1. Why and How

1. Download These Instructions

If you wish to have an offline copy of these instructions, please download the PDF version of this Installation Manual by clicking here.

2. Licensing in FuGE

The FuGE STK is licensed under the MIT license. This license covers the FuGE-OM and all code within the STK generated from it. Full details are available from the FuGE website .

3. Licensing in SyMBA

SyMBA is licensed under the Lesser GNU Public License (LGPL). However, SyMBA is also based on the FuGE Hibernate STK. All unmodified sections of the FuGE Hibernate STK retain the original MIT license statement. However, most aspects of the STK were modified in the creation of SyMBA. We still acknowledge the original creators and licensing of the FuGE Hibernate STK, but retain rights under the LGPL on those modifications made during SyMBA development.

4. Origins

SyMBA is currently built using the FuGE Version 1 Object Model. However, SyMBA was originally built directly from the FuGE Milestone 3 STK. This already used Maven 2 and AndroMDA to build a number of auto-generated files from the FuGE Object Model (FuGE-OM). However, a number of additional modules were added over time to form SyMBA, including:

- Modifications to the existing FuGE-OM to allow more complex versioning within the database for Identifiable objects
- LSID webservices (one to create LSIDs, and another to resolve them)
- Web Interface for users to deposit data and metadata into the SyMBA database
- Parser for OBI , from OWL format into a FuGE OntologyCollection, providing a developer-specified level of granularity
- JAXB2 codebase created from the FuGE XSD (itself autogenerated from the FuGE-OM)
- A large number of canned searches used by the web interface for searching the database

Many of these things were later incorporated into the FuGE Milestone 3 STK, and in July 2008,

into the newly-created FuGE Version 1 Hibernate STK. However, the web interface and LSID implementation are SyMBA-specific, as well as many of the searches, remain within the scope of SyMBA.

SyMBA is now built on top of the FuGE Version 1 Hibernate STK. However, while it is the intent of the FuGE developers that FuGE be extended (without modification), SyMBA needed to make changes to the OM itself in order to allow the more complex versioning. This meant that the FuGE-OM itself needed to be changed. This set of documentation identifies all changes that were made to the FuGE Version 1 Hibernate STK in order to bring it up to the capabilities required for SyMBA. In considering these changes, it is useful to keep in mind that all of these only result in a single addition to the standard FuGE XSD. This addition (the "endurant" attribute of all Identifiable objects) can be safely REMOVED from any exported XML file to make the resulting FuGE-ML completely compliant with the FuGE standard.

The following pages outline all changes made to the FuGE-OM found within the FuGE Version 1 Hibernate STK. Further changes to the FuGE STK will be propogated across to the SyMBA project.

5. Renaming of the UML file

The FuGE-OM UML file has been renamed from FuGE-v1-profile.mdzip to SyMBA-FuGEv1.mdzip to emphasize the changes made to the FuGE-OM for SyMBA.

6. The net.sourceforge.symba package within the FuGE-OM

Except for a couple of specifically-defined instances, all changes made to the FuGE-OM are contained in the net.sourceforge.symba package within the FuGE-v1-profile.mdzip file. This file lives inside SyMBA's symba-mda/src/main/uml . Therefore, the first change to the FuGE-OM was to create this package.

7. Introduction

Some aspects of the FuGE-OM do not have any code that needs to be changed, but restrictions need to be placed on the usage of that code within SyMBA. This section illustrates those restrictions.

8. net.sourceforge.fuge.EntityService

8.1. Addition of an Audit item within the Audit Trail

The FuGE Hibernate STK has two options for determining if an item should be annotated with an additional Audit item when it is loaded into the database.

- net.sourceforge.fuge.addDbAuditTrail within fuge-hibernate.properties
- addAuditInfo as a boolean argument for save, create, or update

The first option, set via a properties file, overrides a "true" value passed from the calling method with whatever the value in the properties file is. This is meant for uses such as unit testing, where you want the data that goes into the database to exactly match the data coming out, to ensure that all information is being passed correctly.

