

# realMethods Framework

## Quick Start Guide

# realMethods Framework

## Quick Start Guide

<http://www.realmethods.com>

### realMethods Framework Quick Start Guide

Copyright © 2001-2017 applied to realMethods, Inc. including this documentation, all demonstrations, and all software. All rights reserved. The document is furnished as is without warranty of any kind, and is not intended for use in production. All warranties on this document are hereby disclaimed including the warranties of usability and for any purpose.

### **Trademarks**

"realMethods Framework" is a trademark of realMethods in the United States and other countries. Microsoft, Windows, Windows NT, the Windows logo, and IIS are trademarks or registered trademarks of Microsoft Corporation in the United States, other countries, or both. UNIX is a registered trademark of The Open Group in the United States and other countries. Sun, Solaris, Java, Enterprise Java Beans, EJB, J2EE, and all Java-based trademarks or logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. IBM and MQ Series are trademarks or registered trademarks of International Business Machines Corp. in the United States and other countries.

Copyright (C) 1999 - The Apache Software Foundation. All rights reserved.  
This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

### **Disclaimer**

The use of the software developed by the Apache Software Foundation is only for the purpose of demonstrating integration with Jakarta Struts, Log4J, Hibernate, and Velocity.

Other company, product, and service names mentioned in this document may be trademarks or service marks of others.

What's New .....	4
AIB (Application Generator) Improvements.....	4
Hibernate Integration.....	4
UI Modeling and Code Generation.....	4
Enumerated Type Support .....	4
Auditable .....	5
Struts Validator Generation for Forms .....	5
Create a Proof of Concept (POC) Application .....	6
Run the Code Generator.....	6
Create the POC Project.....	6
Load an XMI File.....	6
Tier Options.....	6
Hibernate Settings.....	6
Generating the POC Code .....	7
Building the (POC).....	8
Deployment .....	8
Execution .....	8
Generated Application Client-side Overview .....	9
Building Your Application.....	15
Designing the Class Diagram .....	15
Declaring a Primary Key Attribute .....	16
Declaring an Association.....	16
Composite Association.....	17
Enumerator Type.....	17
Declaring a Business Method.....	18
Declaring Auditable.....	18
Exporting the Model .....	18
Stereotype Overview.....	18
Creating an AIB Project.....	19
What Next?.....	25
Proof of Concept FAQ .....	26
What is Hibernate, and where can I learn more about Hibernate? .....	26
Where can I find log output? .....	26
I cannot run ANT using the generated build.xml file?.....	27
Why am I having issues reading or writing data to my database? .....	27
Can I load the POC's XMI file into my UML tool? .....	28
Why did the POC fail to deploy on my application server? .....	28

# What's New

## AIB (Application Generator) Improvements

The AIB continues to come in two flavors - Standalone or Eclipse plugin:

Both versions of the AIB have been much improved with support for reverse engineering, a project creation wizard, better project management, faster XMI file parsing, application preferences, as well as a superior look and feel, as compared to previous versions.

## Hibernate Integration

The AIB continues to generate a Data Access layer that utilizes the extremely popular persistence open source project Hibernate. Hibernate is a powerful, ultra-high performance object/relational persistence and query service. The framework and AIB handle all code generation surrounding the interaction and configuration of Hibernate, both during the ANT build process as well as application deployment. Your generated DAO classes leverage Hibernate's HQL for each business object persistence transaction, allowing you to easily extend to allow for more custom queries.

Importantly, with the integration of Hibernate within the generate application DAO layer, the AIB is now easily able to generate an application to target all SQL database management systems, as well as all popular J2EE application servers.

## UI Modeling and Code Generation

### Enumerated Type Support

#### *Description:*

You are now able to define enumerated types within a class diagram. Simply create a new class and apply one more attributes to it. Importantly, each attribute must be of the same type, and ideally should be of type Integer, Short, Long, or String. In order to make use of the enumerated type class, you have to draw a single association to it from the owning class. Also, you must apply a stereotype by the name of *enumeration* to the enumerated class.

*UML Stereotype:*      enumeration

## Auditable

### *Description:*

This feature is useful when you wish to automatically apply a created timestamp and last updated timestamp to a business object. This is useful for the purpose of auditing. All you have to do is define a stereotype by the name of *Auditable* to the targeted class.

*UML Stereotype:*      Auditable

## Struts Validator Generation for Forms

The AIB now generates a single *validation.xml*, which contains all the Struts syntax required for validation within each generated form.

# Create a Proof of Concept (POC) Application

## Run the Code Generator

The code generator application is named the *AIB* (Application Infrastructure Builder). It can be run as a stand alone Java application, or as an Eclipse 3.x plugin. To use the Eclipse plugin, locate the `aib.eclipseplugin.html` file in the `\docs` directory of the installation.

Read on in order to learn how to use the stand alone Java application version.

Locate and execute the *runAIB* file in the `\aib` directory of the realMethods installation.

## Create the POC Project

Once the AIB is up and running, right-mouse click in the Navigation pane and select the *New → Project* option. When the New Project wizard appears, open the *realMethods Framework* folder, select the *Application Infrastructure Builder*, and click *Next*. Now provide a unique name for your project. For this document, the name is assumed to be *poc*.

## Load an XMI File

Use the *AIB* menu option to locate and open one of the provided XMI files (*catalog.xmi*, *company.xmi*) in the *poc/model* directory of the realMethods installation. Once the XMI file is loaded, the Model View will be populated with a tree control that represents the loaded domain model. Take a few minutes to traverse the tree and select the different entities to inspect their contents.

## Tier Options

Now select the *Tier Options* tab. This tab contains the different tier-by-tier options you will want to apply in order to create the specific type of J2EE application you are interested in.

## Hibernate Settings

Under the *Data Tier* of the *Tier Options Tab*, you will need to correctly set the following Hibernate related parameters:

- Database Type
- Transaction Manager
- User ID

- Password
- Driver Name
- URL

Close the Hibernate Settings dialog box, click the File menu option and select Save. This step is extremely important since these settings are used by Hibernate to connect to your database in order to create the db schema of the POC.

#### **Special Note for MS SQL Server Users**

[Hibernate FAQ](#) indicates that when using a MS SQL Server 2000 JDBC driver, you will have to include the following as part of the URL:

SelectMethod=Cursor

For example -

```
jdbc:microsoft:sqlserver://LITTLEGUY_NT:1433;SelectMethod=Cursor
```

## **Generating the POC Code**

In order to generate all the files necessary to create and deploy the POC application, select the *AIB* → *Generate application files* menu option. You may need to change the default *Output Directory* value. Once the code generation is complete, you are ready to build the Proof of Concept into a single *poc.ear* file.

