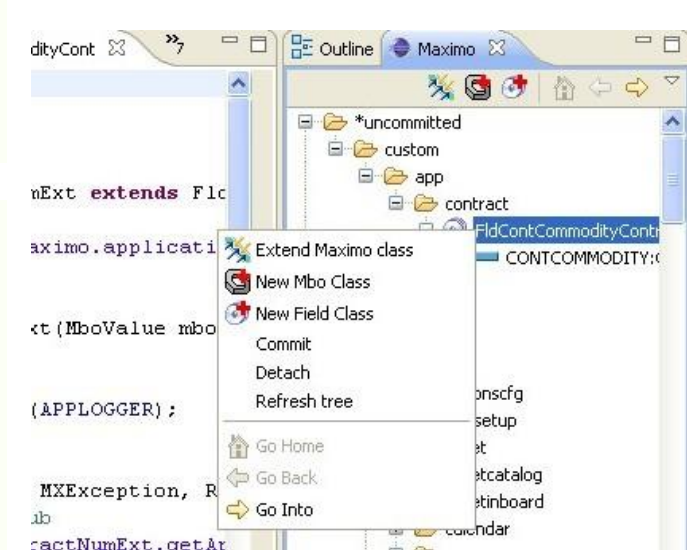


What is Maximo Developer?

Maximo Developer is an Eclipse plug-in designated for customization of IBM Maximo Asset Management System. It can help Maximo consultant or developer by automatically doing lot of boring and repetitive tasks.

- Extending of existing Mbo and field classes
- Automatic project setup for Maximo
- Creation of new Mbo and field classes
- Automatic remote interface creation
- Remote interface synchronization with Mbo classes
- Automatic rmic
- Easy visual browsing
- Easy detaching and attaching of classes to Mbos and Mbo attributes
- ...and more



Prerequisites

Maximo 5, Maximo 6 or Maximo 7 has to be installed on the user's machine. Java 1.5 or higher and Eclipse 3.3 or higher are also required. If you have already set default java home on your machine (typically this is java used for running your application server), you can edit eclipse.ini file, and set the virtual machine used for running Eclipse in this file, like on the picture below. Another option is to create shortcut (for Windows system) and define this option in the target fields.



Installing Maximo developer

In the Help menu of Eclipse choose:

Software Updates -> Find and Install -> Search for new features to install

Click on "New Remote Site" button. Enter the following values:

Name: Maximo Developer

URL: <http://maximodeveloper.sourceforge.net/update-site/>

Select the newly created remote site in dialog, and go through the process of installation.

Alternately, if you are not able to access update site , you can get plugin jar file on the project site:

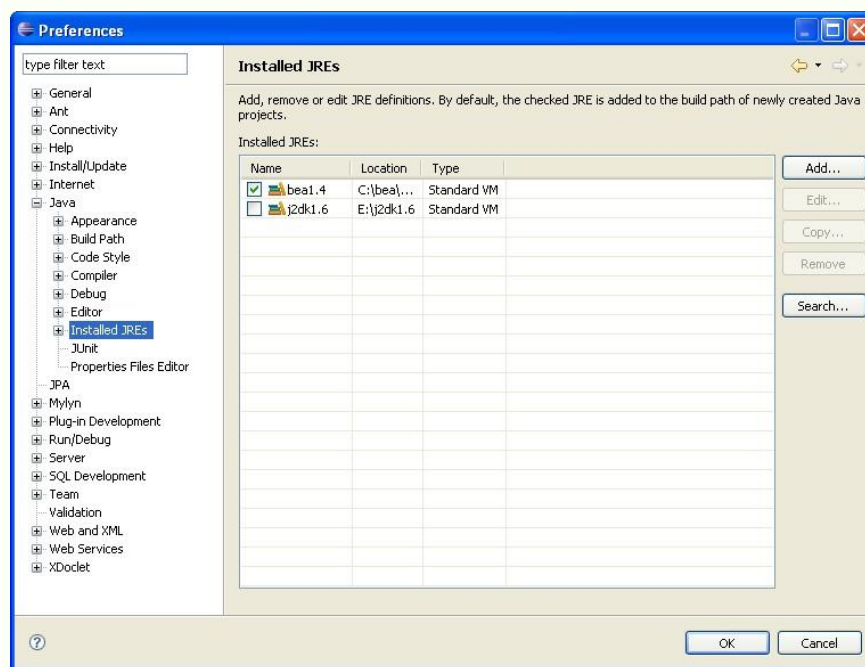
<http://downloads.sourceforge.net/maximodeveloper>