However, it is important that, when SyMBA is running in production mode, the audit information is always added to Identifiable objects (SyMBA does not require its addition to Describable objects that are not Identifiable). This is because the audit date is the attribute compared when retrieving the latest version of an object. Therefore you must not override the true value from the net.sourceforge.fuge.addDbAuditTrail property for production use. Further, when adding items to the database, please ensure that you use one of the varieties of save() or create() within the EntityService that will add an audit trail item to the object.

9. net.sourceforge.fuge.EntityService

The EntityService of the FuGE Hibernate STK is left alone except the following modifications.

9.1. Create, Save and Update

As SyMBA deals with versioning within its code, it is very important that no existing object in the database be deleted or modified. As the standard implementation of EntityService allows for the update of rows in the database, this feature must be disabled in SyMBA. This could be done by deleting the update() methods within EntityService, but the SyMBA developers wish to retain as much possibility of compatibility with pre-existing FuGE Hibernate STK code, as well as change as little of the Hibernate STK FuGE-OM as possible. Therefore save() and update() methods have been re-implemented. If any Identifiable object is passed to save() that already has a database ID, an EntityServiceException will be thrown. If any Describable object is passed to save() that already has a database ID, that database ID will be set to null and a new Describable object created. If update() is used at all, an EntityServiceException will be thrown.

For Identifiable objects, in order to ensure that you don't constantly pass objects to save() or create() that have already existing database IDs in the database, you may use net.sourceforge.symba.mapping.hibernatejaxb2.DatabaseObjectHelper 's assignAndSave() method. This method overwrites the LSID identifier of an Identifiable object, and sets the database

ID to null.

Please note that this is NOT a change within the FuGE-OM, but instead a change that only needs to be implemented at the level of the EntityServiceImpl class.

9.2. Addition of Dependency to Endurant

In order for the EntityService (which handles all create and save work for the project) to be able to create and save Endurant objects, a link was created between these two classes. There is now a dependency starting at EntityService and ending at Endurant.

10. net.sourceforge.fuge.common.Identifiable

10.1. New Association to net.sourceforge.symba.versioning.Endurant

The main addition to the FuGE-OM to generate SyMBA is the addition of an association from Identifiable to Endurant. This allows Identifiable objects to be grouped together, and therefore provide a history for FuGE objects. This change to the FuGE-OM is then propogated through into the auto-generated code. To examine this change, please open the "Endurant" class diagram within the SyMBA-FuGEv1.mdzip file stored in the symba-mda/src/main/uml directory.

11. Additional Lazy Initializations Set to False

- net.sourceforge.fuge.common.protocol.ParameterValue : set its association to Parameter (parameter) to lazy = false.
- net.sourceforge.fuge.common.protocol.ParameterValue : set its association to Parameter (parameter) to lazy = false.
- net.sourceforge.fuge.common.measurement.Measurement : set its association to OntologyTerm (unit) to lazy = false.
- net.sourceforge.fuge.common.measurement.Measurement : set its association to OntologyTerm (dataType) to lazy = false.
- net.sourceforge.fuge.bio.material.GenericMaterial : set its association to OntologyTerm (materialType) to lazy = false.

12. New Queries

This section describes the set of new queries that were added to specific entities within the FuGE-OM. They do NOT modify the behavior, attributes, XSD, or associations of any of the FuGE model. They simply provide more searches than are available in the standard FuGE Hibernate STK. However, such queries belong within the entities they are searching, and as such they modify the FuGE-OM entities.

This is just a summary of the added queries. For documentation on the queries themselves, please see the UML.

