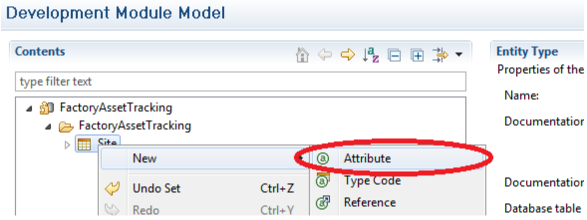
- Open the context menu on the **FactoryAssetTracking** subfolder (shown below) and select **New -> Entity Type**



- In the right hand panel, change the name of the new Entity Type to **Site**

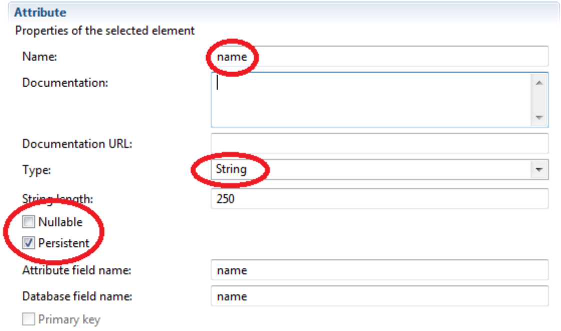

Entity Type
Properties of the selected element
Name: Site
Documentation:
Documentation URL:
Database table name: SITE
□ Support soft delete
Supertype:
□ Allow subtypes

- Open the context menu on the **Site** Entity Type in the model tree view and select **New -> Attribute**



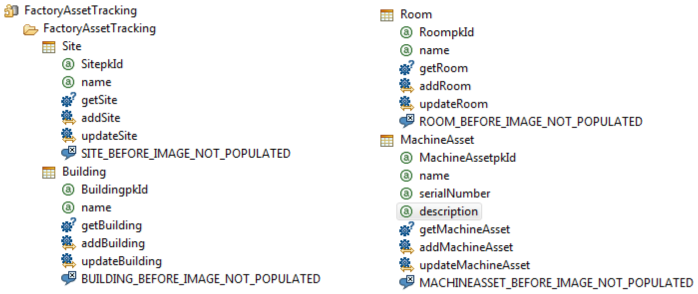
- In the right hand panel, change the attribute name to **name**, the type to **String**, ensure that the **Nullable** field is not checked and that the **Persistent** field is checked.

Tip: The default string length of 250 characters is fine for all String types you create in this tutorial


Attribute
Properties of the selected element
Name: name
Documentation:
Documentation URL:
Type: String
String length: 250
□ Nullable
☑ Persistent
Attribute field name: name
Database field name: name
□ Primary key

- Repeat this process to create the Entity Types **Building** and **Room**, both of which should have a single **String** attribute called **name**.
- Repeat this process one more time to create the Entity Type **MachineAsset** which should have attributes named **name** (String), **serialNumber** (String) and **description** (String).

The Model should now look like this:



- FactoryAssetTracking
  - FactoryAssetTracking
    - Site
      - SitepkId
      - name
      - getSite
      - addSite
      - updateSite
      - SITE_BEFORE_IMAGE_NOT_POPULATED
    - Building
      - BuildingpkId
      - name
      - getBuilding
      - addBuilding
      - updateBuilding
      - BUILDING_BEFORE_IMAGE_NOT_POPULATED
    - Room
      - RoompkId
      - name
      - getRoom
      - addRoom
      - updateRoom
      - ROOM_BEFORE_IMAGE_NOT_POPULATED
    - MachineAsset
      - MachineAssetpkId
      - name
      - serialNumber
      - description
      - getMachineAsset
      - addMachineAsset
      - updateMachineAsset
      - MACHINEASSET_BEFORE_IMAGE_NOT_POPULATED

Tip: The workbench automatically adds primary key fields and basic Create/Read/Update/Delete (CRUD) style transactions for each entity you create. For example you can add a new machine asset using the addMachineAsset transaction or query rooms by primary key using the getRoom transaction. These basic CRUD transactions are available for all entity types developed in the workbench.

An introduction to containment structures (BACKGROUND!)

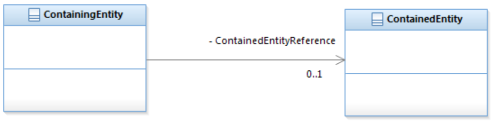
What's this section about? *This section provides a high level overview of what containment relationships are and how they are represented in the development model. Skip this section if required and refer to it as necessary.*

In this model, a Site consists of a set of Buildings, a Building consists of a set of Rooms and a Room contains a set of MachineAssets. We can establish these relationships as “containment” relationships where we say a “containing” entity such as Building “contains” another “contained” entity such as Room.

Exactly how this is modelled depends on the “cardinality” of the relationships and here there are two key scenarios. Depending on whether the relationship is a “one-to-one” or “one-to-many”, the workbench will create different model content when the containment is defined.

“One to one” relationship:

In this case, the containing entity has a single contained entity underneath it. This can be captured in the model as a simple reference from the containing entity to the contained entity as shown below:

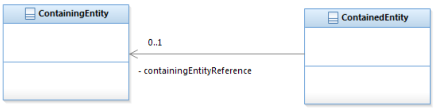


For example, if we assert that a Room only ever houses a single MachineAsset, we could have a reference to a MachineAsset from Room. This is simplest case.

“One to Many” relationship:

In many cases however, this is not realistic. For example a Room might well contain many different MachineAssets. Whilst it is possible to create multiple references from a ContainingEntity to different instances of the ContainedEntity, it is not possible to create an arbitrary number of such references. Using this approach we would have to know in advance exactly how many references were required and give them all separate names in the model – this is impractical and messy.

The solution is to turn the problem around and have a reference from the contained entity to the containing entity as shown in the model below:



For example a MachineAsset would have a reference to the Room in which it resides. By doing this, we can associate an arbitrary number of MachineAssets with a single Room.

As mentioned above, the workbench creates simple CRUD transactions to support the data model. For example, the “getRoom” transaction returns a specific Room given a primary key. If you introduce a one-to-many containment relationship, you will see an additional transaction gets added to the model under the contained entity. This additional transaction provides a service to retrieve all instances of the contained entity that reference a given containing entity.

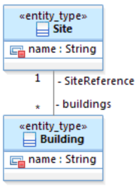
For example, when the transaction getRoom(53) is invoked to retrieve the Room with primary key 53, it will include in the response all those MachineAssets that reside in this specific room – derived by the reference from MachineAsset to Room. In this case the transaction would be called “**getMachineAssetByRoom**” and would return all those instances of MachineAsset that have a reference to Room 53.

Step 5 - Creating “containment” relationships (KEY STEPS!)

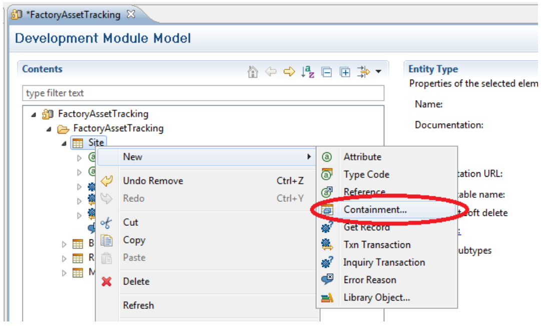
What's this section about? *The “containment” relationships between these key Entity Extensions will be added to the model. You must follow these steps to complete the tutorial.*

To simplify this process, a wizard dialog is provided to automatically create the required model components for a given containment structure. We will use this wizard now to build up these relationships.

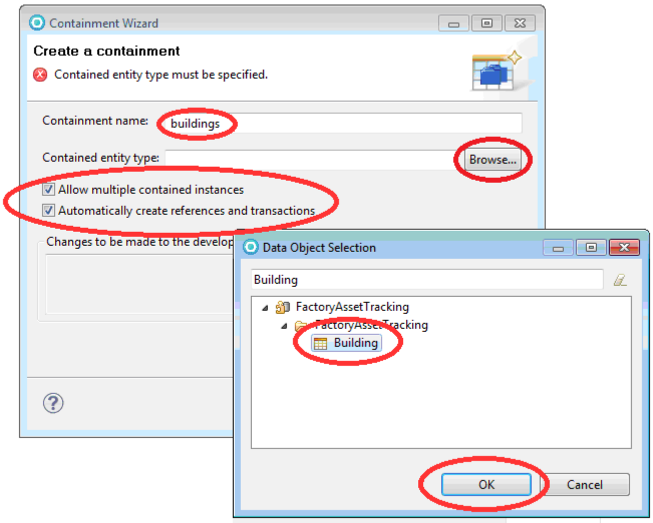
First, we will create the containment of **Building** within a **Site:**



- Open the context menu on the **Site** Entity Type and select New -> **Containment…**



- In the **Containment Wizard** dialog, enter a containment name **buildings**
- Ensure the **Allow multiple contained references** option is checked
- Ensure the **Automatically create references and transactions** option is checked
- Click the **Browse**… button and select the **FactoryAssetTracking -> FactoryAssetTracking -> Building** Entity Type.
- Click **OK**



Containment Wizard

Create a containment

Contained entity type must be specified.

Containment name: buildings

Contained entity type:

☑ Allow multiple contained instances

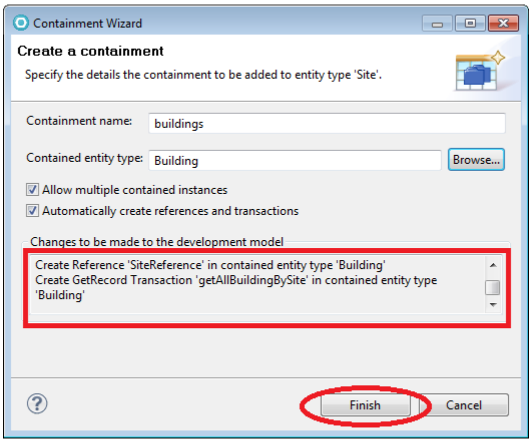
☑ Automatically create references and transactions

Changes to be made to the develop

Data Object Selection

Building

- The **Containment Wizard** panel should then show a summary of the changes that will be made to the model. Check this matches those listed below and click **Finish**



Containment Wizard

Create a containment

Specify the details the containment to be added to entity type 'Site'.

Containment name: buildings

Contained entity type: Building

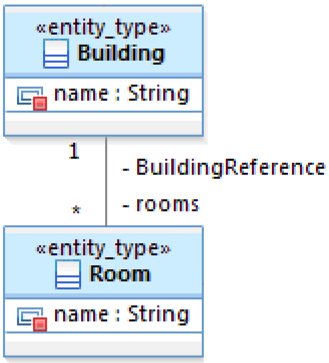
☑ Allow multiple contained instances

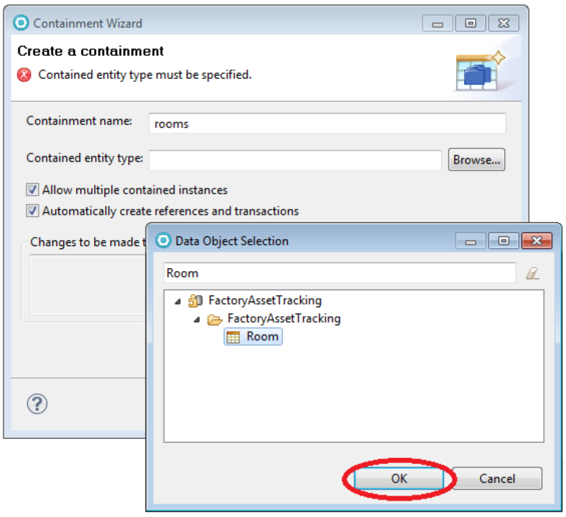
☑ Automatically create references and transactions

Changes to be made to the development model

Create Reference 'SiteReference' in contained entity type 'Building'
Create GetRecord Transaction 'getAllBuildingBySite' in contained entity type 'Building'

- Open the context menu on the **Building** Entity Type and repeat this process to create a new containment of **Room** named **rooms**.





Containment Wizard

Create a containment

Contained entity type must be specified.

Containment name: rooms

Contained entity type:

☑ Allow multiple contained instances

☑ Automatically create references and transactions

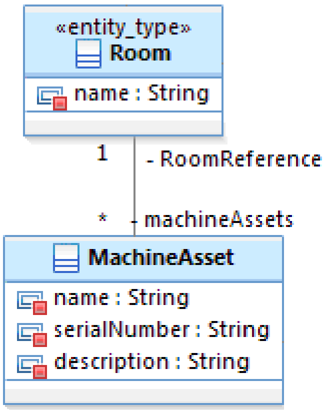
Changes to be made t

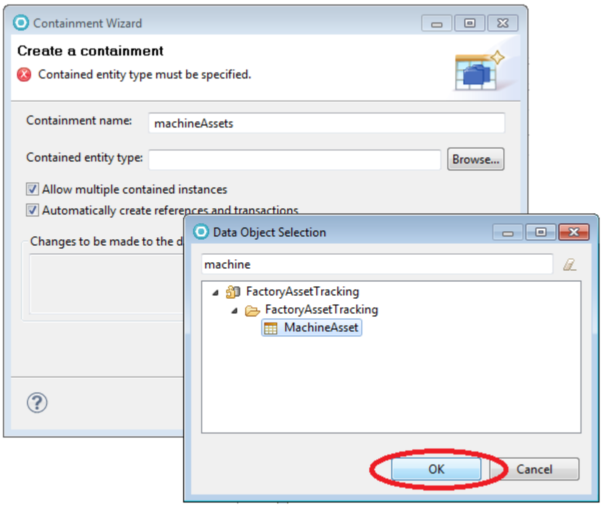
Data Object Selection

Room

FactoryAssetTracking
FactoryAssetTracking
Room

- Open the context menu on the **Room** Entity Type and repeat this process to create a new containment of **MachineAsset** named **machineAssets**.





Containment Wizard

Create a containment

Contained entity type must be specified.

Containment name: machineAssets

Contained entity type:

☑ Allow multiple contained instances

☑ Automatically create references and transactions

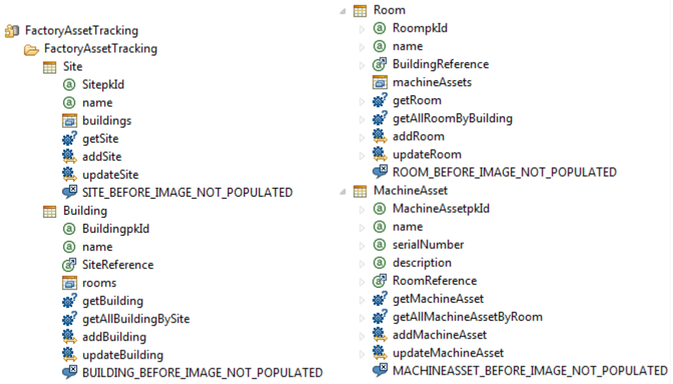
Changes to be made to the d

Data Object Selection

machine

FactoryAssetTracking
FactoryAssetTracking
MachineAsset

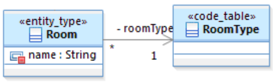
The model should now look like this:



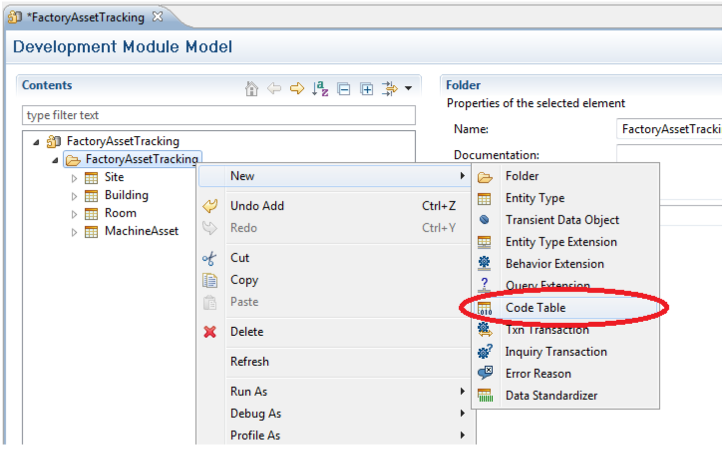
```
FactoryAssetTracking
  FactoryAssetTracking
    Site
      SitepkId
      name
      buildings
      getSite
      addSite
      updateSite
      SITE_BEFORE_IMAGE_NOT_POPULATED
    Building
      BuildingpkId
      name
      SiteReference
      rooms
      getBuilding
      getAllBuildingBySite
      addBuilding
      updateBuilding
      BUILDING_BEFORE_IMAGE_NOT_POPULATED
    Room
      RoompkId
      name
      BuildingReference
      machineAssets
      getRoom
      getAllRoomByBuilding
      addRoom
      updateRoom
      ROOM_BEFORE_IMAGE_NOT_POPULATED
    MachineAsset
      MachineAssetpkId
      name
      serialNumber
      description
      RoomReference
      getMachineAsset
      getAllMachineAssetByRoom
      addMachineAsset
      updateMachineAsset
      MACHINEASSET_BEFORE_IMAGE_NOT_POPULATED
```

Step 6 – New Code Table & Type Code for Room (KEY STEPS!)