The generate files will be placed in the folder specified within the *Application Tier* options of the *Tier Options* tab. Take a few minutes to browse the generated files in order to gain an appreciation for the 1000's of lines of codes and hundreds of files automatically generated. Importantly, each file comes from a modifiable Velocity Template file, allowing you to easily customize.

## Building the (POC)

Once the code generation is completed the **AIB** will automatically build the application EAR file using Jakarta ANT against a generated *build.xml* file. You may want to locate the generated ANT *build.xml* in the root of the output directory. As needed change the following ANT properties located at the top of the *build.xml* file

- *rm.home*      realMethods Installation Root Directory
- *j2ee.home*    location of the J2EE installation
- *db.jars*      vendor db driver jars if using Hibernate
- *deploy.dir*   location where to copy the application EAR file

The *db.jars* property must be set appropriately so the *hibernate* task of the ANT build process can create the necessary tables and other db artifacts of the targeted database server.

## Deployment

The resulting EAR file from the ANT build will be copied to the directory specified by the *deploy.dir* property location. The ear, war, and jar files built during the ANT run are self-contained, meaning no other external jar or zip files are needed to be in the classpath **with the exception** of the jar files necessary to access your database. These are the same jar files you should have provided as the value to the *db.jars* property of the generated *build.xml* file.

You are now ready to deploy your application to your application server. There is nothing special or tricky about the built EAR file, so your application server documentation deployment instructions should be sufficient.

If you are unsure how to deploy an application to your application server, please review the step-by-step instructions in the realMethods Framework Developer's Guide.

## Execution

Once your application server is started, and the application (*poc.ear*) is deployed, type the following URL into your Web browser:

***http://<your application server name>:<port#>/poc/***

For example, if you deploy the POC on Weblogic, the URL is:

*http://localhost:7001/poc/*

***Do not forget the terminating '/' character.***



## Generated Application Client-side Overview

Generated logon page and logon Struts Action ties directly into the framework's security manager, which can be easily customized to integrate into your security solution

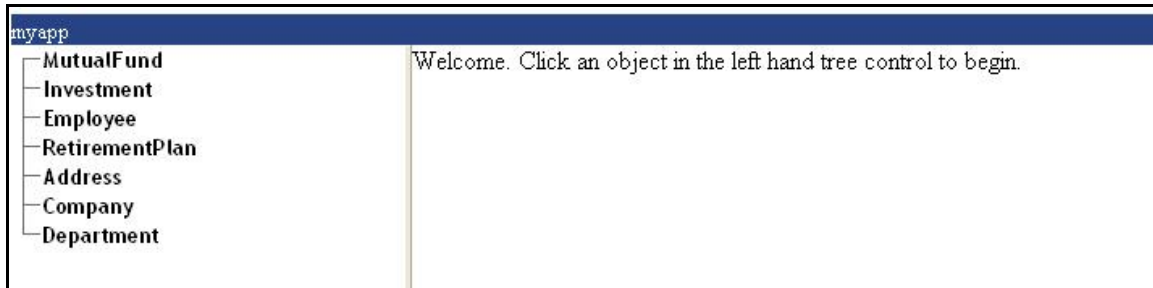


myapp Access

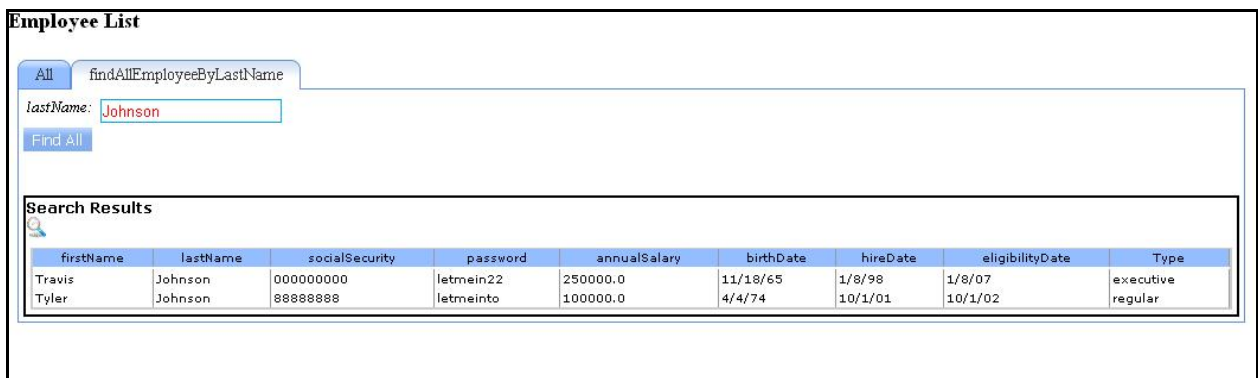
User ID:

Password:

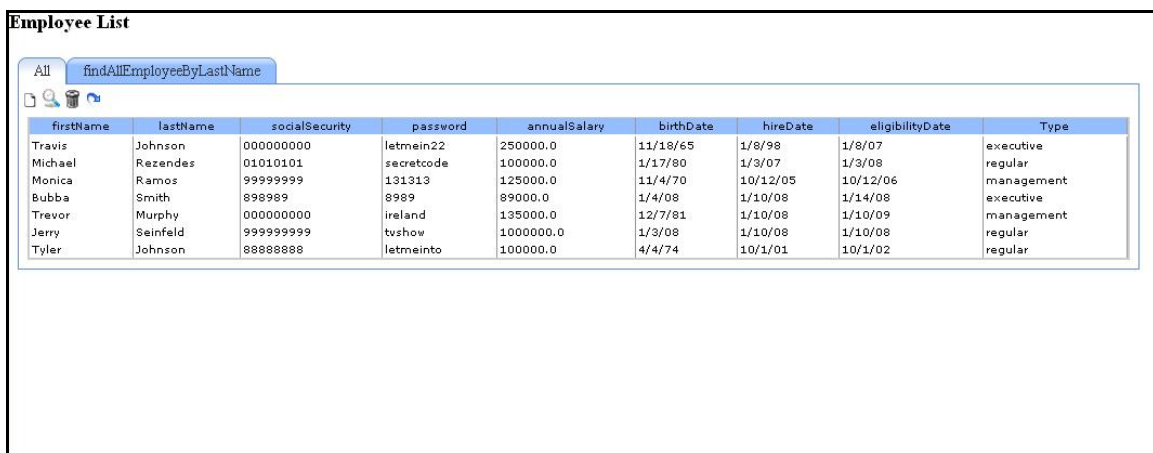
Tree control simplifies navigation to desired functionality



Easily view all objects of a certain type...



