Using JBoss for Tactical Space Operations

Kenneth Rabe

Agenda

- Introduction
- JBoss applications being used
- Migration from Hibernate 2.1.6 to 3.1
- Extending Hibernate Tools

Speaker

Kenneth Rabe

- Software Engineer at JPL
- Primary developer of most database interaction
- Lead developer for adaptation of Ensemble adaptation for Robonaut

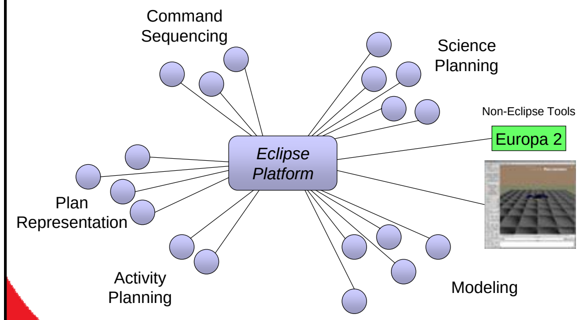
What is Tactical Operations software?

- Looking at recent data
- Finding interesting stuff
- Deciding what should be investigated
- Building the next set of activities

Customer-Tailored Development

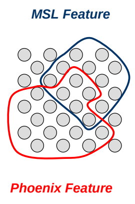
- Not all missions need the same capabilities
- Not desirable to give extraneous functionality
  - ✓ Unneeded functionality can be confusing
  - ✓ Not all functions are applicable
    - Image viewing is not important for missions that cannot generate images.
    - Orbital tracking of data is not relevant for missions without orbital assets.
  - ✓ Extra functionality can reduce stability

Ensemble Plugin Architecture



Ensemble Plugin Architecture

- Based on Eclipse plugins
- Use many core plugins for common functionality
- Custom plugins for adaptations



JBoss tools used

- Hibernate
- Hibernate Tools
- JBoss Application Server
- JBoss JMS

Hibernate

- Manages all database interactivity
  - ✓ No more messy JDBC
- All long term persistence
  - ✓ No longer filesystem based
- Database neutral
  - ✓ Postgres
  - ✓ MySQL
  - ✓ HSQL

Hibernate Tools

- Schema generation
  - ✓ Database neutral
  - ✓ Taken straight from hbm.xml files
- Code generation
  - ✓ Common behavior in all generated classes
  - ✓ No chance to mistype / introduce bugs
  - ✓ Taken straight from hbm.xml files

JMS

- Real-time collaboration
- Used in conjunction with database persistence to keep everyone in sync

Apache

- Serves data not installed with application
- Treat data browsing like a browser
  - ✓ Cache data locally
  - ✓ Fetch from a server

Servlets

- Some data is too large to pass to clients in realtime (MBs)
- Not all data is needed
  - ✓ Client typically interested in ~80 B

Migrating from 2.1 to 3.1

The Good

- Relatively straight forward to upgrade
- Eclipse organize imports works mostly
  - ✓ Does not work in bulk mode
- Standard support for MySQL innoDB tables
- Scrolling result sets

The Bad

- Relatively minor changes in API
  - ✓ Significant reuse of names
    - org.hibernate
    - org.hibernate.classic
  - ✓ Wrapped API delegation classes took some time to verify correct

The Ugly

- Hibernate tools completely changed
  - ✓ Previous source code generation was no longer functional
  - ✓ Took quite a while to get to a functional state (3.1 beta2)

Extending Hibernate Tools

Tool Configuration

```
<hibernatetool destdir="${generated-source.root}" >
      <configuration>
        <fileset dir="${source.root}">
          <include name="**/*.hbm.xml"/>
        </fileset>

  </configuration>
  <hbmtemplate
     templatepath="${workspace.dir}/${plugin}/vt“
    template="Pojo.vm"
    exporterclass="${plugin}.EnsembleHibernateExporter"
    filepattern="{package-name}/{class-name}">
  <property key="jdk5" value="true"/>
  </hbmtemplate>
</hibernatetool>
```

Conventions used

- Use both generated code and non-generated code
- Generated code is the superclass
- All custom code is in the concrete non-generated subclass
- Prefix generated code with Hbn_
- Never instantiate the generated class
  - ✓ Not always able to make the generated class abstract