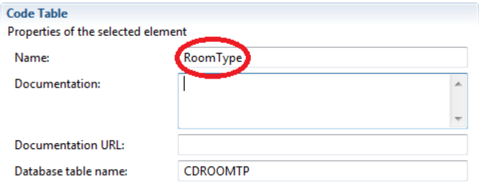
What's this section about? *A code table is created to define the possible room types that can be present in a building. We then create a "Type Code" reference to this code type in the Room entity type. These steps must be followed to complete the tutorial.*



- Open the context menu over the **FactoryAssetTracking** subfolder (see below) and select **New -> Code Table**



- Select the new code table entry in the tree view and in the right hand panel, change the name to **RoomType**



Code Table

Properties of the selected element

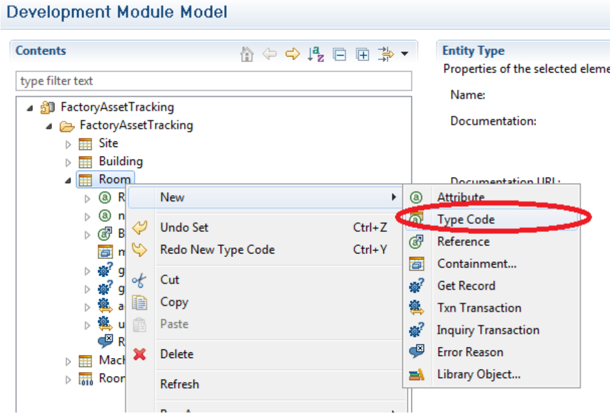
Name: RoomType

Documentation:

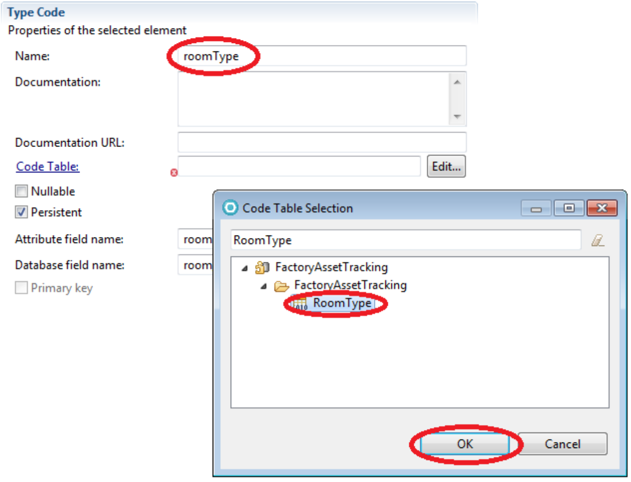
Documentation URL:

Database table name: CDROOMTP

- Open the context menu on the **Room** Entity Type and select **New -> Type Code**



- Select the new type code in the tree view and in the right hand panel change the name to **roomType**
- Click on the **Edit** button, **select FactoryAssetTracking -> FactoryAssetTracking -> RoomType** in the Code Table Selection dialog and click **OK**



Type Code

Properties of the selected element

Name: roomType

Documentation:

Documentation URL:

Code Table: Edit...

☐ Nullable

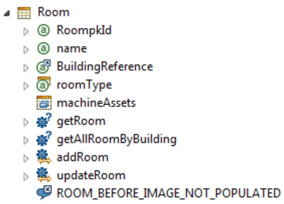
☑ Persistent

Attribute field name: room

Database field name: room

☐ Primary key

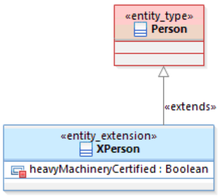
The model for the **Room** Entity Type should now look like this:



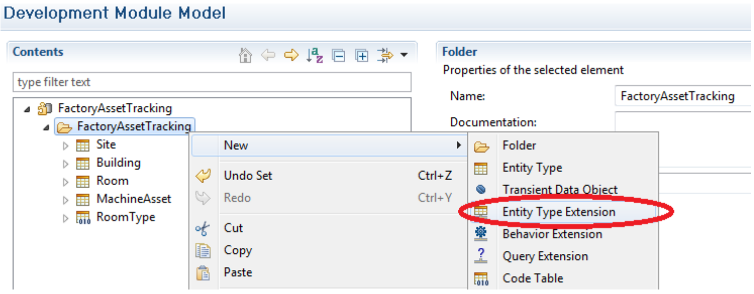
Tip: We will add the various room types later once we have generated the code. This is not something we capture in the model.

Step 7 - Extending the "Person" entity type (KEY STEPS!)

What's this section about? *In our model we need to be able to capture whether or not a person has been trained to operate heavy machinery. We will create an Entity Type Extension of the Person Entity Type to hold this information. These steps must be followed to complete the tutorial.*

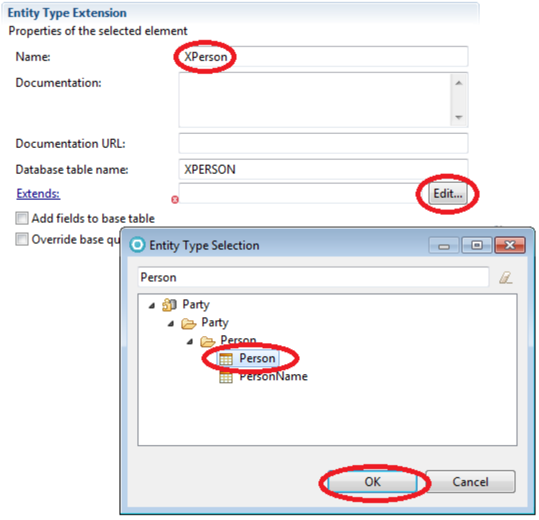


- Open the context menu on the **FactoryAssetTracking** subfolder (see below) and select **New -> Entity Type Extension**



Name: FactoryAssetTracking

- Select the new Entity Type Extension in the tree view and in the right hand panel, change the name to **XPerson**
- Click the **Edit** button and in the **Entity Type Selection** dialog select **Party -> Party -> Person -> Person**
- Click **OK**

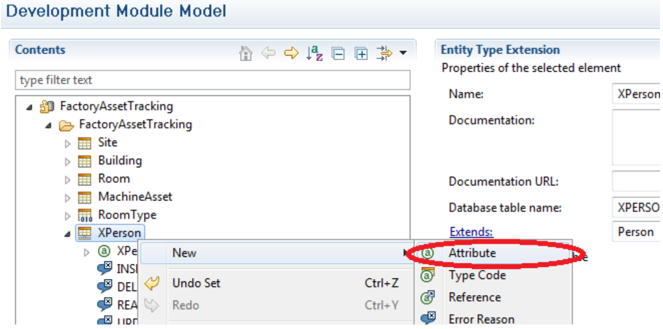


Name: XPerson

Database table name: XPERSON

Person

- Open the context menu on the **XPerson** Entity Type Extension and select **New -> Attribute**



Name: XPerson

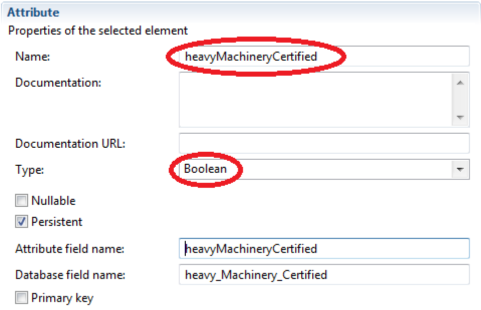
Documentation:

Documentation URL:

Database table name: XPERSO

Extends: Person

- Select the new attribute in the tree view and in the right hand panel change the name to **heavyMachineryCertified** and the type to **Boolean**.



Name: heavyMachineryCertified

Documentation:

Documentation URL:

Type: Boolean

□ Nullable

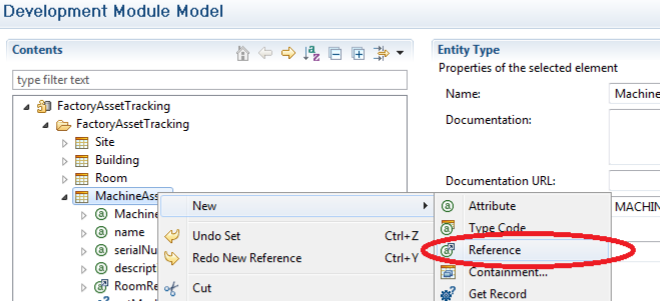
☑ Persistent

Attribute field name: heavyMachineryCertified

Database field name: heavy_Machinery_Certified

□ Primary key

- Open the context menu on the **MachineAsset** Entity Type and select **New -> Reference**



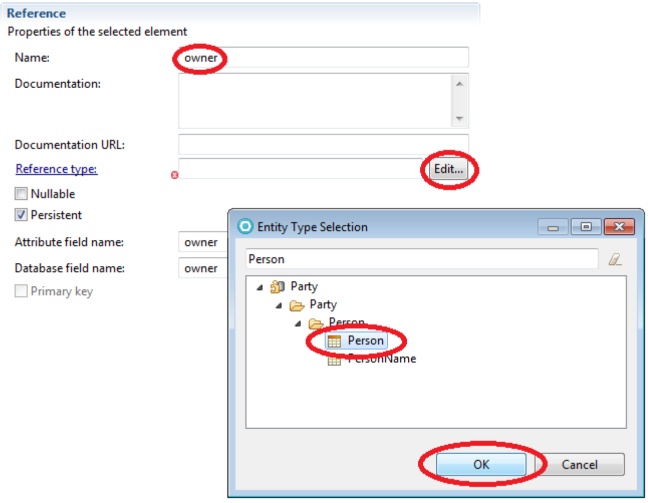
Name: Machine

Documentation:

Documentation URL:

MACHIN

- Select the new reference in the tree view and in the right hand panel change the name to **owner**
- Click the **Edit** button and in the **Entity Type Selection** dialog select **Party -> Party -> Person -> Person** and click **OK**



Reference

Properties of the selected element

Name: owner

Documentation:

Documentation URL:

Reference type: Edit...

□ Nullable

☑ Persistent

Attribute field name: owner

Database field name: owner

□ Primary key

Entity Type Selection

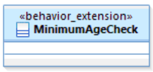
Person

- Party
  - Party
    - Person
      - Person
      - PersonName

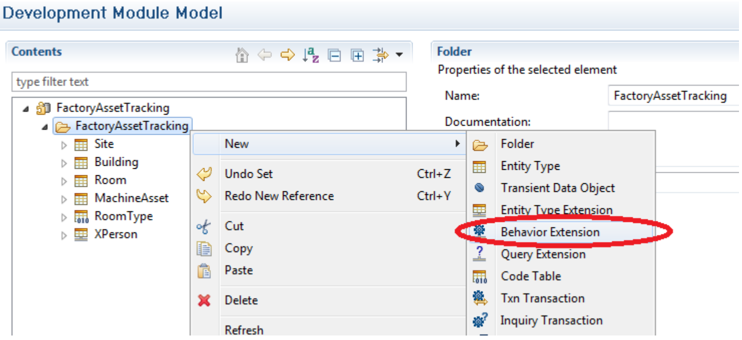
Tip: Although we have extended Person with XPerson, remember that extensions are really nothing more than "adjustments" to the underlying entity type. They should not be thought of as subtypes which are a separate concept. For this reason we are creating a reference to Person and not to XPerson. We are still able to pass the extended field value for 'heavyMachinaryCertified" when creating instances of Person.

Step 8 - Enforcing 'addPerson' transaction validation (KEY STEPS!)

What's this section about? *We create a "behaviour extension" in the model which will allow us to introduce some additional validation logic to the out of the box "addPerson" transaction. The logic itself will be added later as Java code. For the moment we must create a place-holder in the model. These steps must be followed to complete the tutorial.*



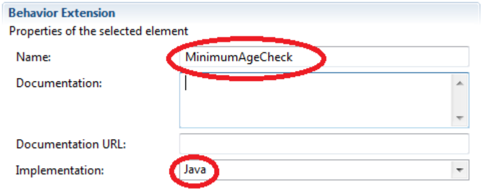
- Open the context menu on the **FactoryAssetTracking** subfolder (see below) and select **New -> Behaviour Extension**



Name: FactoryAssetTracking

Documentation:

- Select the new Behaviour Extension in the tree view and in the right hand panel change the name to **MinimumAgeCheck** and the implementation type to **Java**



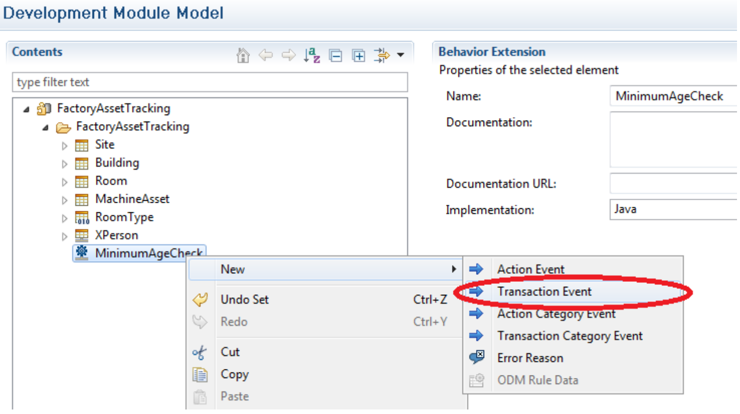
Name: MinimumAgeCheck

Documentation:

Documentation URL:

Implementation: Java

- Open the context menu on the **MinimumAgeCheck** Behaviour Extension and select **New -> Transaction Event**



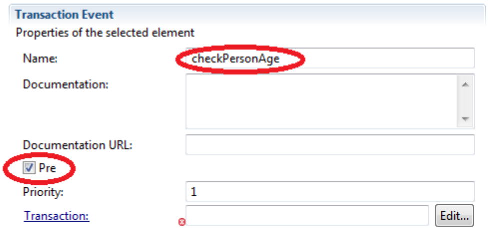
Name: MinimumAgeCheck

Documentation:

Documentation URL:

Implementation: Java

- Select the new transaction event in the tree view and in the right hand panel set the name to **checkPersonAge**
- Ensure the **Pre** option is checked
- Click the **Edit** button and in the **Transaction Selection** dialog select the **Party -> CoreParty -> addPerson** transaction. Click **OK**.