... or view via one of your custom defined finder methods



Toolbars provide multiple options to execute application features



Forms make data entry simple and fast

**Employee Data Form**

*firstName:*

*lastName:*

*socialSecurity:*

*password:*

*annualSalary:*

*birthDate:*  

*hireDate:*  

*eligibilityDate:*  

*Type:*  

**Address**

*addressLine1:*

*addressLine2:*

*city:*

*stateAbbrev:*

*postalCode:*

Form field validation is automatically provided for on a field by field basis

**Employee Data Form**

*firstName:*


*lastName:*


*socialSecurity:*


**This is a required field.**


*password:*

*annualSalary:*

*birthDate:*  

*hireDate:*  

*eligibilityDate:*  

*Type:*  

**Address**

*addressLine1:*

*addressLine2:*

*city:*

*stateAbbrev:*

*postalCode:*

Each business entity can be easily inspected, allowing for the quick view and modification of native data, composites, as well as single and multiple associations. Each pane of the inspector view is collapsible for easy viewing.

Company Inspector

...

name:

Gypsum Financial Group

Address

addressLine1:

336 Main Street

addressLine2:

Suite 1000

city:

Plympton

stateAbbrev:

MA

postalCode:

02333

Save

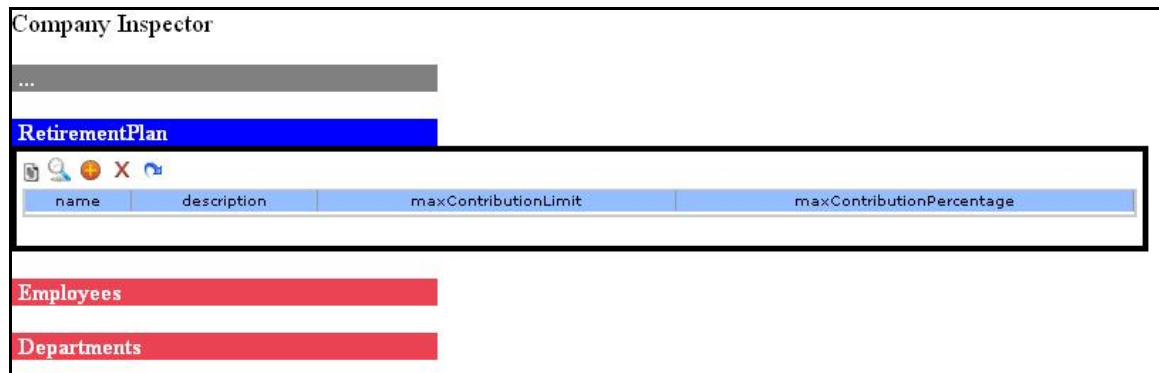
Reset


RetirementPlan

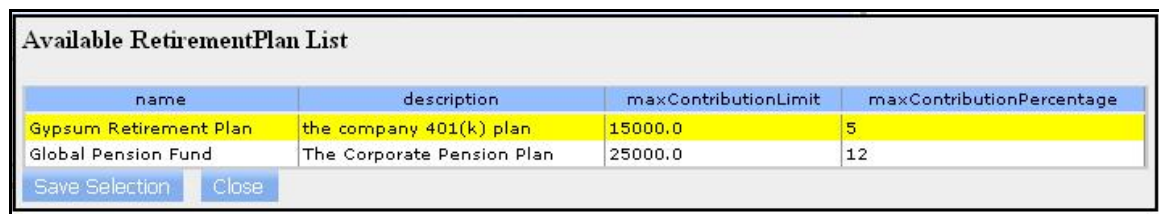
Employees

Departments

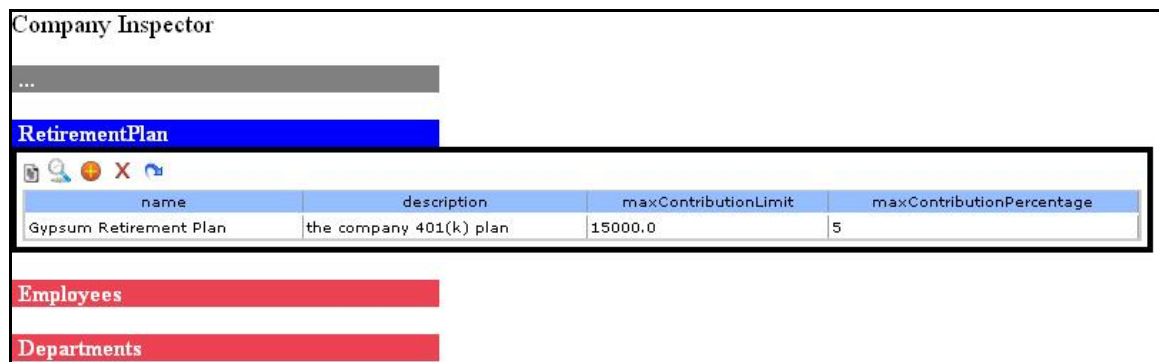
Expand on a single association panel in order to apply a single association to the object being inspected








Click the  icon to select a single object from a list



Only the list will be automatically updated with the selection



- Click on the  to edit the immediate data of the single association object
- Click on the  to inspect all data (immediate and its associations) of the single association
- Click on the  to associate or re-association an object with the object being inspected
- Click on the  to remove the current association
- Click on the  to refresh the current association view

Expand on a multiple association panel in order to apply one or more associations to the object being inspected

Company Inspector


...