Example mapping file

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping
        PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0/EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
  <class name="gov.nasa.ensemble.core.activityDictionary.Hbn_Choice"
    table="choice" discriminator-value="H">
    <meta attribute="class-description">Possible value for an Argument</meta>
    <meta attribute="scope-class">public abstract</meta>
    <meta attribute="implement-equals">true</meta>
    <id name="id" type="int" column="choice_ID">
        <meta attribute="use-in-equals">true</meta>
        <generator class="native" />
    </id>
    <discriminator column="discriminator" type="character"/>
    <property name="value" type="string" not-null="true"/>
    <many-to-one name="argumentDef" column="argumentDef_ID“
                 class="gov.nasa.ensemble.core.activityDictionary.Hbn_ArgumentDef" />
    <subclass name="gov.nasa.ensemble.core.activityDictionary.Choice"
              discriminator-value="S"/>
  </class>
</hibernate-mapping>
```

What gets generated

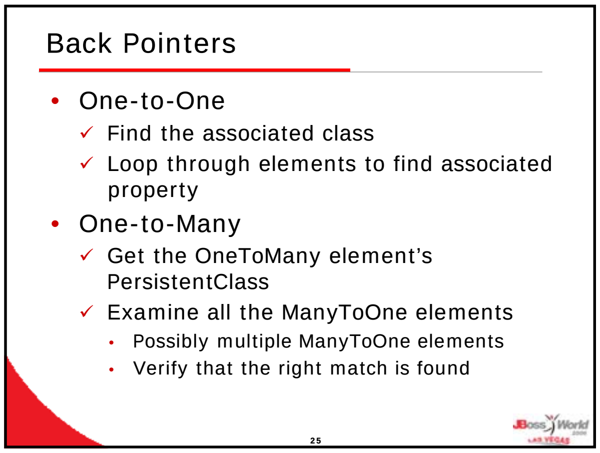
Default	Extended
• getFoo()	• addToFoo()
• setFoo()	• removeFromFoo()
• constructors	• automatic associations
• equals()	• notification
• hashCode()	• field enum
	• clone()
	• equals()
	• hashCode()

Methods generated for a List

```
public List<Choice> getChoices() {
   if (this.choices == null) {
      this.choices = new ArrayList<Choice>();
   }
   return this.choices;
}

public void setChoices(List<Choice> choices) {
   this.choices = choices;
}

public synchronized void addToChoices (Choice aChoice) {
   getChoices().add(aChoice);
   aChoice.setArgumentDef((ArgumentDef) this);
}

public synchronized boolean removeFromChoices (Choice aChoice) {
   if (!getChoices().contains(aChoice))
      return false;
   getChoices().remove(aChoice);
   aChoice.setArgumentDef(null);
   return true;
}