12.1. Queries over Identifiable

- getForDate(String identifier, Date date)
- getWithEndurantForDate(String endurantIdentifier, Date date)

12.2. Queries over FuGE

- countLatest()
- getSummaries()
- getSummariesWithPartialName()
- getSummariesWithContact()
- getSummariesWithOntologyTerm()

12.3. Queries over ExternalData

- countRealData()

12.4. Queries over GenericProtocolApplication

- getLatest()
- getLatestDummies()
- getLatestMaterialTransformations()

12.5. Queries over OntologyTerm

- getLatestUnsourced()
- getLatestWithSource()
- getDistinctTermInfo()

12.6. Queries over GenericMaterial

- getLatest()
- getLatestDummies()

13. Changed Methods

Due to the addition of the Endurant class, and the association to this class from the Identifiable class, a number of base methods must be modified. Specifically, any helper methods that create new, empty, Identifiables in the database must ensure that an Endurant is assigned to that identifiable and loaded in the database prior to returning the newly-created object. Below are a list of methods which, therefore, now deviate from the standard method in the FuGE Version 1 Hibernate STK.

- net.sourceforge.symba.mapping.hibernatejaxb2.DatabaseObjectHelper getOrCreate() . This method now also creates a new Endurant if it also needs to create a new Identifiable. As it loads the Endurant into the database, it also may optionally pass a performer to assign to the audit trail. (This means it becomes an overloaded function.) It also means there is an additional argument - the Endurant.identifier . If this is null, it will create a new Endurant, but if present, then will either retrieve the existing Endurant or make a new one using that ID.
- net.sourceforge.symba.mapping.hibernatejaxb2.DatabaseObjectHelper createEndurantAndIdentifiable() . A new method that will create both a new Endurant and Identifiable.
- net.sourceforge.symba.mapping.hibernatejaxb2.DatabaseObjectHelper getOrCreateEndurant() . A new method that will either retrieve the Endurant from the database or create a new one
- net.sourceforge.symba.mapping.hibernatejaxb2.DatabaseObjectHelper createAndLoadEndurant() . A private new method that will create a new Endurant and load it into the database.
- net.sourceforge.symba.mapping.hibernatejaxb2.DatabaseObjectHelper getLsid() Helper method so the user doesn't have to create their own instance of the Identifier maker.
- net.sourceforge.symba.mapping.hibernatejaxb2.DatabaseObjectHelper save() . New core logic has been added (as per the intended spec of this method) so that assignAndSave will be run if there is already a database id in the object.
- net.sourceforge.symba.mapping.hibernatejaxb2.DatabaseObjectHelperTest . Modified to use Endurants.
- net.sourceforge.symba.mapping.hibernatejaxb2.StoreAndRetrieveTest . Modified to use Endurants.
- net.sourceforge.symba.mapping.hibernatejaxb2.GenericSoftwareMappingHelperTest . Modified to use Endurants.
- net.sourceforge.symba.mapping.hibernatejaxb2.helperIdentifiableMappingHelper.marshal() Modified to set the Endurant identifier.

- net.sourceforge.symba.mapping.hibernatejaxb2.helperIdentifiableMappingHelper.generateRandomXML() Modified to set the Endurant identifier.

14. Changed Class Names

Due to the FuGE Hibernate STK changing all instances of URI to Uri (The "URI" entity is problematic for Spring, as any class name whose second letter is upper case can cause problems. The best way to solve it is to change the name of this entity to "Uri"), the resulting XSD and JAXB2 code is different from the XSD STK's equivalents. Any place in the code from the Hibernate STK (which uses the XSD STK's official fuge-jaxb2 jar rather than creating one itself) that uses URI had to be changed to URI.

15. Added XSD creation to SyMBA from the FuGE XSD STK

Because we've added the Endurant class to the OM, all Identifiable objects now have an additional attribute that is mandatory when loading into the database: "endurant". This is represented in the XML as an Endurant_ref, and the XSD and associated JAXB2 code must therefore be made afresh for SyMBA.

16. Template Changes

Within the symba-mda subproject are custom Velocity cartridge templates taken from the FuGE XSD STK. These templates are used to generate the correct FuGE XSD. The list below details the changes made to this template for SyMBA.