**RetirementPlan**

**Employees**

firstName	lastName	socialSecurity	password	annualSalary	birthDate	hireDate	eligibilityDate	Type

**Departments**

Click the  icon to select a one or more objects from a list

**Current Employee List**

firstName	lastName	socialSecurity	password	annualSalary	birthDate	hireDate	eligibilityDate	Type

Apply as Selection(s) Below

**Available Employee List**

firstName	lastName	socialSecurity	password	annualSalary	birthDate	hireDate	eligibilityDate	Type
Travis	Johnson	000000000	letmein22	250000.0	11/18/65	1/8/98	1/8/07	executive
Michael	Rezendes	01010101	secretcode	100000.0	1/17/80	1/3/07	1/3/08	regular
Monica	Ramos	99999999	131313	125000.0	11/4/70	10/12/05	10/12/06	management
Bubba	Smith	898989	8989	89000.0	1/4/08	1/10/08	1/14/08	executive
Trevor	Murphy	000000000	ireland	135000.0	12/7/81	1/10/08	1/10/09	management
Jerry	Seinfeld	99999999	tvshow	1000000.0	1/3/08	1/10/08	1/10/08	regular
Tyler	Johnson	88888888	letmeinto	100000.0	4/4/74	10/1/01	10/1/02	regular

Save Selection(s) Close

Only the list will be automatically updated with the selection

Company Inspector

...

**RetirementPlan**

**Employees**

firstName	lastName	socialSecurity	password	annualSalary	birthDate	hireDate	eligibilityDate	Type
Travis	Johnson	000000000	letmein22	250000.0	11/18/65	1/8/98	1/8/07	executive
Michael	Rezendes	01010101	secretcode	100000.0	1/17/80	1/3/07	1/3/08	regular
Monica	Ramos	99999999	131313	125000.0	11/4/70	10/12/05	10/12/06	management
Bubba	Smith	898989	8989	89000.0	1/4/08	1/10/08	1/14/08	executive
Trevor	Murphy	000000000	ireland	135000.0	12/7/81	1/10/08	1/10/09	management

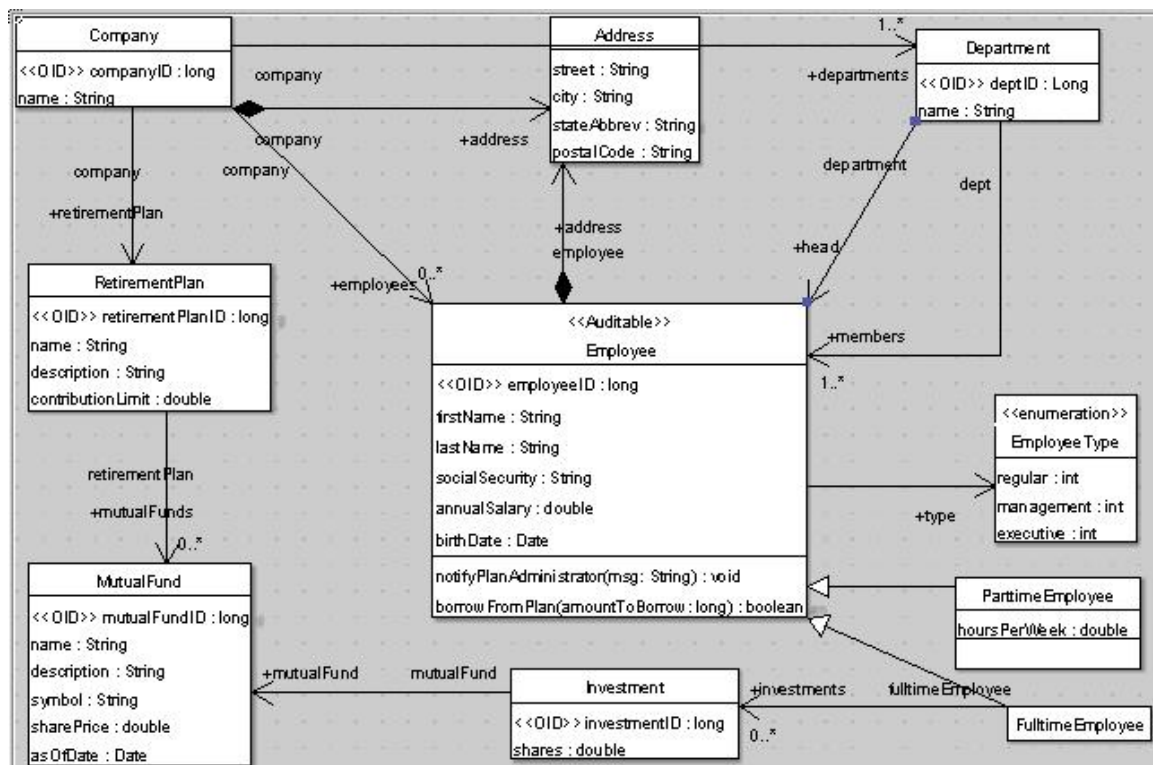
**Departments**

# Building Your Application

Being an MDA compliant platform specific tool, the realMethods Framework and AIB Code Generator require that you have first defined your business problem in terms of a class diagram. This is intentional since a Model Driven Architecture should encapsulate the business object domain, yet be devoid of any technology and design pattern specifics, which is the job of realMethods Framework and the AIB.

## Designing the Class Diagram

This is by far the most important step of building your application. A small error here will be magnified when the application is generated. Let's take a look at the POC's class diagram to reveal some important aspects to understand and consider.



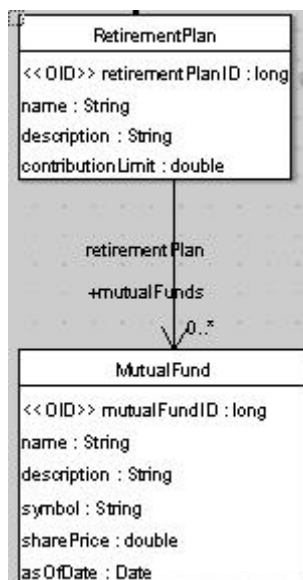
## Declaring a Primary Key Attribute

You can indicate that a class attribute is to be considered a primary key of the class by applying a stereotype named *OID* to the attribute. Although incredibly discouraged, you can apply this stereotype to more than one attribute, indicating that the class contains a compound key.

```
Investment
<<OID>> investmentID : long
shares : double
```

## Declaring an Association

When creating an association between two classes, make sure to denote the correct multiplicity on both ends. It is important that each end of the association be named, even if only one end is navigable (i.e. uni-directional). Take care to apply a meaningful name since this name will be used when generating associated variables and methods in your code. Also, make sure that no two associations of a single class share the same name, meaning each name must be unique.