Download the latest release jar file, and put it in <Eclipse installation dir>/plugins directory

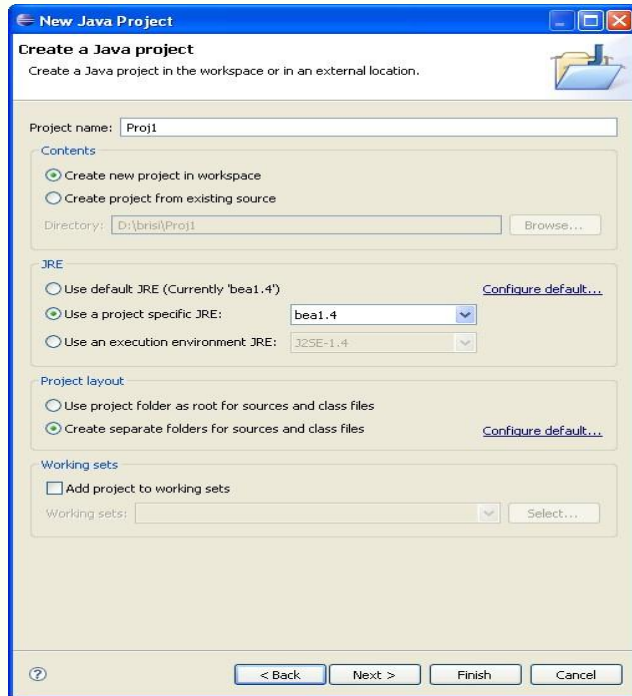
Starting Maximo developer

First, switch to Java perspective, Maximo Developer is designed to work in this perspective.

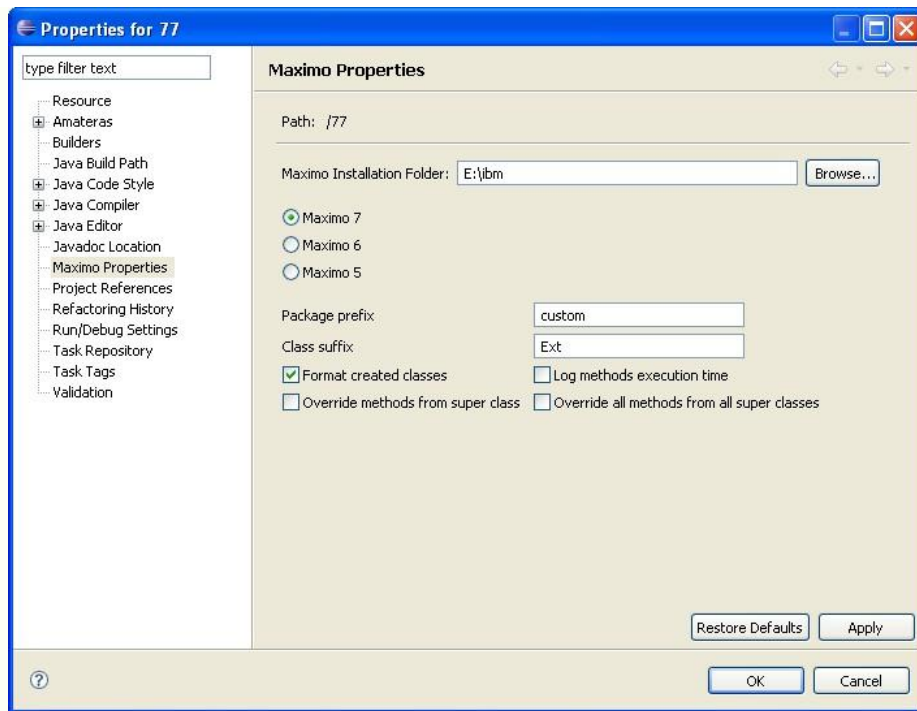
Next, make sure that you have configured all VMs that are needed for compiling your project classes. Maximo 7 uses Jdk1.5.0, Maximo 6 uses Jdk1.4.2, Maximo 5 uses Jdk1.3. Typically, you will use virtual machine provided by application server vendor. Below is an example for Maximo 6 and Weblogic 8.1. Go to Window->Preferences->Java->Installed JREs, and configure all JREs that you need.