Transaction Event

Properties of the selected element

Name: checkPersonAge

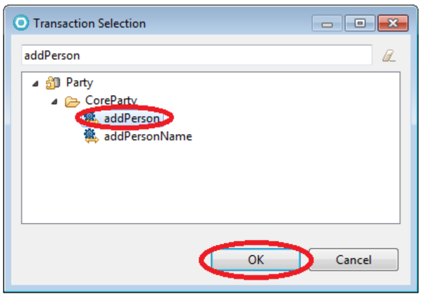
Documentation:

Documentation URL:

☑ Pre

Priority: 1

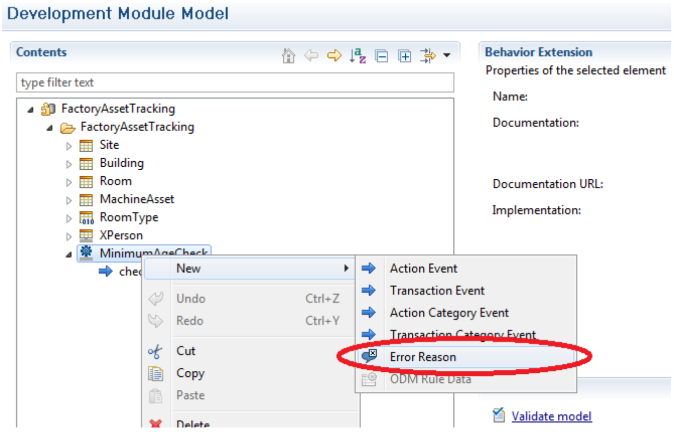
Transaction:



Transaction Selection

addPerson

- Open the context menu on the **MinumumAgeCheck** Behaviour Extension and select **New -> Error Reason**



Development Module Model

Behavior Extension

Properties of the selected element

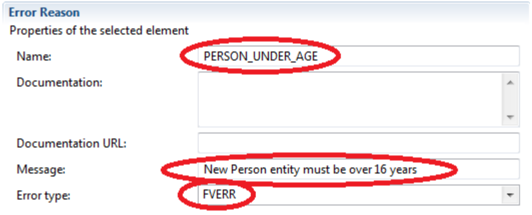
Name:

Documentation:

Documentation URL:

Implementation:

- Select the new Error Reason in the tree view and in the right hand panel set the **Name** to **PERSON_UNDER_AGE**
- Set the **Message** field to "**New Person entity must be over 16 years**"
- Set the **Error type** to **FVERR** (Field Verification Error type)



Error Reason

Properties of the selected element

Name: PERSON_UNDER_AGE

Documentation:

Documentation URL:

Message: New Person entity must be over 16 years

Error type: FVERR

- Repeat this process to create a second Error Reason with these properties:

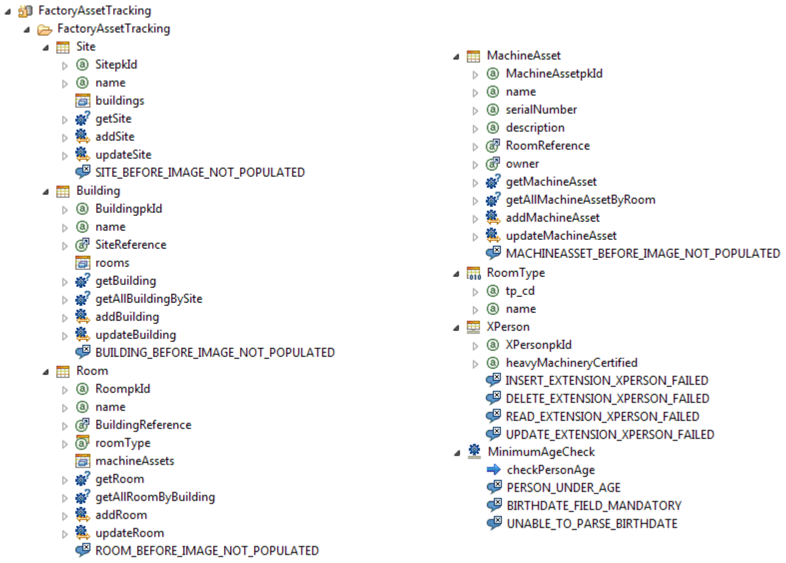
Name - **BIRTHDATE_FIELD_MANDATORY**
Message - **BirthDate field must be provided**

Error type – **FVERR**

- Repeat this process to create a third Error Reason with these properties:

  Name – **UNABLE_TO_PARSE_BIRTHDATE**
  Message - **Unable to parse BirthDate field**
  Error type – **FVERR**

Now that the model is complete, it should look like this:



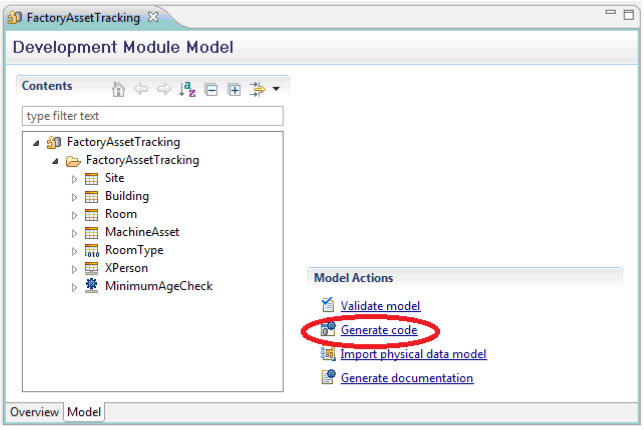
- FactoryAssetTracking
  - FactoryAssetTracking
    - Site
      - SitepkId
      - name
      - buildings
      - getSite
      - addSite
      - updateSite
      - SITE_BEFORE_IMAGE_NOT_POPULATED
    - Building
      - BuildingpkId
      - name
      - SiteReference
      - rooms
      - getBuilding
      - getAllBuildingBySite
      - addBuilding
      - updateBuilding
      - BUILDING_BEFORE_IMAGE_NOT_POPULATED
    - Room
      - RoompkId
      - name
      - BuildingReference
      - roomType
      - machineAssets
      - getRoom
      - getAllRoomByBuilding
      - addRoom
      - updateRoom
      - ROOM_BEFORE_IMAGE_NOT_POPULATED
    - MachineAsset
      - MachineAssetpkId
      - name
      - serialNumber
      - description
      - RoomReference
      - owner
      - getMachineAsset
      - getAllMachineAssetByRoom
      - addMachineAsset
      - updateMachineAsset
      - MACHINEASSET_BEFORE_IMAGE_NOT_POPULATED
    - RoomType
      - tp_cd
      - name
    - XPerson
      - XPersonpkId
      - heavyMachineryCertified
      - INSERT_EXTENSION_XPERSON_FAILED
      - DELETE_EXTENSION_XPERSON_FAILED
      - READ_EXTENSION_XPERSON_FAILED
      - UPDATE_EXTENSION_XPERSON_FAILED
    - MinimumAgeCheck
      - checkPersonAge
      - PERSON_UNDER_AGE
      - BIRTHDATE_FIELD_MANDATORY
      - UNABLE_TO_PARSE_BIRTHDATE

Step 9 - Generating code from the model (KEY STEPS!)

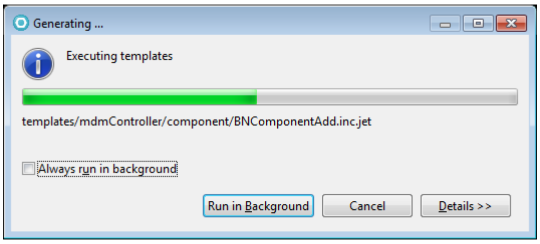
***What's this section about?** Now that the model is complete, we need to create the equivalent code to implement this solution. These steps must be followed to complete the tutorial.*

The bulk of the code will be automatically created by the workbench by running a process referred to as "code generation". Once the code has been generated, some manual extension is required to configure the code table values and implement the behaviour extension logic.

- Click the **Generate code** option at the bottom right hand side of the **Development Module Model panel**



- The **Generating…** dialog shows the progress of code generation.

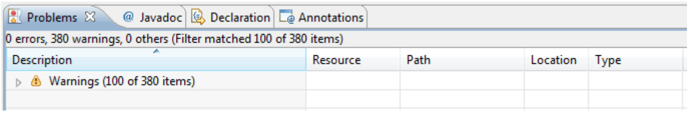


Executing templates

templates/mdmController/component/BNComponentAdd.inc.jet

□ Always run in background

Tip: Once code generation is complete you should check to see if there are any errors in the ***Problems*** *view. If there are problems this indicates that something has failed during code generation and you will need to investigate further.*



0 errors, 380 warnings, 0 others (Filter matched 100 of 380 items)

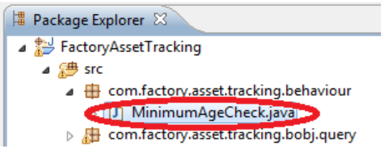
Description	Resource	Path	Location	Type
Warnings (100 of 380 items)				

Step 10 - Implementing the behaviour extension (KEY STEPS!)

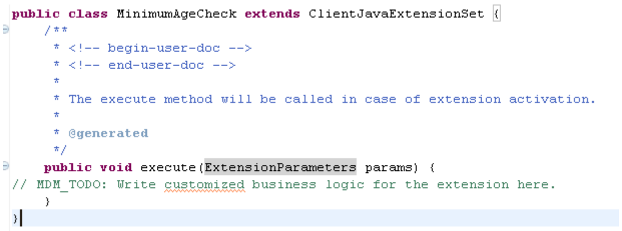
Whats this section about? *We add the Java code required to implement the behaviour extension. You must follow these steps to complete the tutorial.*

As discussed above, we need to add a behaviour extension to ensure that no-one under 16 years of age can be added to the system using the “addPerson” transaction. Code generation will have created a template class that must be manually modified to implement the required logic.

- Open the generated source code file **MinimumAgeCheck.java** from the **FactoryAssetTracking/src/com.factory.asset.tracking.behaviour folder**


Package Explorer
FactoryAssetTracking
src
com.factory.asset.tracking.behaviour
MinimumAgeCheck.java
com.factory.asset.tracking.bobj.query

Tip: This file contains a blank execute method that will be triggered each time the addPerson transaction is invoked. We will need to add code to this method to make the necessary age checks and reject any transactions where the party is less than 16 years old.



```
public class MinimumAgeCheck extends ClientJavaExtensionSet {
    /**
     * <!-- begin-user-doc -->
     * <!-- end-user-doc -->
     *
     * The execute method will be called in case of extension activation.
     *
     * @generated
     */
    public void execute(ExtensionParameters params) {
// MDM_TODO: Write customized business logic for the extension here.
    }
}
```

- Replace the **MinimumAgeCheck.java** file in the workspace with the version supplied with this tutorial.

Tip: This replacement code extracts the root business object from the request (an instance of TCRMPersonBObj) obtains the birth date from it and uses this to evaluate if the person is older than 16 years. If the birth date field is not provided, can't be parsed, or was less than 16 years ago from the current time an error is reported and the transaction is aborted.

Step 11 - Customizing the Code Table values (KEY STEPS!)

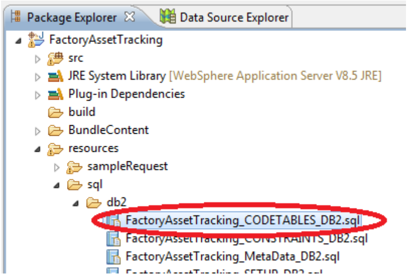
What's this section about? *We configure the possible values for the **RoomType** Code Table. You must follow these steps to complete the tutorial.*

As previously discussed, the **RoomType** provides a mechanism for defining a set of possible room type strings. In this example, we will use the following set of room types:

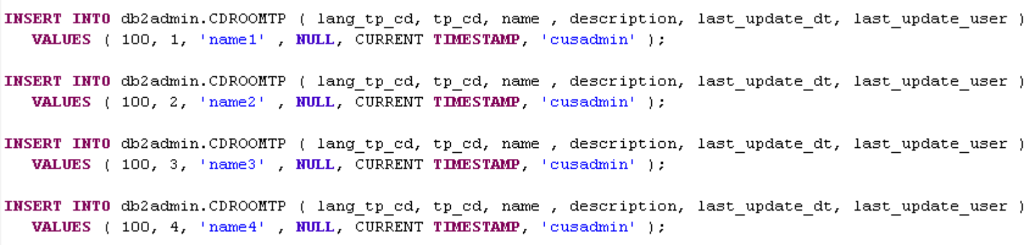
- Office
- PlantRoom
- MeetingRoom
- StoreRoom

Tip: These String values are NOT stored in the Development model. The generated SQL must be manually modified to add the required values.

When code is generated for the model, an SQL file named **FactoryAssetTracking_CODETABLES_DB2.sql** is created in the workspace under **FactoryAssetTracking\resources\sql\db2.**

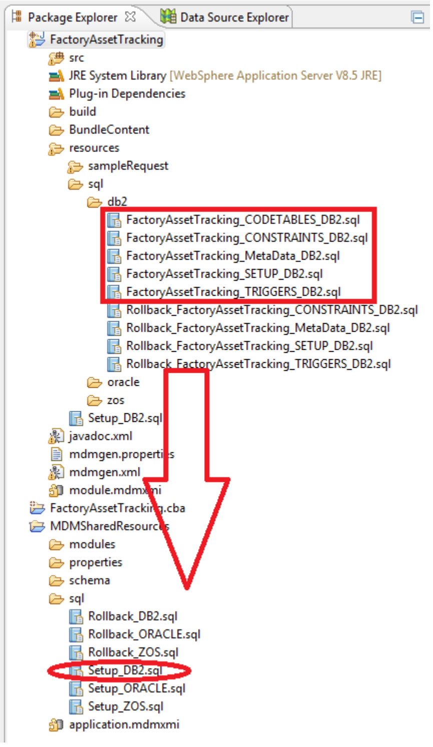

Package Explorer
Data Source Explorer
FactoryAssetTracking
src
JRE System Library [WebSphere Application Server V8.5 JRE]
Plug-in Dependencies
build
BundleContent
resources
sampleRequest
sql
db2
FactoryAssetTracking_CODETABLES_DB2.sql
FactoryAssetTracking_CONSTRAINTS_DB2.sql
FactoryAssetTracking_MetaData_DB2.sql

This file contains INSERT statements to a add 4 sample values to the CDROOMTP database table as shown below:



```
INSERT INTO db2admin.CDROOMTP ( lang_tp_cd, tp_cd, name , description, last_update_dt, last_update_user )
   VALUES ( 100, 1, 'name1' , NULL, CURRENT TIMESTAMP, 'cusadmin' );

INSERT INTO db2admin.CDROOMTP ( lang_tp_cd, tp_cd, name , description, last_update_dt, last_update_user )
   VALUES ( 100, 2, 'name2' , NULL, CURRENT TIMESTAMP, 'cusadmin' );

INSERT INTO db2admin.CDROOMTP ( lang_tp_cd, tp_cd, name , description, last_update_dt, last_update_user )
   VALUES ( 100, 3, 'name3' , NULL, CURRENT TIMESTAMP, 'cusadmin' );

INSERT INTO db2admin.CDROOMTP ( lang_tp_cd, tp_cd, name , description, last_update_dt, last_update_user )
   VALUES ( 100, 4, 'name4' , NULL, CURRENT TIMESTAMP, 'cusadmin' );
```

This SQL setup file, along with the others in the same folder are combined together to create a single SQL setup file named **Setup_DB2.sql** in the **MDMSharedResources\sql\db2** folder as shown below. This process is dynamic and any changes to the SQL files in the **FactoryAssetTracking** project are automatically incorporated into the single merged file.