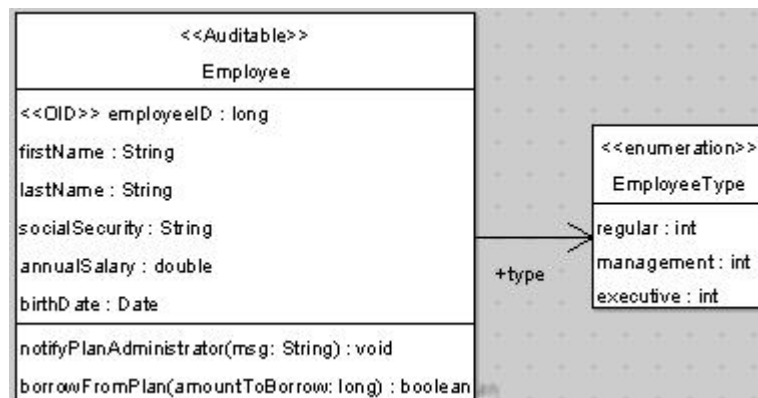
## Composite Association

This is a special association between a parent class and child class. A composite association is interpreted by the AIB as the parent owning the child. It causes the parent to have a reference to the child, but for the sake of presentation and persistence, the child is always created, saved, and loaded with the parent. Looking at the POC model again, this means each time a Company is created the Company's Address data is filled in by the user, persisted, and then loaded each time the Company is displayed.



## Enumerator Type

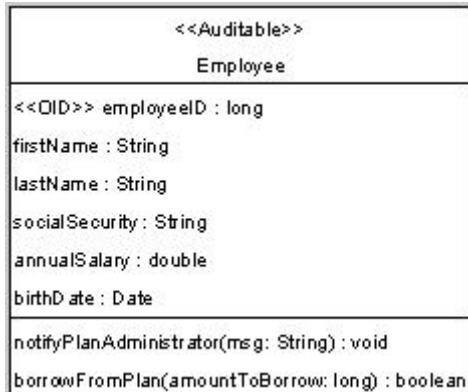
An Enumerated type isn't really a business object, but rather encapsulates a set of valid values a particular abstract can take. When modeling an enumerator, you must take the following steps so the AIB Code Generator can appropriately identify and generate the relevant code:



1. Create the enumerated class as regular class in the model, but make sure to apply a stereotype by the name of *enumeration*. This stereotype tells the AIB Code Generator to treat this class as an enumerator.
2. Make sure all attributes of the enumerator class are of the same type. The supported types are int, short, long, and String.
3. Like any other association, create a uniquely named association between the parent class and the enumerator class. It is only necessary to name the navigable end of the association (the one from the parent to the enumerator class itself)

## Declaring a Business Method

Although the AIB Code Generator isn't able to magically fill in the business logic of your application, it can generate the code of any declared business methods.



## Declaring Auditable

You can apply this stereotype to any class you want to automatically have a last updated and created timestamps associated with it. When using Hibernate these fields are automatically populated by the framework. These fields can be useful for audit trail usage.

## Exporting the Model

All popular UML tools (Rose, TogetherJ, Poseidon, MagicDraw, and Argos) support the ability to export your business model to a single UML 1.3 compliant XMI file. Importantly, this file must be of **XMI 1.0** type. You will load this file into the AIB Code Generator when you build an AIB project.

### Enterprise Edition Users

Refer to the Architecture Guide for more information on exporting your model.

## Stereotype Overview

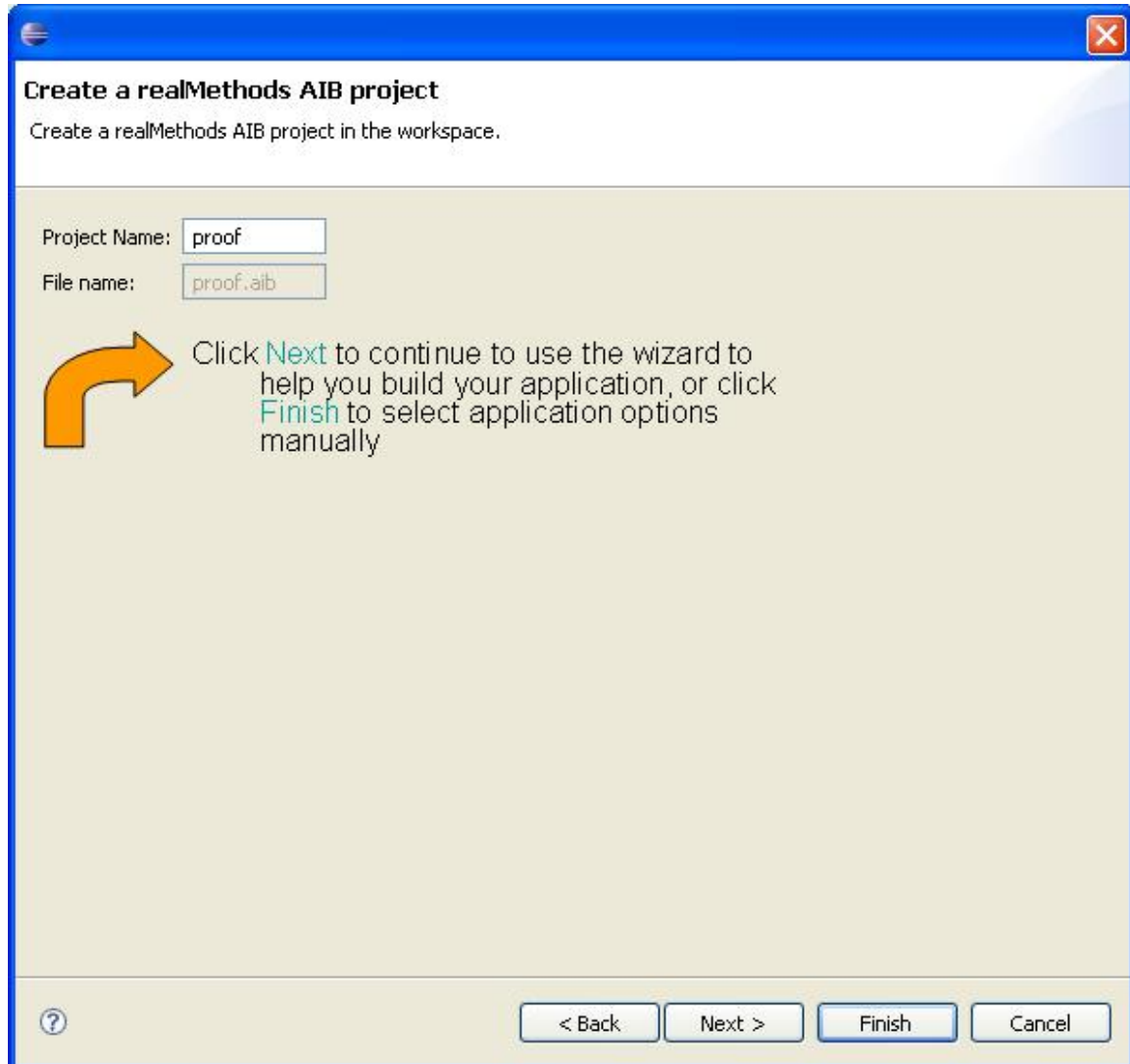
Name	Description	Model Usage
<i>OID</i>	Declares an attribute as a primary key	Attribute
<i>Enumeration</i>	Declares a class as an Enumerated type	Class
<i>Auditable</i>	Generates a <i>created</i> and <i>lastUpdated</i> timestamp attributes	Class