Now, create a new Java project, and set JVM to the JVM of the application server you will be using for running Maximo. You can use existing source, download source from version control system, or create everything from the scratch, it won't affect Maximo Developer.



Next step is to define project's Maximo installation directory. Once you have defined it, project will become Maximo project, and Maximo Developer will start working. Go to Project->Properties->Maximo properties



First text box, Maximo installation folder is self explanatory for versions 5 and 6. For Maximo 7 this is the uppermost folder in hierarchy, "ibm"(For example: E:\ibm). Maximo Developer will validate entered value, and check if this is really a Maximo folder.

You have to choose correct version of Maximo. Maximo Developer will check property file, and if it doesn't exist, it will report an error. For Maximo 5 you need to use MXServer.properties, Maximo Developer will not recognize any other property file.

Package prefix and class suffix properties need some additional explanation. As someone working with Maximo, you are certainly acquainted with class and package organization in Maximo. Package psdi is a root package for all business logic and web tier classes. When you customize some behavior in Maximo, typically you extend the existing class, and attach it to the object previously used by original class. You have to name a new class, and this name has to be different from original class. Prefix and suffix properties are designated to help with this and automatically choose the name that best suit you. Maximo Developer will keep the original hierarchy of packages for customized classes also, but will use package prefix as a root package instead of psdi. Class suffix will be amended to the class name, so that you can differentiate it from original class.

Example: you are extending the MboSet class psdi.app.asset.AssetAncestorSet (how you will do it will be explained in the rest of the document). You have set prefix and suffix properties as above. Maximo Developer will automatically create two new classes and two remote interfaces:

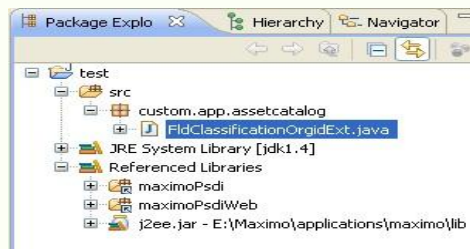
- custom.app.asset.AssetAncestorExt (Mbo class)
- custom.app.asset.AssetAcestorExtRemote (Remote interface for Mbo class)
- custom.app.asset.AssetAncestorExtSet (MboSet class)
- custom.app.asset.AssetAncestorExtSetRemote (Remote interface for MboSet class)

"Format created classes" checkbox should be set to true in most cases. Maximo Developer uses Eclipse formatting of code, and your code should always be formatted according to the formatting standards of your company, and this can be easily configured in Eclipse.

However, if you extend huge number of Maximo objects, formatting operation can be very slow. Typically, you would do such thing only for debugging purposes, because Maximo Developer inserts debug statements automatically. You don't need to format your code in that case, because you will delete it after you discover the bug. In such case, set this option to no.

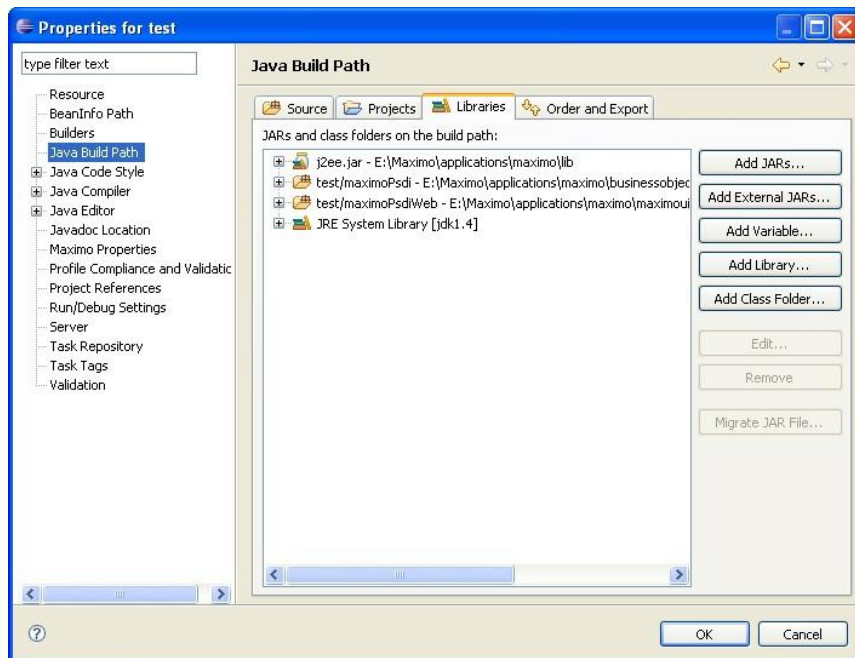
Some of the classes that you extend using Maximo Developer have large number of methods. If you choose to override all methods from the superclass, you will litter your code with unnecessary methods. So, typically you will uncheck "Override all methods from superclass" checkbox. You will leave it checked when you are extending classes for debugging purposes, because each extended method will have logging statement automatically inserted, so you can easily discover where the problem is originating. If you leave this checkbox blank, only abstract and most commonly used methods will be overridden.