Package Explorer
Data Source Explorer
FactoryAssetTracking
src
JRE System Library [WebSphere Application Server V8.5 JRE]
Plug-in Dependencies
build
BundleContent
resources
sampleRequest
sql
db2
FactoryAssetTracking_CODETABLES_DB2.sql
FactoryAssetTracking_CONSTRAINTS_DB2.sql
FactoryAssetTracking_MetaData_DB2.sql
FactoryAssetTracking_SETUP_DB2.sql
FactoryAssetTracking_TRIGGERS_DB2.sql
Rollback_FactoryAssetTracking_CONSTRAINTS_DB2.sql
Rollback_FactoryAssetTracking_MetaData_DB2.sql
Rollback_FactoryAssetTracking_SETUP_DB2.sql
Rollback_FactoryAssetTracking_TRIGGERS_DB2.sql
oracle
zos
Setup_DB2.sql
javadoc.xml
mdmgen.properties
mdmgen.xml
module.mdmxmi
FactoryAssetTracking.cba
MDMSharedResources
modules
properties
schema
sql
Rollback_DB2.sql
Rollback_ORACLE.sql
Rollback_ZOS.sql
Setup_DB2.sql
Setup_ORACLE.sql
Setup_ZOS.sql
application.mdmxmi

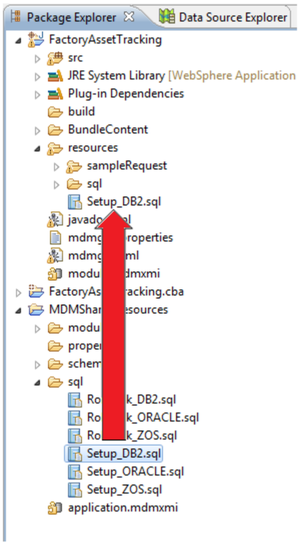
Tip: The Rollback_... scripts are merged in a similar way into the Rollback_DB2.sql file

Tip: Merged script files are also created for the Oracle and zOS database platforms but they are not used within this tutorial.

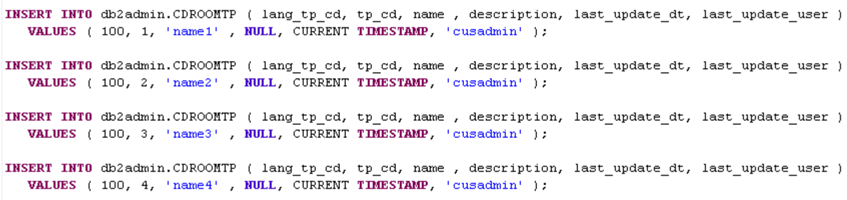
Tip: In this case FactoryAssetTracking is the only development project within the workspace. If you create other development projects and generate code, then the SQL files in those projects are also incorporated within the merged master SQL files.

In order that we can modify the values for the Code Table without them being overwritten during code generation we will modify a copy of the master SQL files. You will need to manually maintain this file if further changes are made to the model so that the SQL files in the MDMSharedResources folder are changed.

- Take a copy of the **Setup_DB2.sql** script from the **MDMSharedResourcessql** folder and place it into the **FactoryAssetTracking\resources** folder

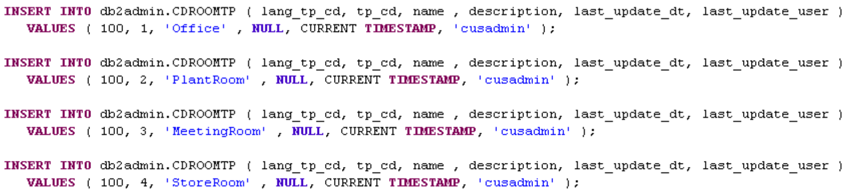


- Open the copied version of the **Setup_DB2.sql** file, scroll to the end and locate the configuration statements for the Code Table as shown below:



```
INSERT INTO db2admin.CDROOMTP ( lang_tp_cd, tp_cd, name , description, last_update_dt, last_update_user )
   VALUES ( 100, 1, 'name1' , NULL, CURRENT TIMESTAMP, 'cusadmin' );

INSERT INTO db2admin.CDROOMTP ( lang_tp_cd, tp_cd, name , description, last_update_dt, last_update_user )
   VALUES ( 100, 2, 'name2' , NULL, CURRENT TIMESTAMP, 'cusadmin' );

INSERT INTO db2admin.CDROOMTP ( lang_tp_cd, tp_cd, name , description, last_update_dt, last_update_user )
   VALUES ( 100, 3, 'name3' , NULL, CURRENT TIMESTAMP, 'cusadmin' );

INSERT INTO db2admin.CDROOMTP ( lang_tp_cd, tp_cd, name , description, last_update_dt, last_update_user )
   VALUES ( 100, 4, 'name4' , NULL, CURRENT TIMESTAMP, 'cusadmin' );
```

- Replace the **name1**, **name2**, **name3** and **name4** values with the required **RoomType** values as shown below:

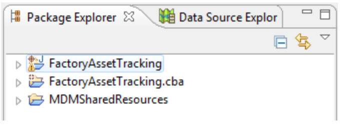


```
INSERT INTO db2admin.CDROOMTP ( lang_tp_cd, tp_cd, name , description, last_update_dt, last_update_user )
   VALUES ( 100, 1, 'Office' , NULL, CURRENT TIMESTAMP, 'cusadmin' );

INSERT INTO db2admin.CDROOMTP ( lang_tp_cd, tp_cd, name , description, last_update_dt, last_update_user )
   VALUES ( 100, 2, 'PlantRoom' , NULL, CURRENT TIMESTAMP, 'cusadmin' );

INSERT INTO db2admin.CDROOMTP ( lang_tp_cd, tp_cd, name , description, last_update_dt, last_update_user )
   VALUES ( 100, 3, 'MeetingRoom' , NULL, CURRENT TIMESTAMP, 'cusadmin' );

INSERT INTO db2admin.CDROOMTP ( lang_tp_cd, tp_cd, name , description, last_update_dt, last_update_user )
   VALUES ( 100, 4, 'StoreRoom' , NULL, CURRENT TIMESTAMP, 'cusadmin' );
```

Step 12 - Deploying the code (KEY STEPS!)

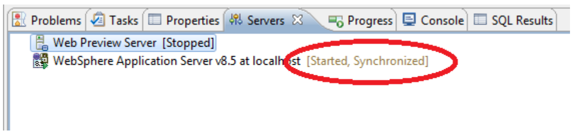
What's this section about? *- Code developed in the workspace will be deployed to the WebSphere application server. You must follow these steps to complete the tutorial.*

Now that the development is complete, the workspace should contain the following projects with code generated ready for deployment:



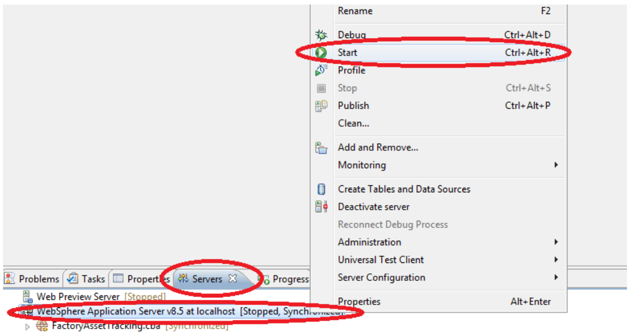
The first step is to deploy the Composite Bundle Archive '**FactoryAssetTracking.cba**' to WebSphere:

- Open the **Servers** view and locate the target WAS server (eg **WebSphere Application Server v8.5 at localhost**). Check to see if the server is **Started**

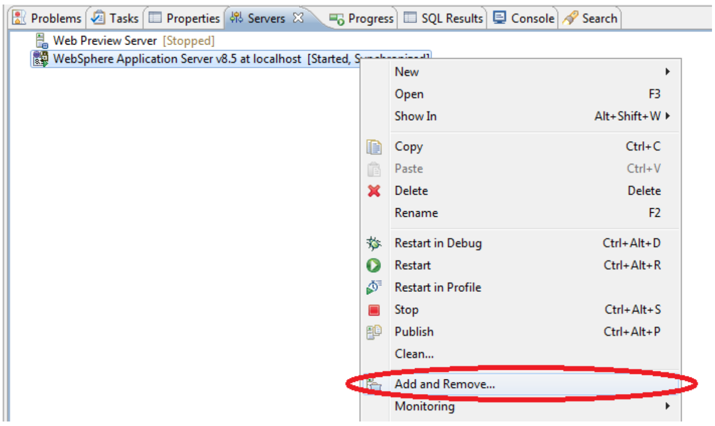


Problems
Tasks
Properties
Servers
Progress
Console
SQL Results
Web Preview Server [Stopped]
WebSphere Application Server v8.5 at localhost [Started, Synchronized]

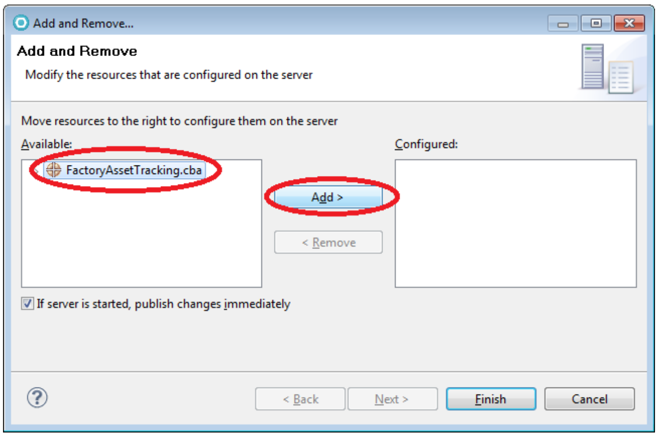
- If the server is Stopped, start it by opening the context menu on the server definition and selecting the **Start** option. Wait for the server start-up to complete.



- Open the context menu for the server definition and select **Add and Remove…**



- In the **Add and Remove…** dialog select the CBA to be deployed from the **Available** list and click **Add >** to move it to the **Configured** list



Add and Remove...

Add and Remove

Modify the resources that are configured on the server

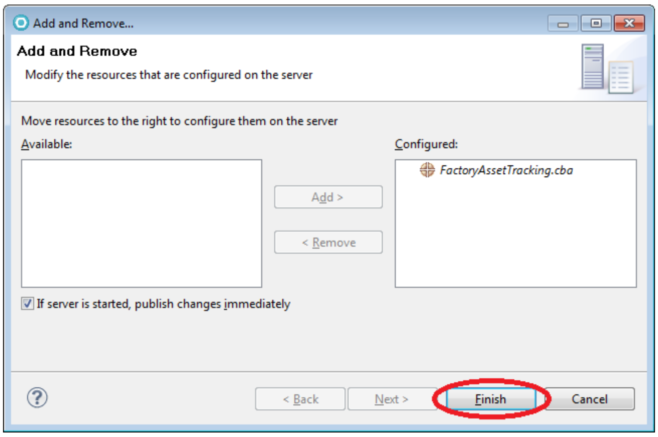
Move resources to the right to configure them on the server

Available: FactoryAssetTracking.cba

Configured:

☑ If server is started, publish changes immediately

- Click the **Finish** button to publish the CBA to the server.



Add and Remove...

Add and Remove

Modify the resources that are configured on the server

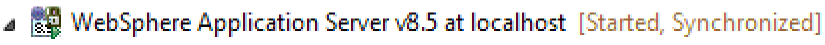
Move resources to the right to configure them on the server

Available:

Configured: *FactoryAssetTracking.cba*

☑ If server is started, publish changes immediately

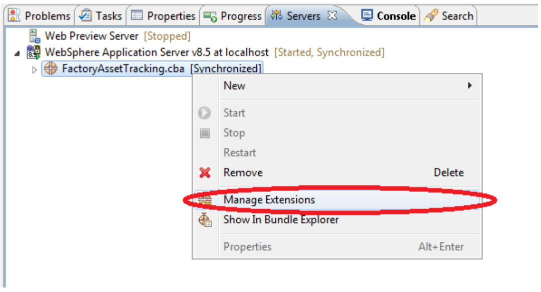
- Wait for the Server connection to become **Synchronized** again



WebSphere Application Server v8.5 at localhost [Started, Synchronized]

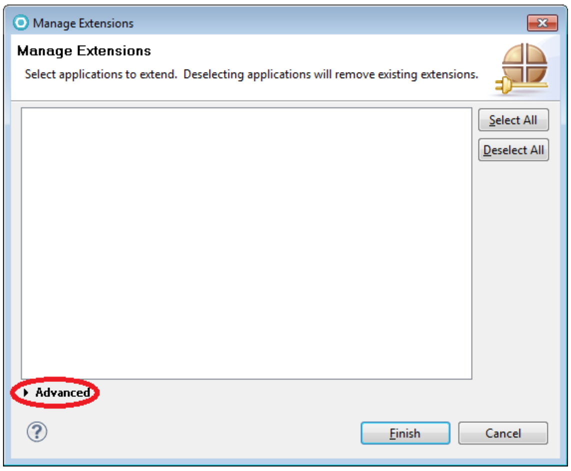
Tip: The CBA is published to the server but not yet associated with a Business Level Application. Before it can be used, it must be configured as a Composition Unit Extension…

- In the **Servers** view, expand the server connection to reveal the published CBA. Open the context menu for the CBA and select **Manage Extensions.**



Web Preview Server [Stopped]
WebSphere Application Server v8.5 at localhost [Started, Synchronized]
FactoryAssetTracking.cba [Synchronized]

- In the **Manage Extensions** dialog, expand the **Advanced** section.

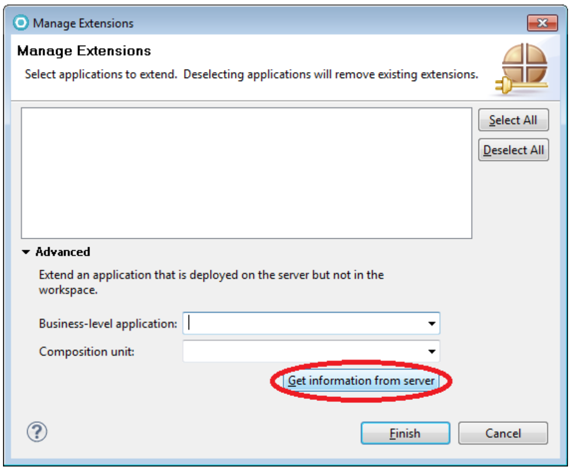


Manage Extensions

Manage Extensions

Select applications to extend. Deselecting applications will remove existing extensions.

- Click the **Get information from server** button to retrieve information about existing deployed applications.



Manage Extensions

Manage Extensions

Select applications to extend. Deselecting applications will remove existing extensions.

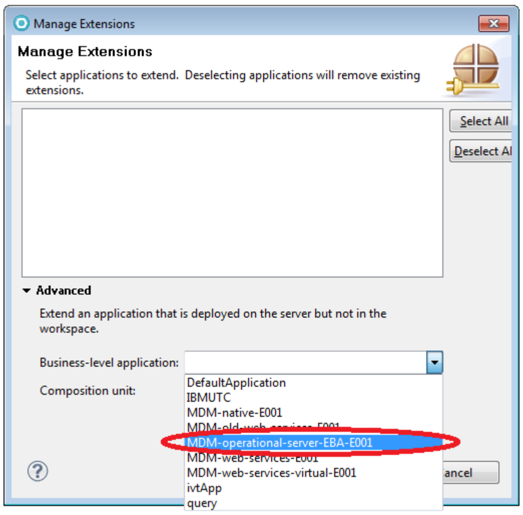
Advanced

Extend an application that is deployed on the server but not in the workspace.