## Creating an AIB Project

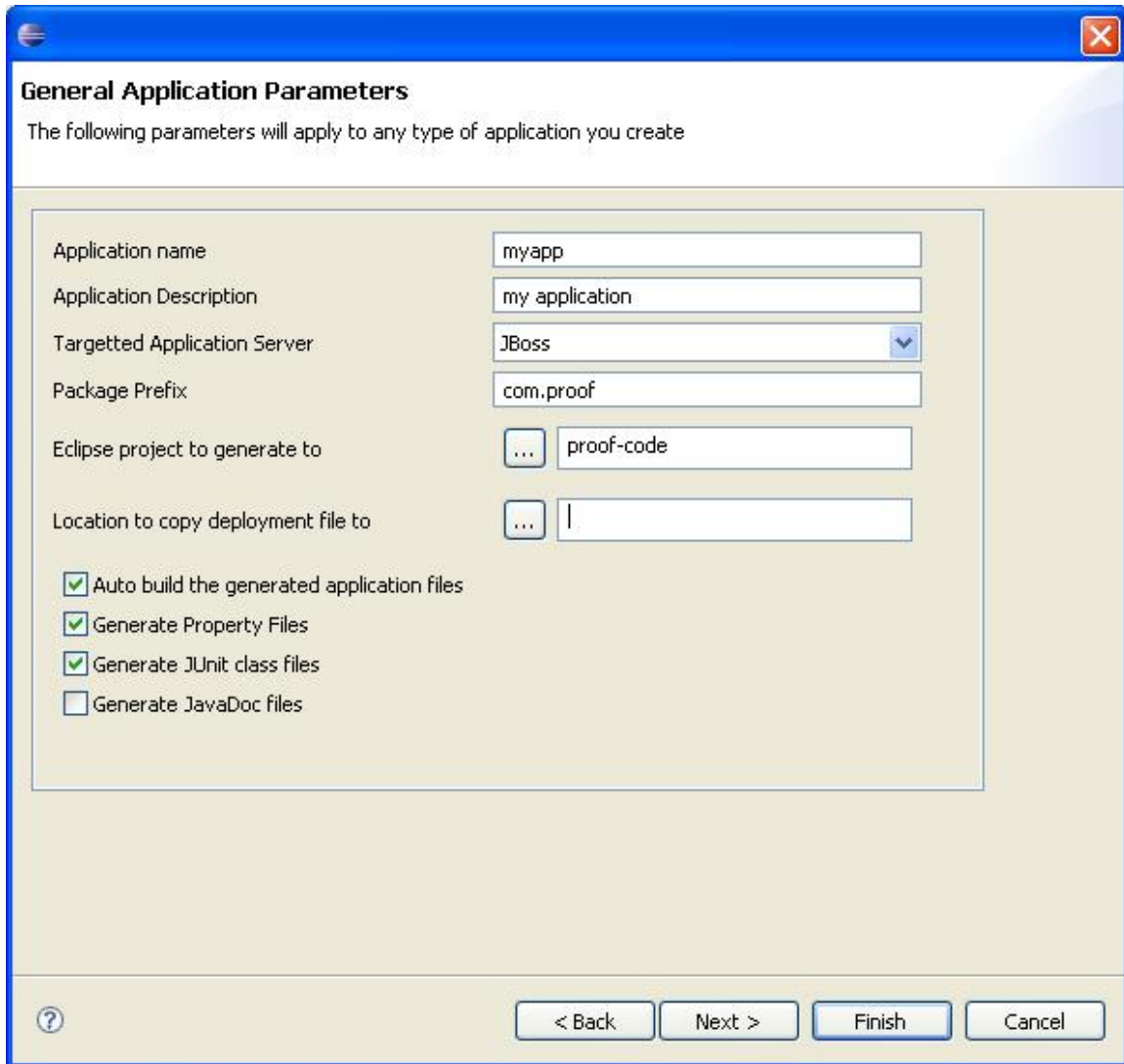
Before creating an AIB project, you should consider the following:

1. *Will you be loading an XMI/UML file, or will you be reverse engineering?*
2. *If you are reverse engineering, will you be doing so from existing POJOs or from an existing database?*
3. *Do you need an EJB Tier?*
4. *Do you want to support Web Services?*
5. *Do you need JUnit test classes?*
6. *What database will you need to support?*

Once you have consider each of these, you are ready to create an AIB project using the AIB Project Wizard.



Once you have provided a unique project name, click the *Next* button. This will reveal a screen to allow you provide general application parameters. It is suggested to provide a name in the “*Eclipse project to generate to*” field that is different than the project name you are using to create the project. Also, using the browse button for the “*Location to copy deployment file to*” field”, browse to the directory of deploy directory of your application server