After validation of Maximo directory, Maximo Developer will automatically import all necessary libraries from Maximo.



On the picture above you can see 3 libraries added automatically, two of them are actually links to Maximo classes.

On the build path you will see:

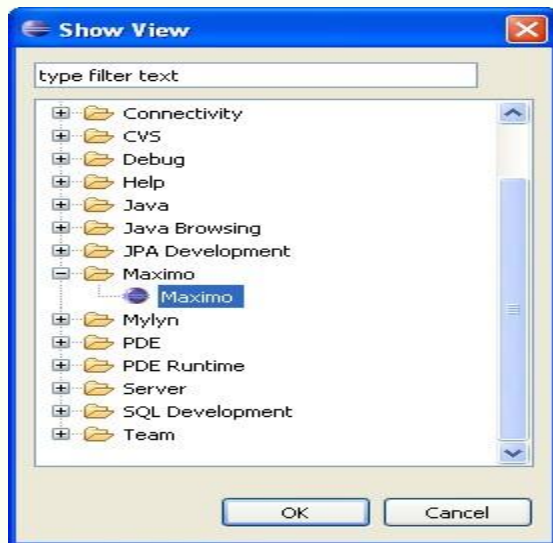


That means that build classpath is set and you don't need to set anything by yourself, unless you are using some other third party libraries.

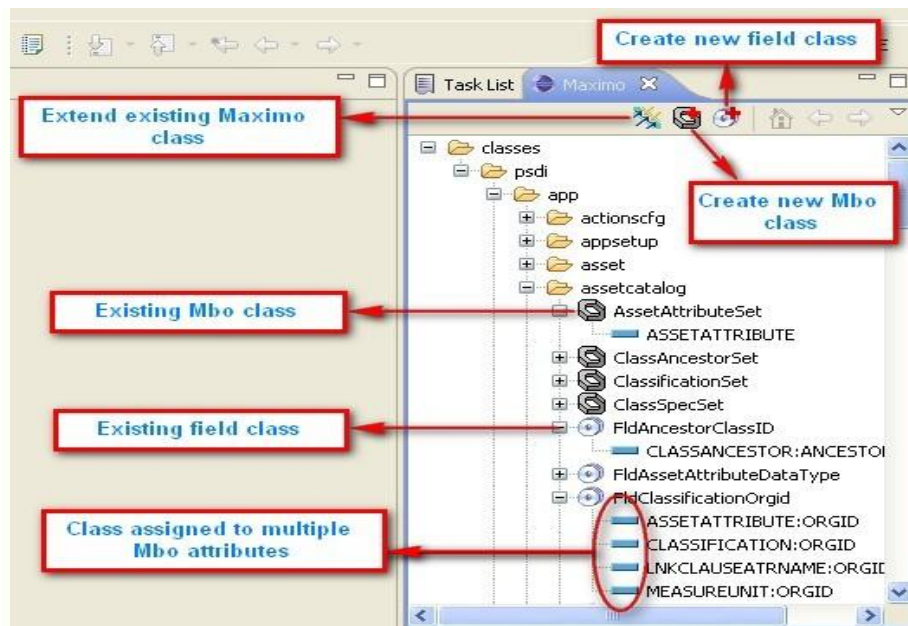
Running Maximo Developer

When you have defined Maximo directory for your project (you can have many different projects with different Maximo directories, but let's focus on just one), make sure connection to the Maximo database is present.

From the Java perspective open Maximo view: Click on Window->Show View->Other...->Maximo->Maximo.