Business-level application:

Composition unit:

- When the **Business-level application** dropdown appears, select the target application (in the case of MDM Server this is **MDM-operation-server-EBA-E001).**



Manage Extensions

Manage Extensions

Select applications to extend. Deselecting applications will remove existing extensions.

Advanced

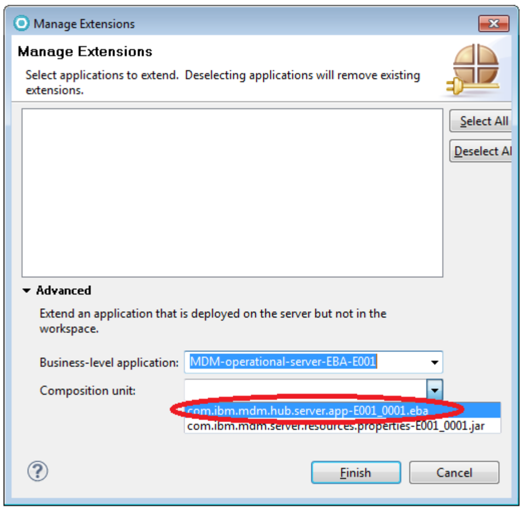
Extend an application that is deployed on the server but not in the workspace.

Business-level application:

Composition unit:

DefaultApplication
IBMUTC
MDM-native-E001
MDM-old-web-services-E001
MDM-operational-server-EBA-E001
MDM-web-services-E001
MDM-web-services-virtual-E001
ivtApp
query

- When the **Composition unit** dropdown appears select the target Composition Unit (in the case of MDM Server this is **com.ibm.mdm.hub.server.app-E001_0001.eba**)



Manage Extensions

Manage Extensions

Select applications to extend. Deselecting applications will remove existing extensions.

Advanced

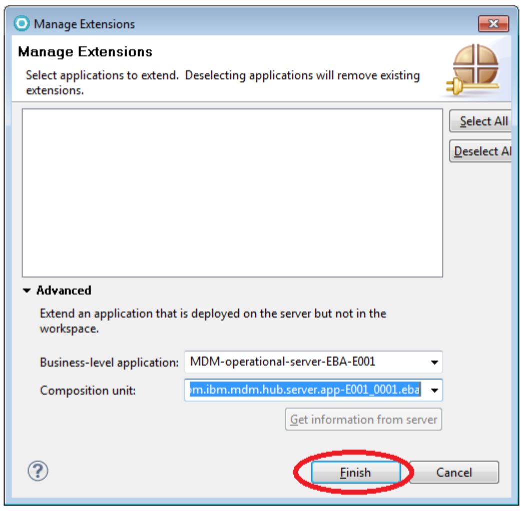
Extend an application that is deployed on the server but not in the workspace.

Business-level application: MDM-operational-server-EBA-E001

Composition unit:

com.ibm.mdm.hub.server.app-E001_0001.eba
com.ibm.mdm.server.resources.properties-E001_0001.jar

- Click **Finish** to apply this configuration



Manage Extensions

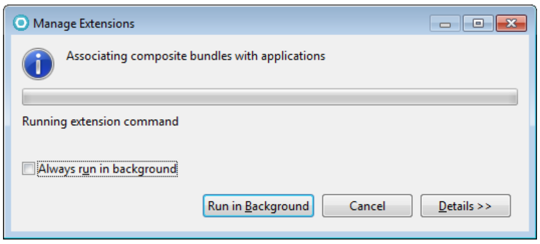
Select applications to extend. Deselecting applications will remove existing extensions.

Extend an application that is deployed on the server but not in the workspace.

Business-level application: MDM-operational-server-EBA-E001

Composition unit: m.ibm.mdm.hub.server.app-E001_0001.eba

- The **Manage Extension** dialog will appear whilst this configuration is performed.

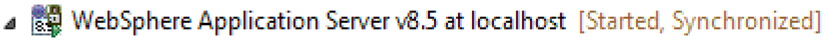


Associating composite bundles with applications

Running extension command

☐ Always run in background

- Once the process is complete, check the Server connection status again and wait for it to become **Synchronized**



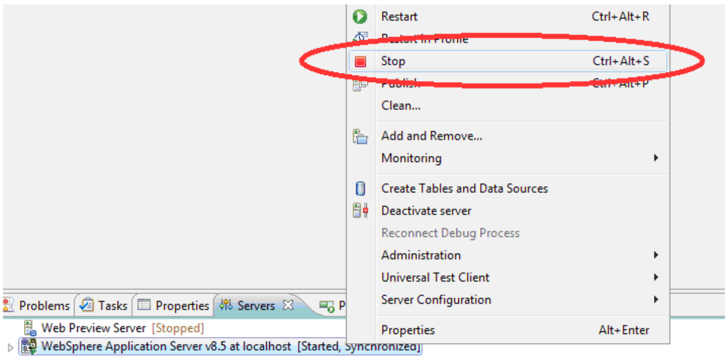
WebSphere Application Server v8.5 at localhost [Started, Synchronized]

Tip: Although the CBA is now fully deployed, the Database configuration must be applied before it can be invoked. The next step is to setup a connection to the MDM database and run an SQL script to apply the necessary configuration.

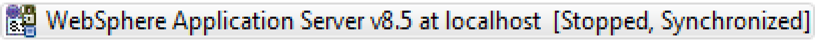
Step 13 - Deploying the metadata configuration (KEY STEPS!)

What's this section about? *- Configuration settings will be deployed to the DB2 database. You must follow these steps to complete the tutorial.*

- In the **Servers** view, open the context menu on the server connection and select **Stop**.

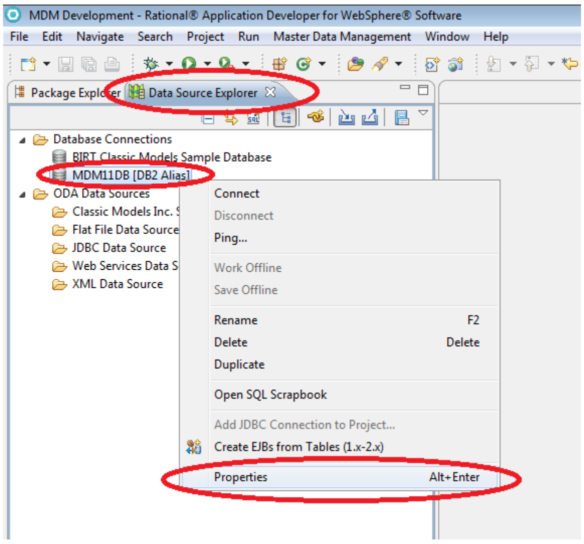


- Wait for the Server connection to become **Stopped**



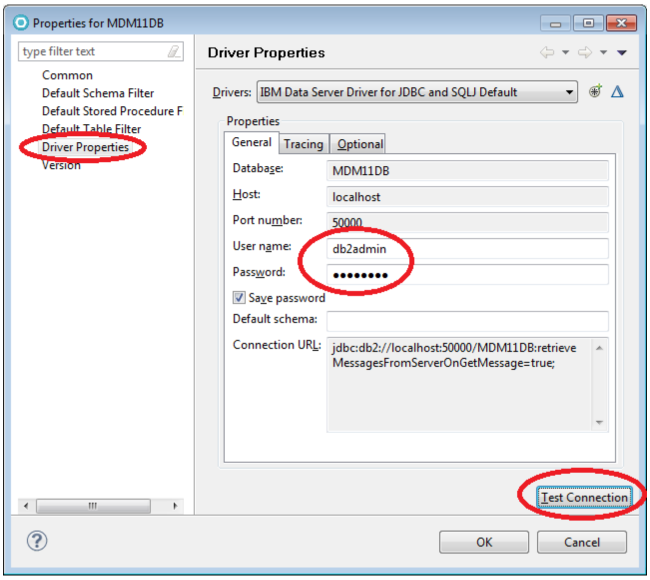
WebSphere Application Server v8.5 at localhost [Stopped, Synchronized]

- Select the **Data Source Explorer** view and expand **Database Connections**
- Open the context menu on the MDM database connection (**MDM11DB**) and select **Properties**



- In the **Properties for MDM11DB** dialog, Select **Driver Properties** in the left hand panel, open the **General** tab in the right hand panel and update the **User name** and **Password** fields if necessary. The default values for the MDM Typical Workbench installation are:

DB2 User name : db2admin
DB2 Password: db3Admin



Properties for MDM11DB

Driver Properties

Drivers: IBM Data Server Driver for JDBC and SQLJ Default

Database: MDM11DB

Host: localhost

Port number: 50000

User name: db2admin

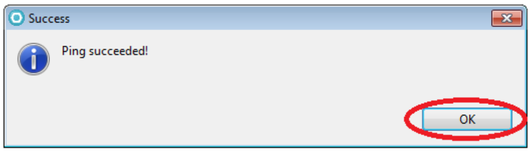
Password: ••••••••

☑ Save password

Default schema:

Connection URL: jdbc:db2://localhost:50000/MDM11DB:retrieveMessagesFromServerOnGetMessage=true;

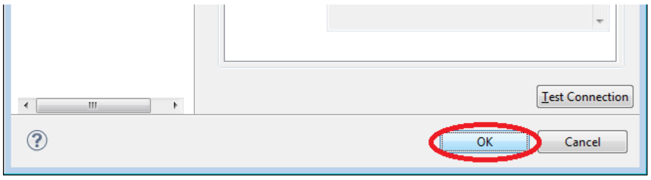
- Click the **Test Connection** button to determine if these settings are correct. You should see the **Success** dialog after a few seconds. Click **OK** to dismiss this window.



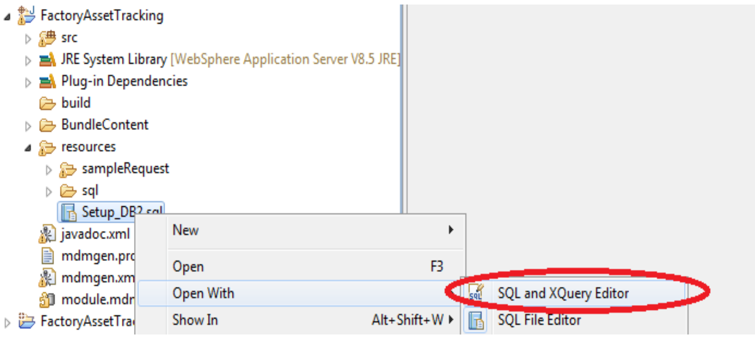
Success

Ping succeeded!

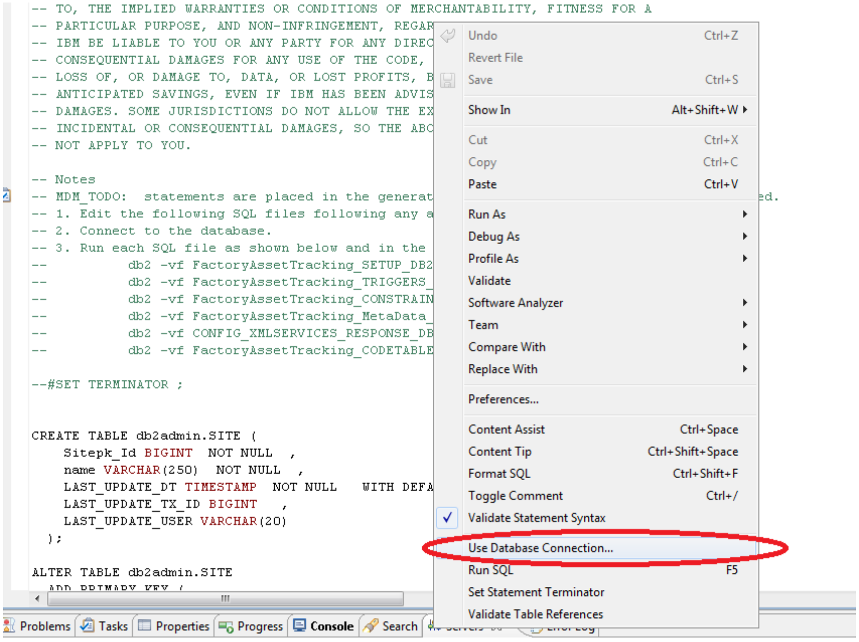
- Once you have confirmed the connection to the database is correctly setup, close the **Properties for MDM11DB** dialog by clicking **OK**



- Select the **Package Explorer** view
- Locate the **Setup_DB2.sql** script that you modified in step 11.
- Open the context menu over this file and select **Open With -> SQL and XQuery Editor**

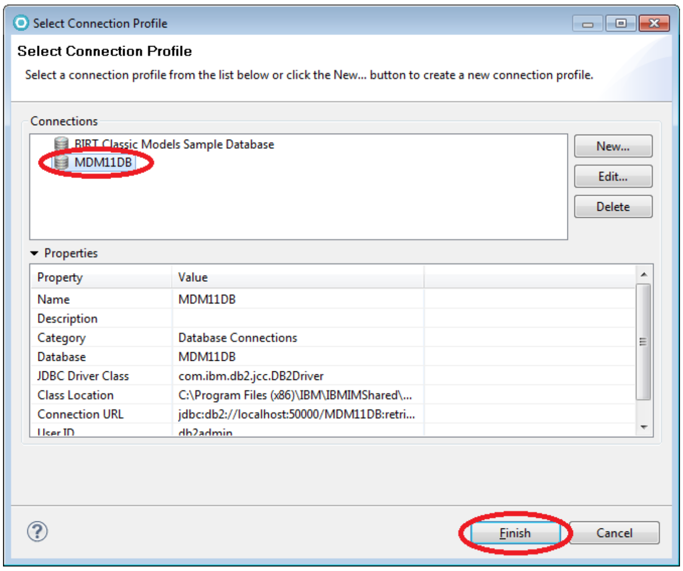


- Open the context menu anywhere in the SQL file and select **Use Database Connection…**



```
-- TO, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A
-- PARTICULAR PURPOSE, AND NON-INFRINGEMENT, REGAR
-- IBM BE LIABLE TO YOU OR ANY PARTY FOR ANY DIREC
-- CONSEQUENTIAL DAMAGES FOR ANY USE OF THE CODE,
-- LOSS OF, OR DAMAGE TO, DATA, OR LOST PROFITS, B
-- ANTICIPATED SAVINGS, EVEN IF IBM HAS BEEN ADVIS
-- DAMAGES. SOME JURISDICTIONS DO NOT ALLOW THE EX
-- INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABC
-- NOT APPLY TO YOU.

-- Notes
-- MDM_TODO:  statements are placed in the generat
-- 1. Edit the following SQL files following any a
-- 2. Connect to the database.
-- 3. Run each SQL file as shown below and in the
--          db2 -vf FactoryAssetTracking_SETUP_DB2
--          db2 -vf FactoryAssetTracking_TRIGGERS_
--          db2 -vf FactoryAssetTracking_CONSTRAIN
--          db2 -vf FactoryAssetTracking_MetaData_
--          db2 -vf CONFIG_XMLSERVICES_RESPONSE_DB
--          db2 -vf FactoryAssetTracking_CODETABLE

--#SET TERMINATOR ;


CREATE TABLE db2admin.SITE (
    Sitepk_Id BIGINT  NOT NULL  ,
    name VARCHAR(250)  NOT NULL  ,
    LAST_UPDATE_DT TIMESTAMP  NOT NULL   WITH DEFA
    LAST_UPDATE_TX_ID BIGINT   ,
    LAST_UPDATE_USER VARCHAR(20)
  );

ALTER TABLE db2admin.SITE
```

- In the **Select Connection Profile** dialog, select the **MDM11DB** database connection and click **Finish**



Select Connection Profile

Select a connection profile from the list below or click the New... button to create a new connection profile.