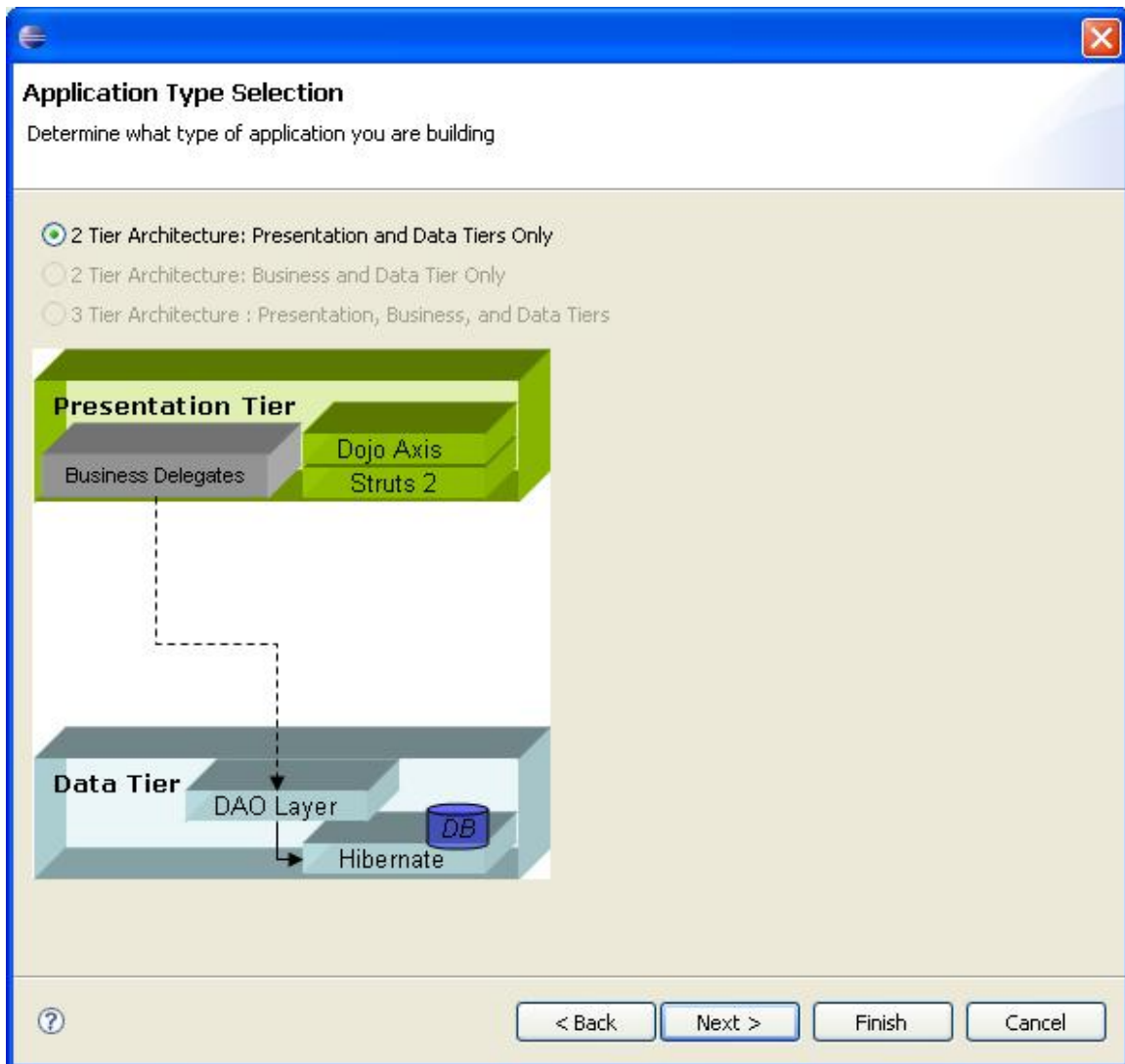


The dialog box is titled "General Application Parameters" and contains the following fields and options:

- Application name: myapp
- Application Description: my application
- Targetted Application Server: JBoss (dropdown menu)
- Package Prefix: com.proof
- Eclipse project to generate to: proof-code (with a browse button)
- Location to copy deployment file to: (with a browse button)
- Auto build the generated application files: ☒
- Generate Property Files: ☒
- Generate JUnit class files: ☒
- Generate JavaDoc files: ☐

At the bottom, there are four buttons: "< Back", "Next >", "Finish", and "Cancel".

Once you are finished with this screen, click the “*Next*” button to determine which Tiers your application will support.



**Note:** Only the "2 Tier Architecture: Presentation and Data Tiers Only" option is available during the evaluation period.

Click the "Next" button to enter your Hibernate core options.

**Hibernate O/R Mapping parameters**

Use this form to enter the necessary parameters to configure Hibernate to connect to your database

Core Options **Connectivity Types**

Database Type	MySQL
Transaction Manager	JBoss
Transaction Factory	JDBC
User ID	root
Password	69cutlass
JNDI User Transaction	java:comp/UserTransaction
Maximum Fetch Depth	1
JDBC Batch Size	20