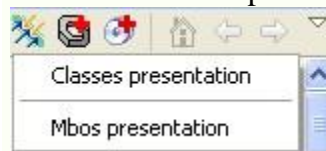


Navigate to the source folder in your project. Maximo view will then be populated with data from the Maximo database. If Maximo view appears on some other part of Eclipse workspace, you can drag and drop it where you want it to be.



Sometimes, you need different view of Maximo mbo and classes data. You can see this data organized by mbos and not by classes.

Switch to the mbo presentation using the drop-down list on the right side of the Maximo view.

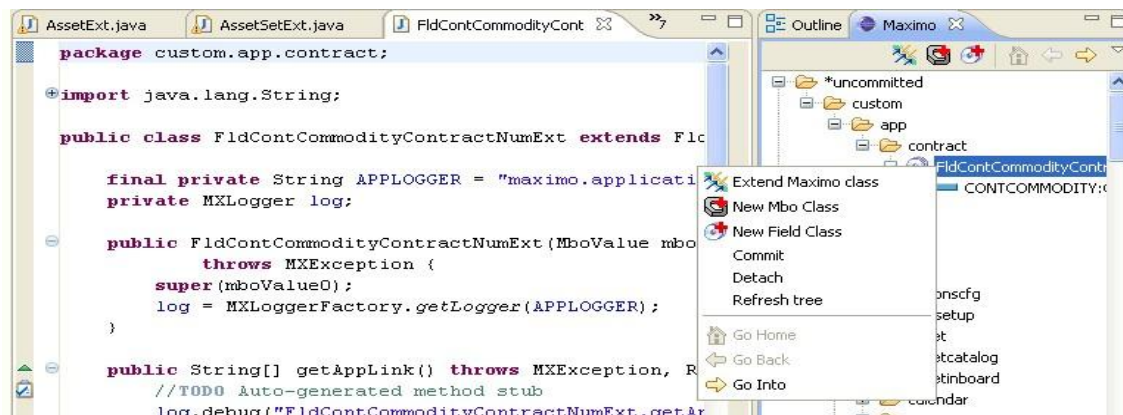


On the toolbar of Maximo view you can see icons for main functionalities. Let's start with one that you'll be using most often: "Extend existing Maximo class".

Extending existing Maximo class

First you have to select class(es) to be extended. As you can see from the picture above, one class can be assigned to multiple Maximo objects or attributes. If you select the class, all this attributes and objects will be also selected. This means that when you extend the class, extended class will be assigned to all the selected attributes. If you want to keep old functionality for some of the objects, select just objects and attributes for which you want new class to be attached to. If you attach it by mistake to all of the classes, don't worry, there is an easy way to remove them, as you will see soon.

When you have selected classes and/or objects and attributes, click on "Extend Maximo View" icon. New classes will be created and formatted with Eclipse formatter you have defined, and one of the new created classes will be opened in editor. Below is the example: user wanted to extend `FldClassificationOrgid` and attach it just to attribute `ORGID` of `ASSETATTRIBUTE` object.



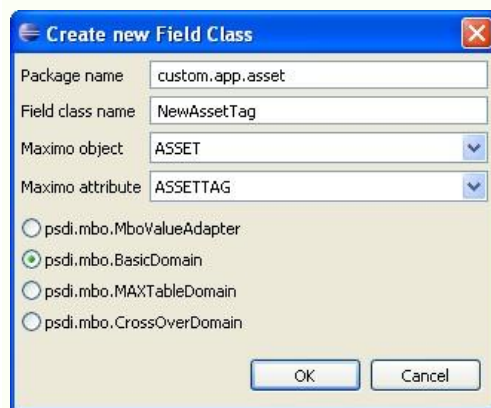
You can see that extended class is opened in new editor. Extended class will extend all public methods of the original class and extend also most commonly used methods from the classes up in the hierarchy. For example, for all field classes, methods init, action and validate will always be overridden in the new class.

Take a look at a Maximo view now; you can see that its content has changed after we have extended the class. Now you can see "*uncommitted" part in the Maximo view, which wasn't there before. This reminds the user that these classes were not yet committed to the database. If you right click any element in that part of the tree, you will see popup menu as above. You should not commit changes to the database until you have made changes that you wanted and compiled the class. When you commit it to the database, new class is ready for testing. You can also detach the created class. Attribute will use original class, but the new class is still in Eclipse, and you can attach it later to the same or any other Maximo object or attribute.