Connections

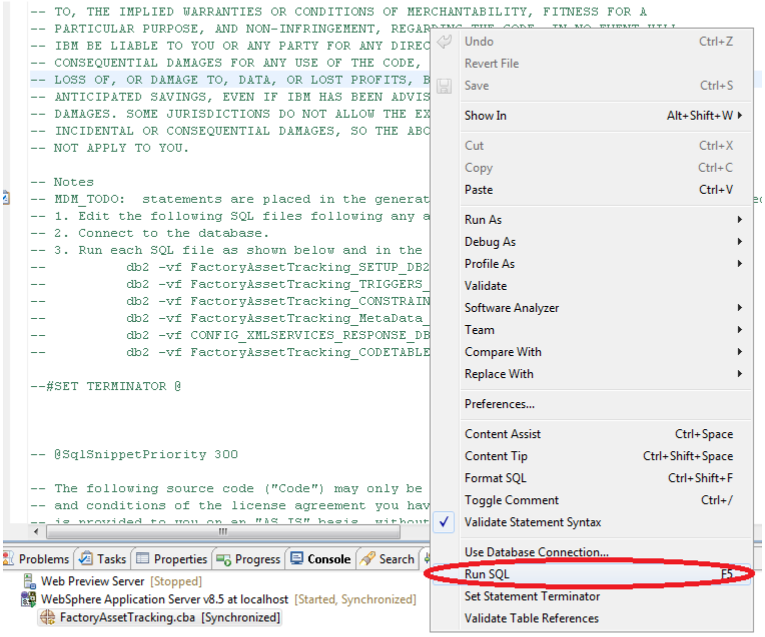
BIRT Classic Models Sample Database
MDM11DB

Properties

Property	Value
Name	MDM11DB
Description	
Category	Database Connections
Database	MDM11DB
JDBC Driver Class	com.ibm.db2.jcc.DB2Driver
Class Location	C:\Program Files (x86)\IBM\IBMIMShared\...
Connection URL	jdbc:db2://localhost:50000/MDM11DB:retri...
User ID	db2admin

Tip: This associates the specific SQL file with the MDM11DB connection. When the script is subsequently executed, this is the database connection that will be used.

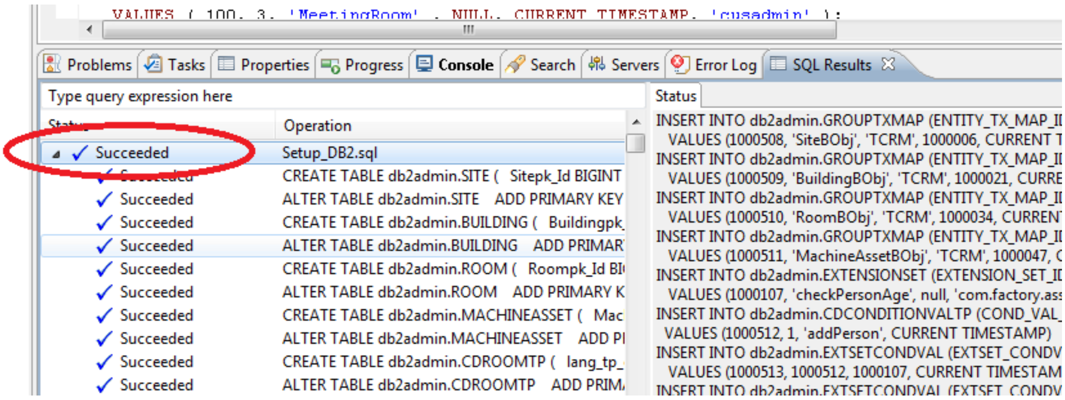
- Open the context menu anywhere in the SQL file and select **Run SQL**



```
-- TO, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A
-- PARTICULAR PURPOSE, AND NON-INFRINGEMENT, REGAR
-- IBM BE LIABLE TO YOU OR ANY PARTY FOR ANY DIREC
-- CONSEQUENTIAL DAMAGES FOR ANY USE OF THE CODE,
-- LOSS OF, OR DAMAGE TO, DATA, OR LOST PROFITS, B
-- ANTICIPATED SAVINGS, EVEN IF IBM HAS BEEN ADVIS
-- DAMAGES. SOME JURISDICTIONS DO NOT ALLOW THE EX
-- INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABC
-- NOT APPLY TO YOU.

-- Notes
-- MDM_TODO:  statements are placed in the generat
-- 1. Edit the following SQL files following any a
-- 2. Connect to the database.
-- 3. Run each SQL file as shown below and in the
--          db2 -vf FactoryAssetTracking_SETUP_DB2
--          db2 -vf FactoryAssetTracking_TRIGGERS_
--          db2 -vf FactoryAssetTracking_CONSTRAIN
--          db2 -vf FactoryAssetTracking_MetaData_
--          db2 -vf CONFIG_XMLSERVICES_RESPONSE_DB
--          db2 -vf FactoryAssetTracking_CODETABLE

--#SET TERMINATOR @



-- @SqlSnippetPriority 300

-- The following source code ("Code") may only be
-- and conditions of the license agreement you hav
```

Web Preview Server [Stopped]
WebSphere Application Server v8.5 at localhost [Started, Synchronized]
FactoryAssetTracking.cba [Synchronized]

- The statements in the SQL file will be executed to configure the MDM database. Progress is displayed in the **SQL Results** view that will open in the workbench.



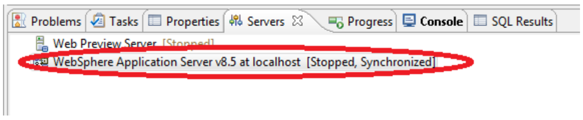
Status	Operation
✓ Succeeded	Setup_DB2.sql
✓ Succeeded	CREATE TABLE db2admin.SITE (Sitepk_Id BIGINT
✓ Succeeded	ALTER TABLE db2admin.SITE ADD PRIMARY KEY
✓ Succeeded	CREATE TABLE db2admin.BUILDING (Buildingpk_
✓ Succeeded	ALTER TABLE db2admin.BUILDING ADD PRIMAR
✓ Succeeded	CREATE TABLE db2admin.ROOM (Roompk_Id BI
✓ Succeeded	ALTER TABLE db2admin.ROOM ADD PRIMARY K
✓ Succeeded	CREATE TABLE db2admin.MACHINEASSET (Mac
✓ Succeeded	ALTER TABLE db2admin.MACHINEASSET ADD PI
✓ Succeeded	CREATE TABLE db2admin.CDROOMTP (lang_tp_
✓ Succeeded	ALTER TABLE db2admin.CDROOMTP ADD PRIM.

```
Status
INSERT INTO db2admin.GROUPTXMAP (ENTITY_TX_MAP_ID
  VALUES (1000508, 'SiteBObj', 'TCRM', 1000006, CURRENT T
INSERT INTO db2admin.GROUPTXMAP (ENTITY_TX_MAP_ID
  VALUES (1000509, 'BuildingBObj', 'TCRM', 1000021, CURRE
INSERT INTO db2admin.GROUPTXMAP (ENTITY_TX_MAP_ID
  VALUES (1000510, 'RoomBObj', 'TCRM', 1000034, CURREN
INSERT INTO db2admin.GROUPTXMAP (ENTITY_TX_MAP_ID
  VALUES (1000511, 'MachineAssetBObj', 'TCRM', 1000047, C
INSERT INTO db2admin.EXTENSIONSET (EXTENSION_SET_ID
  VALUES (1000107, 'checkPersonAge', null, 'com.factory.ass
INSERT INTO db2admin.CDCONDITIONVALTP (COND_VAL_
 VALUES (1000512, 1, 'addPerson', CURRENT TIMESTAMP)
INSERT INTO db2admin.EXTSETCONDVAL (EXTSET_CONDV
  VALUES (1000513, 1000512, 1000107, CURRENT TIMESTAM
INSERT INTO db2admin.EXTSETCONDVAL (EXTSET_CONDV
```

Tip: You should check to see if the overall process Succeeded or if individual statements failed. Where failures do occur, it may be that a previous deployment is causing conflicts (e.g. a table already exists). You can either resolve such issues by hand or try to remove a previous configuration using the "Rollback" scripts created alongside the Setup script before re-running the setup scripts.

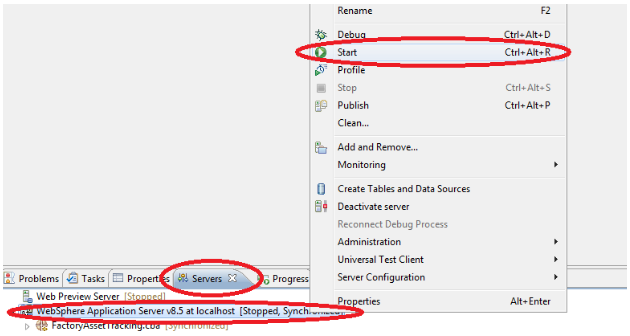
Once the database is configured its time to restart the WAS server:

- Open the **Servers** view and locate the target WAS server (eg **WebSphere Application Server v8.5 at localhost**).



Web Preview Server [Stopped]
WebSphere Application Server v8.5 at localhost [Stopped, Synchronized]

- Open the context menu on the server definition and select **Start**. Wait for the server start-up to complete



Note: The accompanying document "MDMWorkbenchV11Deployment" provides more detail about the development process including:

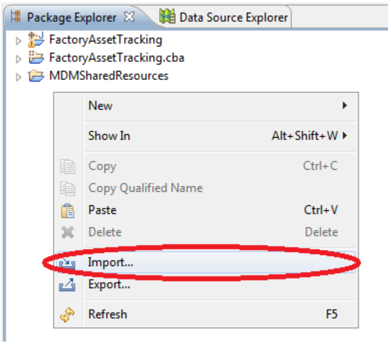
- *How to remove a CBA from the server*
- *How to export the CBA from the workbench and install this via the WebSphere admin console.*

Step 14 - Testing the Solution (KEY STEPS!)

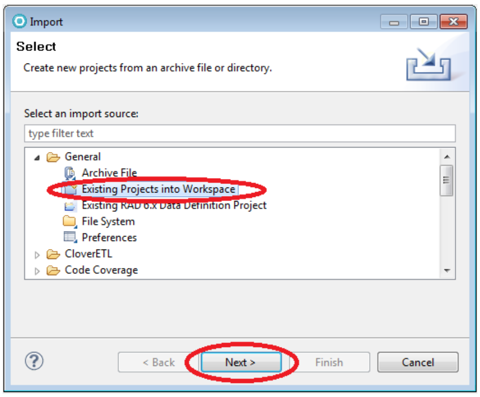
***What's this section about? -** Now that the code is deployed and configured this section shows how the services it provides can be invoked and tested.*

The code can be tested by running a series of transactions against the server to create and retrieve instances of the Entity Types defined by the model. The transaction documents are supplied in an archived project for your convenience.

- Switch to the **Package Explorer** view, open the Context menu and select **Import…**

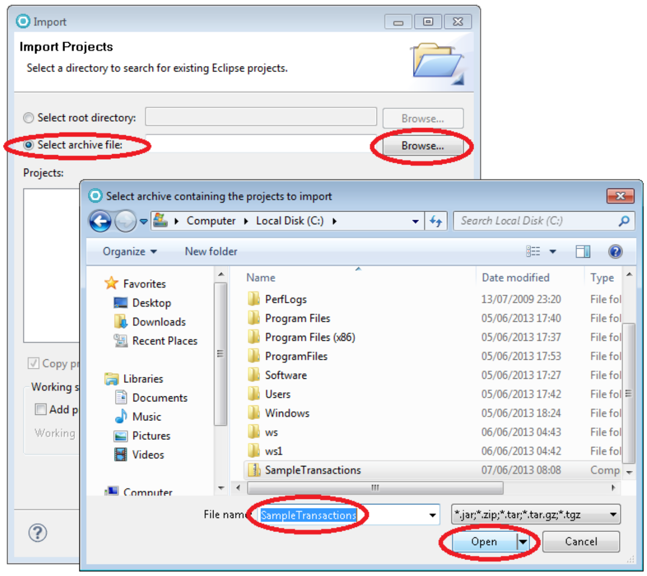


- In the Import dialog, select **General -> Existing Projects into Workspace** and click **Next**



Create new projects from an archive file or directory.

- In the **Import** dialog, Select the **Select archive file** option and click **Browse**…
- In the **Select archive containing the projects to import** dialog, locate the supplied **SampleTransactions.zip** file and click Open



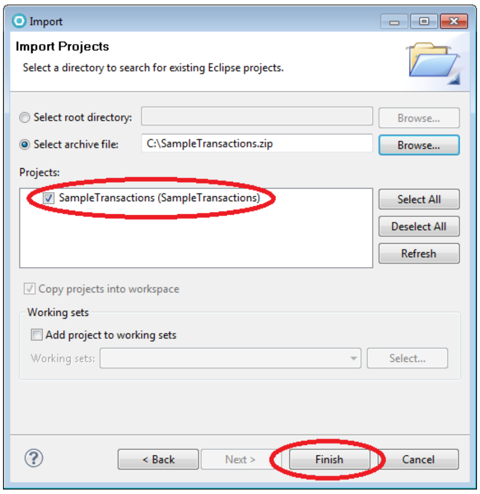
Select a directory to search for existing Eclipse projects.

Name	Date modified	Type
PerfLogs	13/07/2009 23:20	File fol
Program Files	05/06/2013 17:40	File fol
Program Files (x86)	05/06/2013 17:37	File fol
ProgramFiles	05/06/2013 17:53	File fol
Software	05/06/2013 17:27	File fol
Users	05/06/2013 17:42	File fol
Windows	05/06/2013 18:24	File fol
ws	06/06/2013 04:43	File fol
ws1	06/06/2013 04:42	File fol
SampleTransactions	07/06/2013 08:08	Comp

File name: SampleTransactions

.jar;.zip;*.tar;*.tar.gz;*.tgz

- Ensure the **SampleTransactions (SampleTransactions)** option is checked and click **Finish**. This will import the projects containing the sample transactions into the workspace



Import Projects

Select a directory to search for existing Eclipse projects.