- .replaceAll('net.sourceforge.fuge','FuGE') Any time where the fullyQualifiedClassName variable was used in the template, the replaceAll method shown here was added. In the creation of both the EJB3 and Hibernate STKs, the FuGE package name within the FuGE-OM was replaced with "net.sourceforge.fuge" so that more common Java package naming schemes could be used. This had the knock-on effect of changing the way the XSD package names were shown. In order to bring a closer equivalence to the SyMBA-generated XSD and the official FuGE XSD, the following replacement was done.
- Line 72: Replace #if ($attribute.hasExactStereotype("XmlAttribute")==false) with #if ($attribute.hasExactStereotype("XmlAttribute")==false && !($attribute.name == "id")) . This is important when building a database with AndroMDA as well as the XML Schema. Otherwise, when generating a database it decides it needs a unique key on every attribute. Thanks to Andy Jones for helping with this bug. If just building the XML Schema, this problem doesn't crop up.

17. Introduction

Some aspects of the way SyMBA behaves have been modified for this new release. These general behavior changes are described here.

18. Endurant classes

18.1. Endurant is now concrete rather than abstract

In the FuGE Milestone 3 (FM3) implementation of SyMBA, the Endurant class was abstract, and concrete sub-classes were made that mirrored the FuGE-OM structure of concrete implementations of Identifiable. However, it quickly became clear that this multitude of SyMBA Endurant implementations were unnecessary. Therefore, the opportunity of upgrading to FuGE Version 1 (FV1) meant a good chance to solve this situation.

Therefore, with the FV1 version of SyMBA, there is now only one Endurant class. It is still Describable, but it is concrete and all Identifiable classes contain a reference to this concrete class. It is a much simpler, cleaner system.

18.2. Implications for LSIDs

This means that all new LSIDs generated within SyMBA for the Endurant.identifier attribute will have a namespace of "Endurant" rather than the name of the concrete implementation of Endurant that was present in FM3. Nothing in the core will break if you still use the old LSIDs in the new SyMBA, however you may have to modify how the LSID resolver works to cope with the old-style LSID namespaces.

19. Additions to the Object Model within the SyMBA namepace

19.1. net.sourceforge.symba.versioning.Endurant

A new entity, net.sourceforge.symba.versioning.Endurant, has been created within the net.sourceforge.symba namespace within the FuGE-OM. To examine this entity, please open the

"Endurant" class diagram within the SyMBA-FuGEv1.mdzip file stored in the symba-mda/src/main/uml directory.

The Endurant entity is a concrete instance of Describable, as all entities utilizing the FuGE-OM are meant to be. However, in practical terms, none of the Describable aspects of the Endurant class are currently in use within SyMBA. The only attribute of this entity is an "identifier" attribute. Within SyMBA, this is always an LSID. Each Identifiable object within the FuGE-OM now has an association (named "endurant") to one of these objects. The Endurant is a grouping object that links different Identifiables of the same concrete class.

19.2. net.sourceforge.symba.service.SymbaEntityService

net.sourceforge.symba.service.SymbaEntityService is a service class similar in design to net.sourceforge.fuge.service.EntityService . Its purpose is to liase with the persistence layer, and perform additional functions that require connections to the database. While EntityService provides basic functions relating to saving, creating and updating objects in the database, the SymbaEntityService is more geared to complex or specialized search queries over the database. This class is described in the SymbaEntityService class diagram in the UML.

19.3. net.sourceforge.symba.service.SymbaEntityServiceException

net.sourceforge.symba.service.SymbaEntityServiceException is the exception class that is thrown if there is any problem when running methods from SymbaEntityService. This class is described in the SymbaEntityService class diagram in the UML.

19.4. net.sourceforge.symba.mapping.hibernatejaxb2.xml.WorkflowUnmarshaler

This file is taken directly from the original SyMBA code and updated for FuGE version 1.

19.5. net.sourceforge.symba.mapping.hibernatejaxb2.xml.UnmarshalWorkflow

This file is taken directly from the original SyMBA code and updated for FuGE version 1.

20. Testing Within SyMBA

All new queries provided within SymbaEntityService have full unit tests. This means adding unit test classes within the fuge namespace, although these tests have been written specifically for SyMBA, and only exist within the SyMBA project.