Creating new Mbo or field class

Creating new Mbo or field class is not much different from extending existing one. Basically you extend one of the basic Mbo or field classes. You will get the source code; class will be attached to the desired object. Only difference is that you will not extend all the superclass methods, because that would create hundreds of methods in the code. Only most commonly used methods will be extended.

Here is example for extending a field class; procedure is almost identical for a new Mbo.



You have to type in the name of the package and class name. You can choose Maximo object and Maximo attribute. If you leave blank one of these, class will be created, but not attached to any attribute. You have to choose one of the types of Field class from the radio button list.

Compiling Maximo classes

There is not much to say about compiling of Maximo classes. It is the same as for any Java project in Eclipse. However, some important features of Maximo Developer are happening in the background during the compiling, and you as a user should be aware of them.

After you compile the classes, Maximo Developer will check whether these are business objects or web tier classes.

Business objects will be copied to directory designated for them

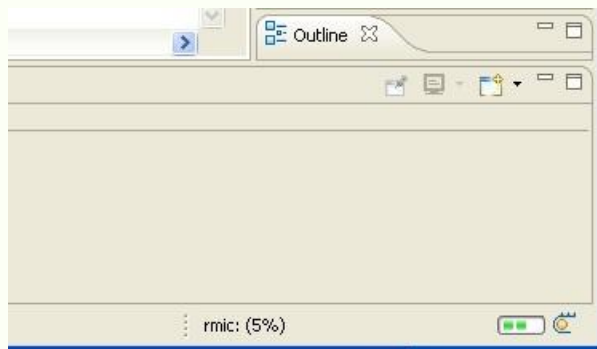
(in Maximo 6 applications/maximo/businessobjects/classes), and web classes will finish in their own directory (applications/maximo/maximouiweb/webmodule/WEB-INF/classes for Maximo 6).

In order for web classes to be copied to the correct directory, they have to follow the naming convention of Maximo Developer.

For example, if you define the root directory for custom classes as "custom", all classes under the custom/webclient will be considered as web classes, and everything else will be copied to the business objects directory.

Please make sure you follow this rule, otherwise you may lose time struggling with classpath issues

After this, Maximo Developer will rmic all Mbos that don't have conforming stubs and skeletons. Rmic'ing will occur in the background, and you can see it happening on Eclipse progress bar.



You can check now, new stubs and skeletons are created.

Name	Ext	Size	lDate	Attr
ContractTypeTermExt_Skel	class	37,637	01/15/2008 11:12-a--	
ContractTypeTermExt_Stub	class	68,919	01/15/2008 11:12-a--	
ContractExtSet_Skel	class	53,302	01/15/2008 11:12-a--	
ContractExtSet_Stub	class	97,901	01/15/2008 11:12-a--	
ContractTermExt_Skel	class	37,629	01/15/2008 11:12-a--	
ContractTermExt_Stub	class	68,907	01/15/2008 11:12-a--	
ContractTypeExtSet_Skel	class	53,310	01/15/2008 11:12-a--	
ContractTypeExtSet_Stub	class	97,913	01/15/2008 11:12-a--	
ContractStatusExtSet_Skel	class	53,314	01/15/2008 11:12-a--	
ContractStatusExtSet_Stub	class	97,870	01/15/2008 11:12-a--	
ContractTermExtSet_Skel	class	53,310	01/15/2008 11:12-a--	
ContractTermExtSet_Stub	class	97,913	01/15/2008 11:12-a--	
ContractStatusExt_Skel	class	37,633	01/15/2008 11:12-a--	
ContractStatusExt_Stub	class	68,867	01/15/2008 11:12-a--	
ContractTypeTermExtSet_Skel	class	53,318	01/15/2008 11:12-a--	
ContractTypeTermExtSet_Stub	class	97,925	01/15/2008 11:12-a--	
ContractTypeExt_Skel	class	37,905	01/15/2008 11:12-a--	
ContractTypeExt_Stub	class	69,433	01/15/2008 11:12-a--	
ContractExt_Skel	class	45,395	01/15/2008 11:12-a--	
ContractExt_Stub	class	83,922	01/15/2008 11:12-a--	
ContractExt	class	7,696	01/15/2008 11:12-a--	

Rmicing is an expensive operation and will occur only when no stubs and skeletons are found for Mbo, or when new remote method is added to Mbo or MboSet.