○ Select root directory:

● Select archive file: C:\SampleTransactions.zip

Projects:

☑ SampleTransactions (SampleTransactions)

☑ Copy projects into workspace

Working sets

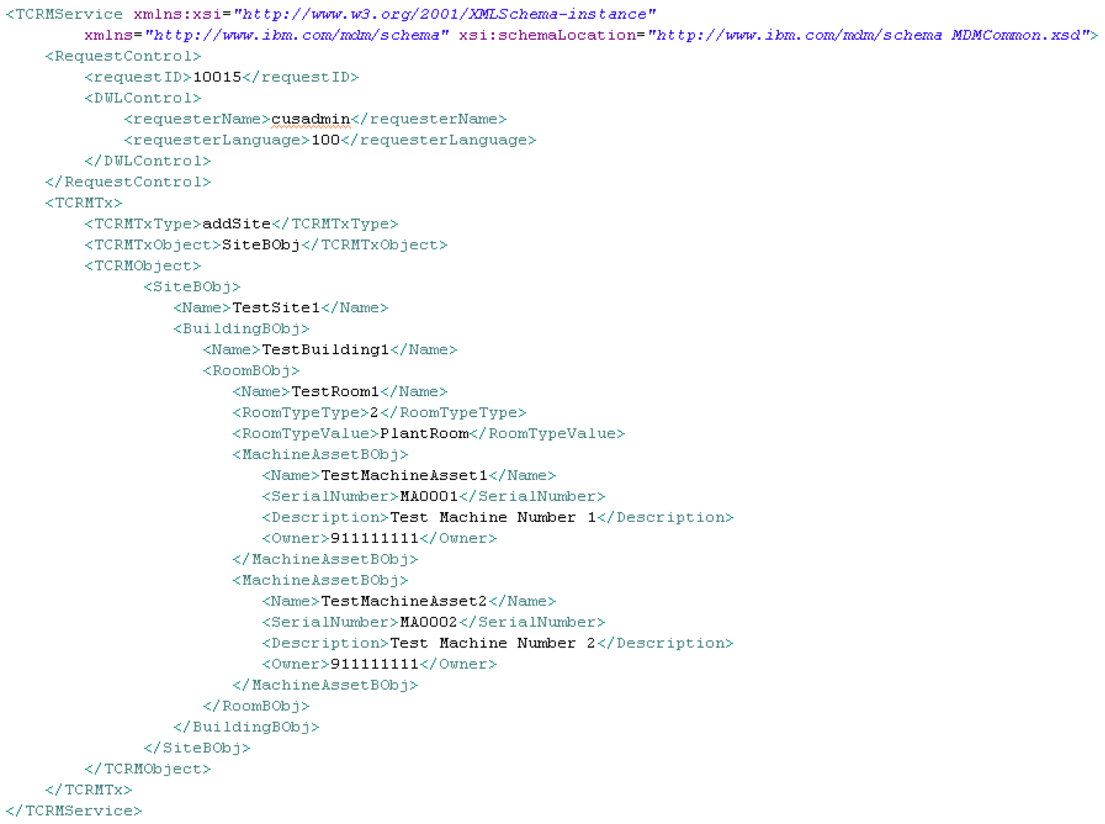
□ Add project to working sets

Working sets:

TEST 1 - Creating a Site with associated content...

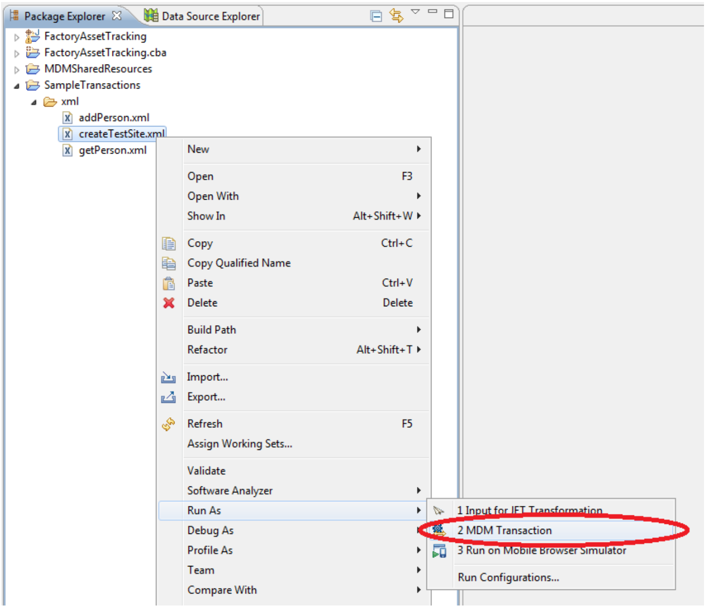
The first XML transaction file we will run is called createTestSite.xml. It will create a new Site instance called “TestSite1”. Contained within the new site will be a single Building instance called “TestBuilding1”. Contained within the new building will be a single Room instance of type PlantRoom called “TestRoom1”. Finally, the Room will have two machine assets listed, TestMachineAsset1 (serial MA0001) and TestMachineAsset2 (serial MA0002). Both of these machine assets are owned by the Person instance created by the MDM Server Installation Verification Test (primary key is 91111111)

Examine the createTestSite.xml document shown below to understand its structure:



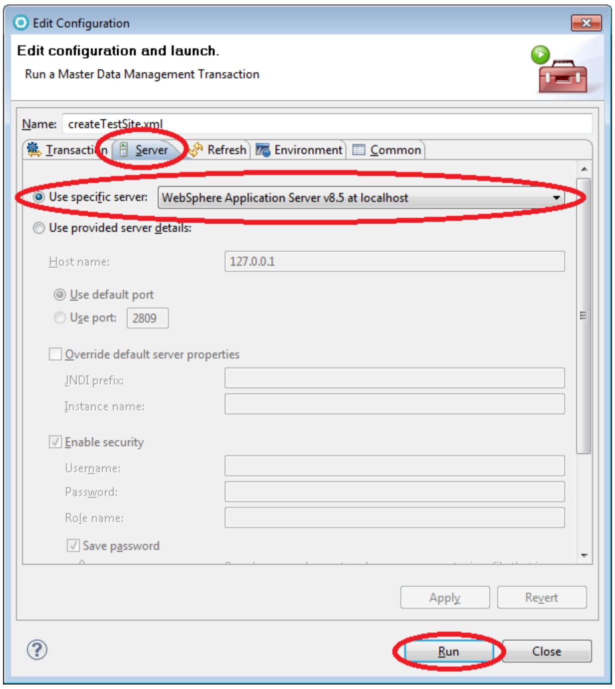
```xml
<TCRMService xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.ibm.com/mdm/schema" xsi:schemaLocation="http://www.ibm.com/mdm/schema MDMCommon.xsd">
    <RequestControl>
        <requestID>10015</requestID>
        <DWLControl>
            <requesterName>cusadmin</requesterName>
            <requesterLanguage>100</requesterLanguage>
        </DWLControl>
    </RequestControl>
    <TCRMTx>
        <TCRMTxType>addSite</TCRMTxType>
        <TCRMTxObject>SiteBObj</TCRMTxObject>
        <TCRMObject>
              <SiteBObj>
                 <Name>TestSite1</Name>
                 <BuildingBObj>
                    <Name>TestBuilding1</Name>
                    <RoomBObj>
                       <Name>TestRoom1</Name>
                       <RoomTypeType>2</RoomTypeType>
                       <RoomTypeValue>PlantRoom</RoomTypeValue>
                       <MachineAssetBObj>
                          <Name>TestMachineAsset1</Name>
                          <SerialNumber>MA0001</SerialNumber>
                          <Description>Test Machine Number 1</Description>
                          <Owner>911111111</Owner>
                       </MachineAssetBObj>
                       <MachineAssetBObj>
                          <Name>TestMachineAsset2</Name>
                          <SerialNumber>MA0002</SerialNumber>
                          <Description>Test Machine Number 2</Description>
                          <Owner>911111111</Owner>
                       </MachineAssetBObj>
                    </RoomBObj>
                 </BuildingBObj>
              </SiteBObj>
        </TCRMObject>
    </TCRMTx>
</TCRMService>
```

- Again in the **Package Explorer** view, open the context menu on **SampleTransactions -> xml -> createTestSite.xml** and select **Run as -> MDM Transaction**



- In the **Edit Configuration** dialog, open the Server tab, enable the **Use specific server** option and select the target WebSphere Application Server.

- Click **Run** to execute the transaction



Edit Configuration

Edit configuration and launch.

Run a Master Data Management Transaction

Name: createTestSite.xml

● Use specific server: WebSphere Application Server v8.5 at localhost

○ Use provided server details:

Host name: 127.0.0.1

● Use default port

○ Use port: 2809

□ Override default server properties

JNDI prefix:

Instance name:

☑ Enable security

Username:

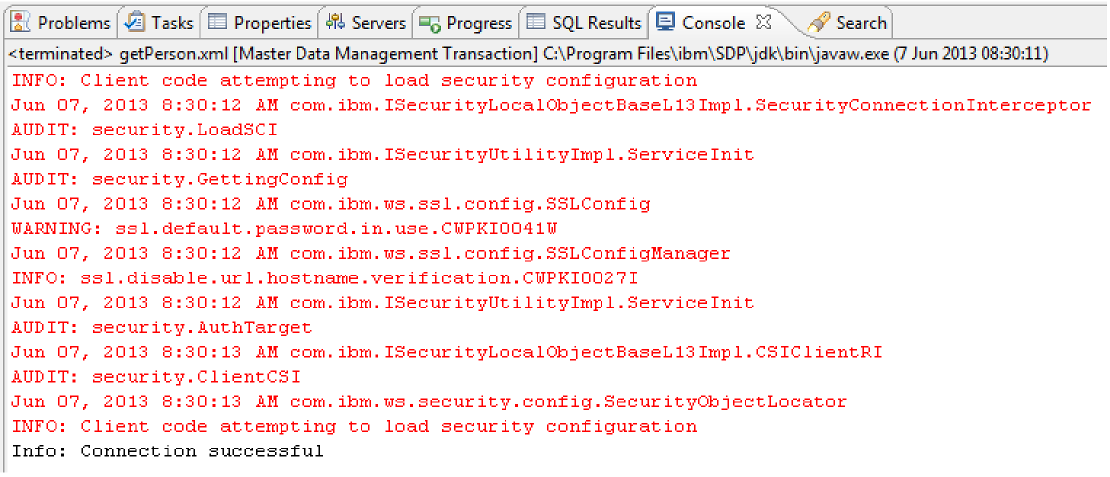
Password:

Role name:

☑ Save password

- Progress is displayed in the **Console** view as shown below. Wait for the **Info: Connection successful** message to be displayed.

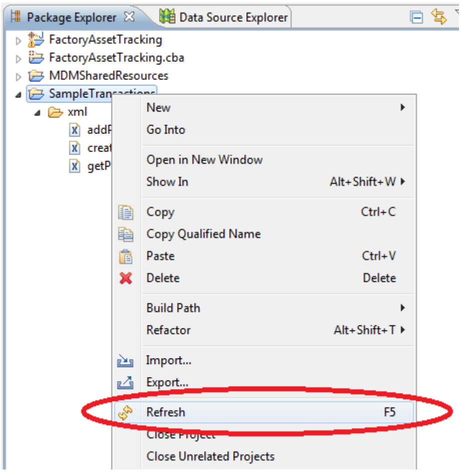
Tip: The first time you run the transaction launcher, it may take longer to complete as client code is loaded and cached. Subsequent transactions will run faster.



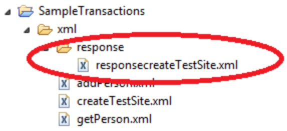
```
<terminated> getPerson.xml [Master Data Management Transaction] C:\Program Files\ibm\SDP\jdk\bin\javaw.exe (7 Jun 2013 08:30:11)
INFO: Client code attempting to load security configuration
Jun 07, 2013 8:30:12 AM com.ibm.ISecurityLocalObjectBaseL13Impl.SecurityConnectionInterceptor
AUDIT: security.LoadSCI
Jun 07, 2013 8:30:12 AM com.ibm.ISecurityUtilityImpl.ServiceInit
AUDIT: security.GettingConfig
Jun 07, 2013 8:30:12 AM com.ibm.ws.ssl.config.SSLConfig
WARNING: ssl.default.password.in.use.CWPKI0041W
Jun 07, 2013 8:30:12 AM com.ibm.ws.ssl.config.SSLConfigManager
INFO: ssl.disable.url.hostname.verification.CWPKI0027I
Jun 07, 2013 8:30:12 AM com.ibm.ISecurityUtilityImpl.ServiceInit
AUDIT: security.AuthTarget
Jun 07, 2013 8:30:13 AM com.ibm.ISecurityLocalObjectBaseL13Impl.CSIClientRI
AUDIT: security.ClientCSI
Jun 07, 2013 8:30:13 AM com.ibm.ws.security.config.SecurityObjectLocator
INFO: Client code attempting to load security configuration
Info: Connection successful
```

Tip: The results of the transaction execution are captured in a file alongside the original request XML file. The file is created on disk and the workspace must be refreshed to detect and show this file.

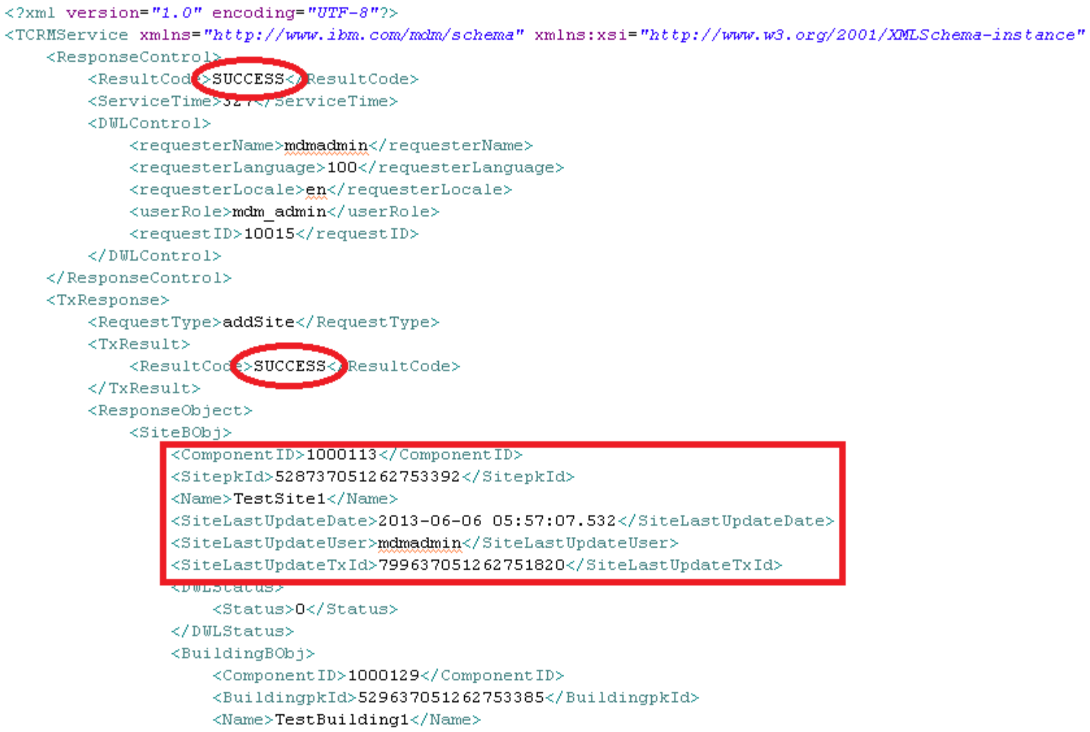
- Open the context menu on the **SampleTransactions** project and select **Refresh**



- Once the project is refreshed, the XML response file will be visible in a **response** folder. Open the **reponsecreateTestSuite.xml** file to see the results.