Test Database Connection

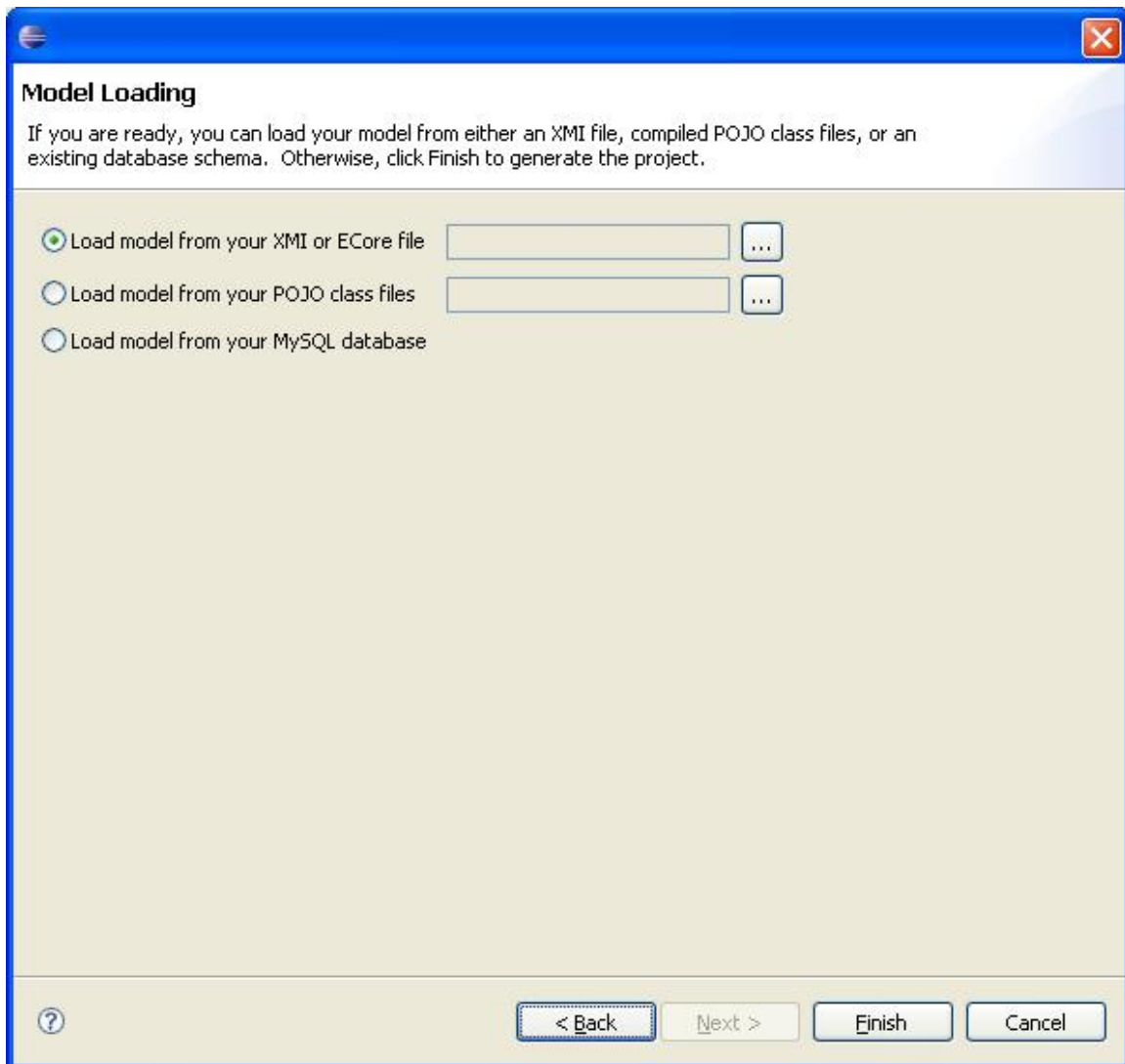
? < Back Next > Finish Cancel

It is **absolutely crucial** you get these settings, and the Connectivity Type settings, correct. This is typically the first settings to review when you are not able to successfully deploy and execute your application.

Click the "*Connection Types*" tab to determine how to connect to your database. Once you are satisfied with your settings, click the "*Test Database Connection*" to make sure your settings are correct.

**Note:** The AIB automatically loads the MySQL jar files during the project creation session. If you are using a different database, you will have to make its jar files available via the Eclipse IDE.

Click the "*Next*" to determine how to load your model.



At this point, you can load your model either from an existing XMI or ECore file, compiled POJO class files, or an existing database schema. For the sake of simplicity, it suggested you attempt to load an XMI file. Loading from compiled POJO class files or an existing database schema is a complicated “reverse engineering process”, often providing you with results you may not have expected due to the structure of the discovered model.

Click the “Finish” button to begin the model loading process and complete the project creation process.

If you decide to reverse engineer your model from either POJOs or a database schema, you will be presented with one final screen before the project creation is complete.



This screen is necessary for the following reason: The AIB is unable to determine what a multiple association actually points to. This is true of a `java.util.Collection` discovered in your POJO as well as a foreign key discovered during the database schema parsing process. This screen allows you to indicate what type of Class should be associated with a collection discovered within another a Class.



## What Next?

From here on, simply follow the Proof of Concept steps above, starting with *Hibernate Properties*. Good luck!!

## Proof of Concept FAQ

### What is Hibernate, and where can I learn more about Hibernate?

Hibernate is a powerful, ultra-high performance object/relational persistence and query service for Java. Hibernate lets you develop persistent classes following common Java idiom - including association, inheritance, polymorphism, composition and the Java collections framework. The Hibernate Query Language, designed as a "minimal" object-oriented extension to SQL, provides an elegant bridge between the object and relational worlds. Hibernate also allows you to express queries using native SQL or Java-based Criteria and Example queries. Hibernate is now the most popular object/relational mapping solution for Java. Importantly, Hibernate is Free Software.

Everything you ever wanted to know about Hibernate can be learned at <http://www.hibernate.org>.

### Where can I find log output?

The AIB generates a single log4j XML file used to configure the different log outputs, and places this file (*log4j.xml*) in the *\properties* directory. By default, the log4j.xml file contains the following configured Log4J loggers:

Log4J Logger Name	File Output
poc	rm_framework_log4j.log
realMethods	rm_framework_log4j.log
struts	struts.log
hibernate	hibernate.log
axis	axis.log

Each of these files will be located in the directory from which you launched your application server.

*axis.log is only generated when the Delegate/Web Services code generation option has been selected in the AIB.*

## I cannot run ANT using the generated build.xml file?

Assuming you have downloaded and installed [Jakarta Ant](#), edit the *build.xml* file and read the notes located at the beginning of the file.

```
=====
NOTE:  Incorporate the hibernate task if this is the
       first build, or the schema/model has changed
```

```
ant -DHibernate=true
=====
```

```
=====
NOTE:  The following global properties should
       be modified to reflect your system configuration
rm.home : realMethods Installation Root Directory
j2ee.home : location of the J2EE installation
db.jars : vendor db driver jars if using Hibernate
deploy.dir : location where to copy the app. EAR file
=====
```

## Why am I having issues reading or writing data to my database?

Double check the following properties in the generated *\properties\hibernate.cfg.xml* file:

```
<property name="hibernate.connection.url">
    jdbc:microsoft:sqlserver://DB:1433;SelectMethod=Cursor
</property>
<property name="hibernate.dialect">
    net.sf.hibernate.dialect.SQLServerDialect
</property>
<property name="hibernate.connection.driver_class">
    com.microsoft.jdbc.sqlserver.SQLServerDriver
</property>
<property name="hibernate.transaction.manager_lookup_class">
    net.sf.hibernate.transaction.WeblogicTransactionManagerLookup
</property>
<property name="hibernate.connection.password">
    johndoe
</property>
<property name="hibernate.connection.username">
    letmein2
</property>
```

### **Can I load the POC's XMI file into my UML tool?**

All major UML tool vendors (Rational (IBM), TogetherSoft (Borland), Poseidon, MagicDraw, etc.) have the capability to import the `\model\poc.xmi` file.

### **Why did the POC fail to deploy on my application server?**

The POC has been successfully deployed on all major application servers without having to make any changes to the generated source code and application configuration files. Please refer to the application deployment guidelines provided by your application server vendor, or refer to the "*App. Server Specific Deployment Instructions*" within the *Developer's Guide*.