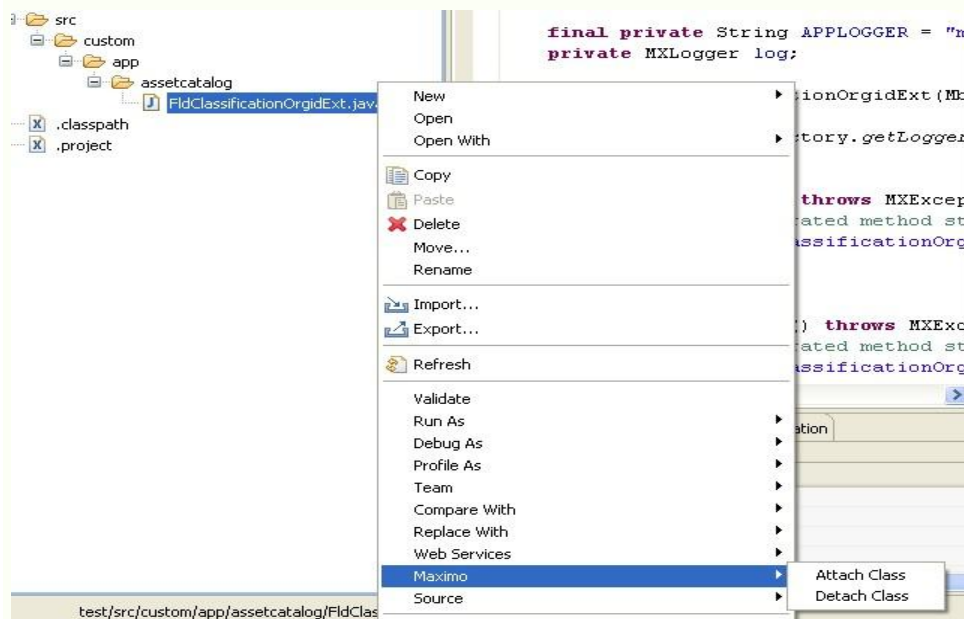
Synchronizing changes with remote interfaces

If you begin to use Maximo Developer, you will never need to worry about remote interfaces again. They are not real business logic interfaces anyway, just artifacts required by RMI protocol. As you have already seen, they will be automatically created when you create a new Mbo. If you change the class and add or remove remote method, or change the signature of remote method, all these changes are reflected in remote interfaces also. Method is considered remote if it throws RemoteException.

When you compile the classes after these changes, rmic will occur again in the background.

Detaching classes from objects and attaching them to objects

You have already seen one way to detach a class from Maximo object or attribute-by right clicking on selected object(s), and select detach from popup menu. Another way is to detach a class is from Package Explorer or Navigator. Select the class you want to detach and right click on it. From the drop down confirm detachment from each object or attribute this class is attached to. If you want to detach class from all objects, just click on "All" button.



Another useful way to detach a class is simple deletion from Package explorer or Navigator (if you delete a class directly from the file system while running Maximo Developer, nothing will happen). This can be useful if you extend lot of classes by mistake, for example. Another possible way to use this feature is for debugging. Maximo Developer will generate debug statement for each extended method. Also, debug level will be on the level of module, like in the rest of Maximo. For example, if you subclass the class from inventory module, logger will be maximo.application.INVENTORY. This gives you the power to use this for debugging. Suppose you have seen some bug in inventory module, and you are not able to find it, because you have lost the track of order of execution of field and Mbo classes. You would simply extend all inventory classes, both original psdi, and customized (if class to be created already exist, it will be used, so you don't need to be afraid that anything will be overwritten). Now change the logging level of maximo.application.INVENTORY to DEBUG in logging properties, and find out how calls are being propagated throw all classes in the module. Once you have found where the root of the problem is, just delete all unnecessary classes, and system will be in the state just before the massive extension.

As you can see from the picture above, you can attach existing class from the package explorer to Maximo objects or attributes. You can attach only field classes and MboSet classes, just like in Maximo Database Configuration. Maximo Developer will automatically recognize what type of a class is dealt with, and give you list of objects if this is MboSet, or list of object and attributes if this is field class.