If the code is working correctly then the response file will contain a **SUCCESS** result code as shown below. In addition, various fields will be populated with data derived from the server (eg the primary key of the new **Site** Entity Type)



```
<?xml version="1.0" encoding="UTF-8"?>
<TCRMService xmlns="http://www.ibm.com/mdm/schema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <ResponseControl>
        <ResultCode>SUCCESS</ResultCode>
        <ServiceTime>327</ServiceTime>
        <DWLControl>
            <requesterName>mdmadmin</requesterName>
            <requesterLanguage>100</requesterLanguage>
            <requesterLocale>en</requesterLocale>
            <userRole>mdm_admin</userRole>
            <requestID>10015</requestID>
        </DWLControl>
    </ResponseControl>
    <TxResponse>
        <RequestType>addSite</RequestType>
        <TxResult>
            <ResultCode>SUCCESS</ResultCode>
        </TxResult>
        <ResponseObject>
            <SiteBObj>
                <ComponentID>1000113</ComponentID>
                <SitepkId>528737051262753392</SitepkId>
                <Name>TestSite1</Name>
                <SiteLastUpdateDate>2013-06-06 05:57:07.532</SiteLastUpdateDate>
                <SiteLastUpdateUser>mdmadmin</SiteLastUpdateUser>
                <SiteLastUpdateTxId>799637051262751820</SiteLastUpdateTxId>
                <DWLStatus>
                    <Status>0</Status>
                </DWLStatus>
                <BuildingBObj>
                    <ComponentID>1000129</ComponentID>
                    <BuildingpkId>529637051262753385</BuildingpkId>
                    <Name>TestBuilding1</Name>
```

SitepkId – Primary key field allocated by MDM Server for the new **Site** instance. This value can be passed to a subsequent **getSite** transaction to retrieve the **Site** instance again.

SiteLastUpdateDate – The date and time of the last update operation (in this case the add operation). This field must be passed to a subsequent updateSite operation to ensure integrity between a getSite and subsequent updateSite. If in independent update occurs between the two transactions then the update will fail as the last update date would no longer be valid.

SiteLastUpdateUser – The user who performed the last update operation

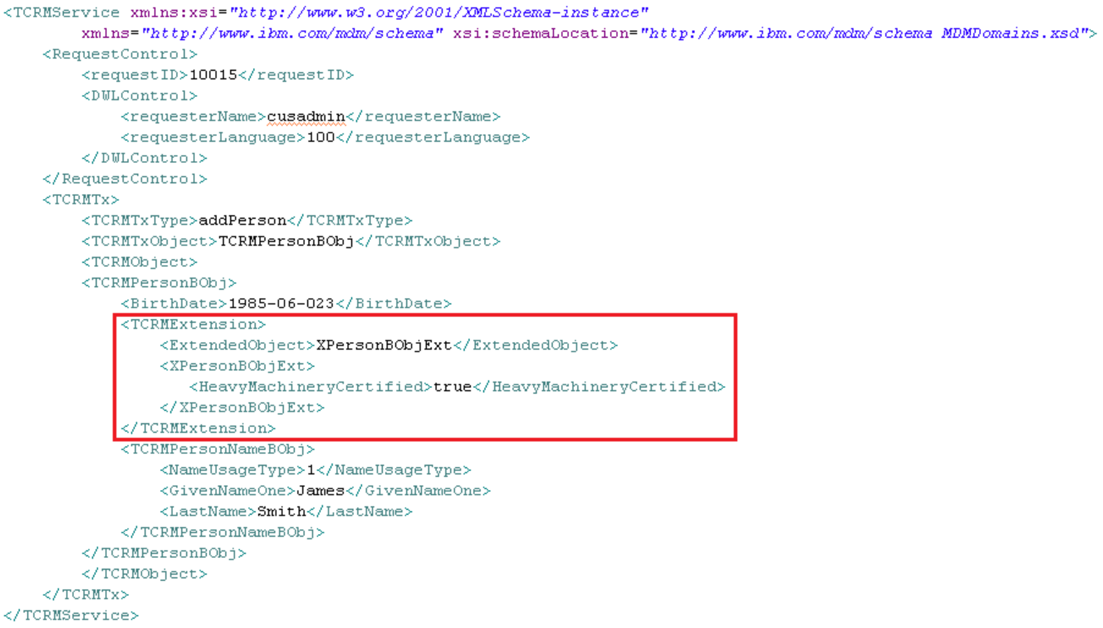
SiteLastUpdateTxID – The transaction id of the last update operation

Test 2 - Creating a new Person Instance With Extended Fields

Next we will test the extended attributes defined by the **XPerson** Entity Type Extension. The additional field defined by **XPerson** is '**HeavyMachineryCertified**' and it applies to instances of **Person**.

When we invoke an add or an update transaction for **Person** we include this new field in the request. Since the new field is not part of the underlying **Person** schema, we include this in an "extended" portion of the request document that all requests support.

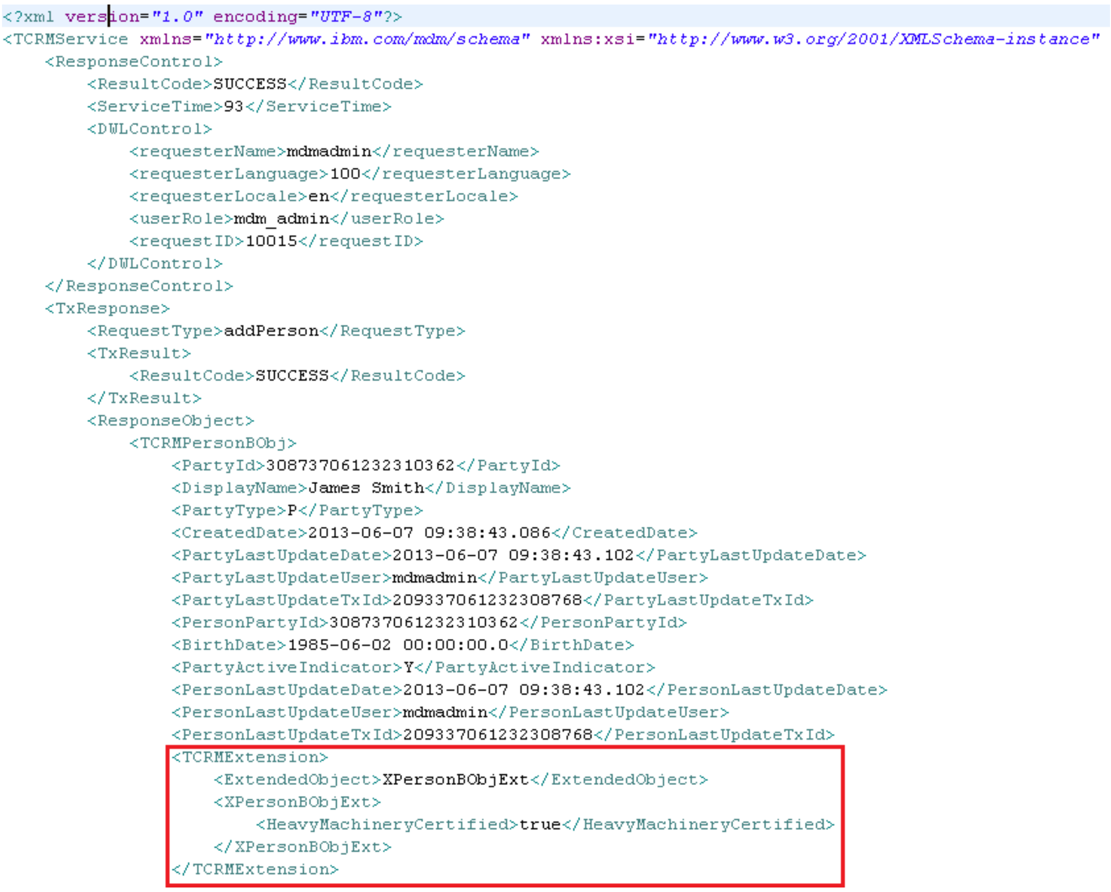
The transaction document **addPersonWithExtendedField.xml** shows how this is done:



```
<TCRMService xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.ibm.com/mdm/schema" xsi:schemaLocation="http://www.ibm.com/mdm/schema MDMDomains.xsd">
    <RequestControl>
        <requestID>10015</requestID>
        <DWLControl>
            <requesterName>cusadmin</requesterName>
            <requesterLanguage>100</requesterLanguage>
        </DWLControl>
    </RequestControl>
    <TCRMTx>
        <TCRMTxType>addPerson</TCRMTxType>
        <TCRMTxObject>TCRMPersonBObj</TCRMTxObject>
        <TCRMObject>
        <TCRMPersonBObj>
            <BirthDate>1985-06-023</BirthDate>
            <TCRMExtension>
                <ExtendedObject>XPersonBObjExt</ExtendedObject>
                <XPersonBObjExt>
                   <HeavyMachineryCertified>true</HeavyMachineryCertified>
                </XPersonBObjExt>
            </TCRMExtension>
            <TCRMPersonNameBObj>
                <NameUsageType>1</NameUsageType>
                <GivenNameOne>James</GivenNameOne>
                <LastName>Smith</LastName>
            </TCRMPersonNameBObj>
        </TCRMPersonBObj>
        </TCRMObject>
    </TCRMTx>
</TCRMService>
```

The extended information is stored in the **TCRMExtension** block (supported by all business objects). We specify the extension object that being provided in the **ExtendedObject** field and then provide the extension object itself – in this case this **XPersonBObjExt.**

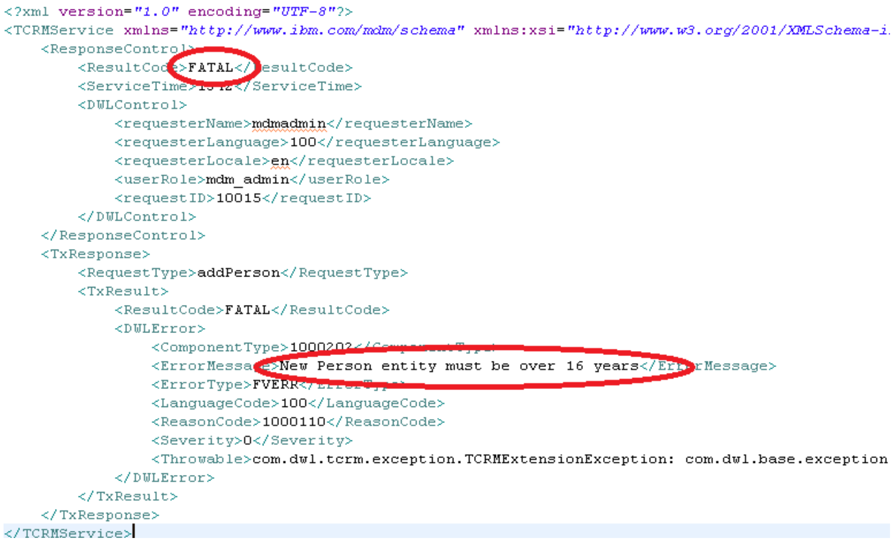
- Invoke this transaction on the server and check the results as before. You should see a successful invocation of **addPerson** with populated field values and the **TXRMExtension** block:



```
<?xml version="1.0" encoding="UTF-8"?>
<TCRMService xmlns="http://www.ibm.com/mdm/schema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <ResponseControl>
        <ResultCode>SUCCESS</ResultCode>
        <ServiceTime>93</ServiceTime>
        <DWLControl>
            <requesterName>mdmadmin</requesterName>
            <requesterLanguage>100</requesterLanguage>
            <requesterLocale>en</requesterLocale>
            <userRole>mdm_admin</userRole>
            <requestID>10015</requestID>
        </DWLControl>
    </ResponseControl>
    <TxResponse>
        <RequestType>addPerson</RequestType>
        <TxResult>
            <ResultCode>SUCCESS</ResultCode>
        </TxResult>
        <ResponseObject>
            <TCRMPersonBObj>
                <PartyId>308737061232310362</PartyId>
                <DisplayName>James Smith</DisplayName>
                <PartyType>P</PartyType>
                <CreatedDate>2013-06-07 09:38:43.086</CreatedDate>
                <PartyLastUpdateDate>2013-06-07 09:38:43.102</PartyLastUpdateDate>
                <PartyLastUpdateUser>mdmadmin</PartyLastUpdateUser>
                <PartyLastUpdateTxId>209337061232308768</PartyLastUpdateTxId>
                <PersonPartyId>308737061232310362</PersonPartyId>
                <BirthDate>1985-06-02 00:00:00.0</BirthDate>
                <PartyActiveIndicator>Y</PartyActiveIndicator>
                <PersonLastUpdateDate>2013-06-07 09:38:43.102</PersonLastUpdateDate>
                <PersonLastUpdateUser>mdmadmin</PersonLastUpdateUser>
                <PersonLastUpdateTxId>209337061232308768</PersonLastUpdateTxId>
                <TCRMExtension>
                    <ExtendedObject>XPersonBObjExt</ExtendedObject>
                    <XPersonBObjExt>
                        <HeavyMachineryCertified>true</HeavyMachineryCertified>
                    </XPersonBObjExt>
                </TCRMExtension>
```

Test 3 - Checking the Behaviour Extension logic is working

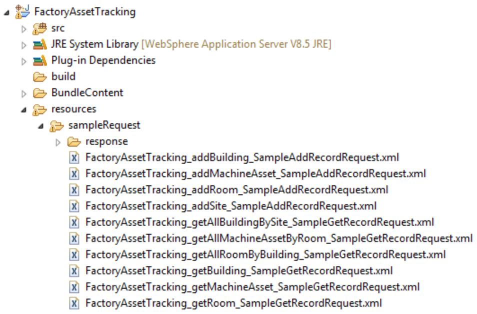
- The file **addPersonNotSixteenYet.xml** contains an **addPerson** request for someone who is not yet sixteen years old. Try running this transaction and ensure the request is rejected with the following error response:



```
<?xml version="1.0" encoding="UTF-8"?>
<TCRMService xmlns="http://www.ibm.com/mdm/schema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-i
    <ResponseControl>
        <ResultCode>FATAL</ResultCode>
        <ServiceTime>[illegible]</ServiceTime>
        <DWLControl>
            <requesterName>mdmadmin</requesterName>
            <requesterLanguage>100</requesterLanguage>
            <requesterLocale>en</requesterLocale>
            <userRole>mdm_admin</userRole>
            <requestID>10015</requestID>
        </DWLControl>
    </ResponseControl>
    <TxResponse>
        <RequestType>addPerson</RequestType>
        <TxResult>
            <ResultCode>FATAL</ResultCode>
            <DWLError>
                <ComponentType>1000202</ComponentType>
                <ErrorMessage>New Person entity must be over 16 years</ErrorMessage>
                <ErrorType>FVERR</ErrorType>
                <LanguageCode>100</LanguageCode>
                <ReasonCode>1000110</ReasonCode>
                <Severity>0</Severity>
                <Throwable>com.dwl.tcrm.exception.TCRMExtensionException: com.dwl.base.exception
            </DWLError>
        </TxResult>
    </TxResponse>
</TCRMService>
```

Test 4 - Try some other transactions yourself

- Try creating your own transaction requests using the templates under **actoryAssetTracking/resources/sampleRequest/**



```
FactoryAssetTracking
    src
    JRE System Library [WebSphere Application Server V8.5 JRE]
    Plug-in Dependencies
    build
    BundleContent
    resources
        sampleRequest
            response
            FactoryAssetTracking_addBuilding_SampleAddRecordRequest.xml
            FactoryAssetTracking_addMachineAsset_SampleAddRecordRequest.xml
            FactoryAssetTracking_addRoom_SampleAddRecordRequest.xml
            FactoryAssetTracking_addSite_SampleAddRecordRequest.xml
            FactoryAssetTracking_getAllBuildingBySite_SampleGetRecordRequest.xml
            FactoryAssetTracking_getAllMachineAssetByRoom_SampleGetRecordRequest.xml
            FactoryAssetTracking_getAllRoomByBuilding_SampleGetRecordRequest.xml
            FactoryAssetTracking_getBuilding_SampleGetRecordRequest.xml
            FactoryAssetTracking_getMachineAsset_SampleGetRecordRequest.xml
            FactoryAssetTracking_getRoom_SampleGetRecordRequest.xml
```