public synchronized void addToChoices (int index, Choice aChoice) {
   getChoices().add(index, aChoice);
   aChoice.setArgumentDef((ArgumentDef) this);
}
```

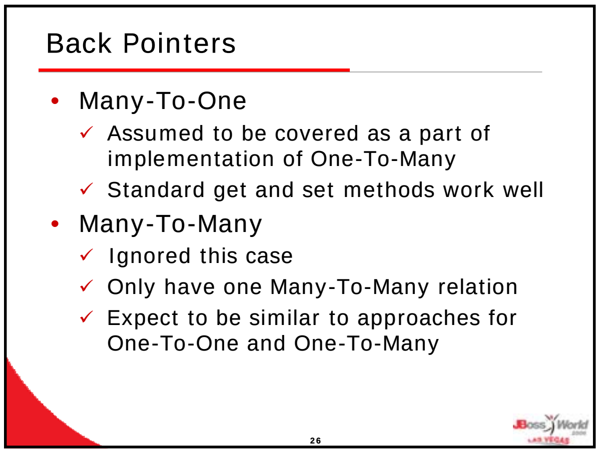
Relevant Files

- Velocity templates
  - ✓ Basic code that needs little/no processing
  - ✓ Property looping
- Cfg2JavaTool
  - ✓ Name munging
  - ✓ Entry point to your POJO class
- EntityPOJOClass
  - ✓ All complex code generation
  - ✓ Examining non-trivial properties



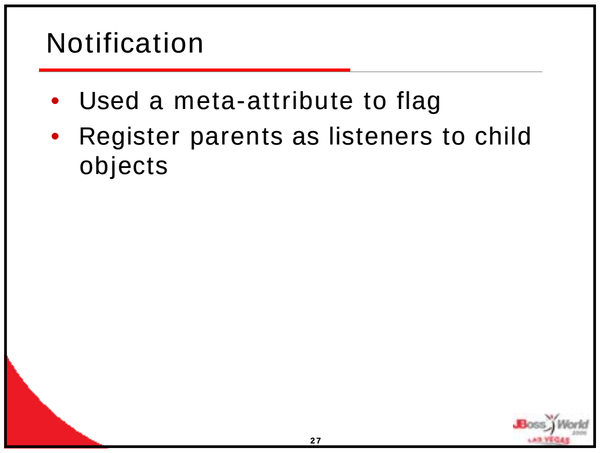
Back Pointers

- One-to-One
  - ✓ Find the associated class
  - ✓ Loop through elements to find associated property
- One-to-Many
  - ✓ Get the OneToMany element's PersistentClass
  - ✓ Examine all the ManyToOne elements
    - Possibly multiple ManyToOne elements
    - Verify that the right match is found



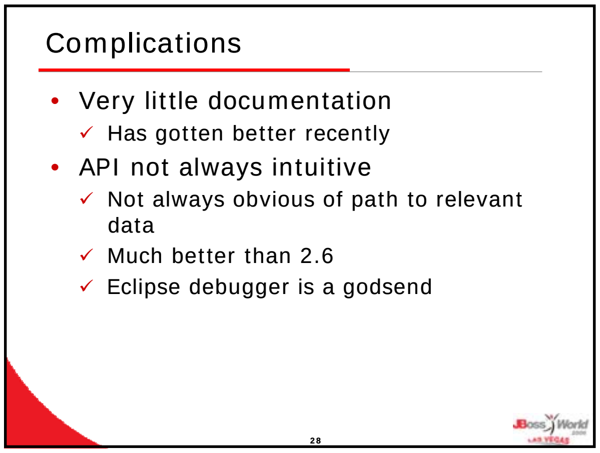
Back Pointers

- Many-To-One
  - ✓ Assumed to be covered as a part of implementation of One-To-Many
  - ✓ Standard get and set methods work well
- Many-To-Many
  - ✓ Ignored this case
  - ✓ Only have one Many-To-Many relation
  - ✓ Expect to be similar to approaches for One-To-One and One-To-Many



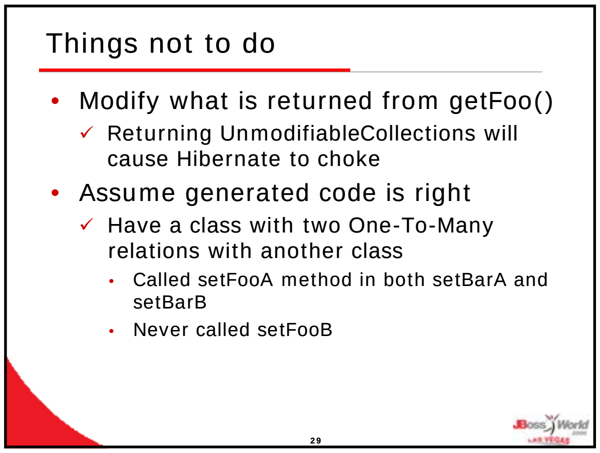
Notification

- Used a meta-attribute to flag
- Register parents as listeners to child objects



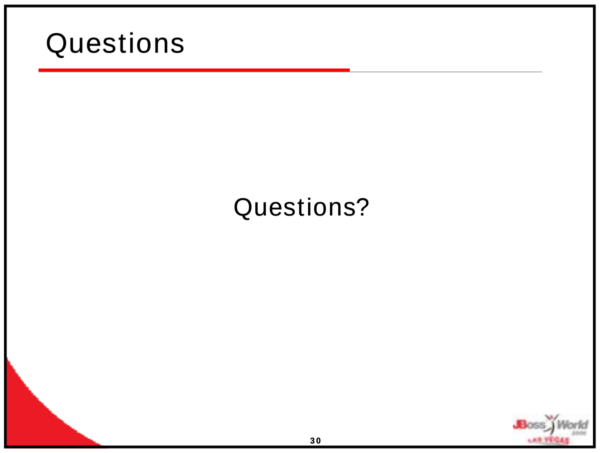
Complications

- Very little documentation
  - ✓ Has gotten better recently
- API not always intuitive
  - ✓ Not always obvious of path to relevant data
  - ✓ Much better than 2.6
  - ✓ Eclipse debugger is a godsend



Things not to do

- Modify what is returned from getFoo()
  - ✓ Returning UnmodifiableCollections will cause Hibernate to choke
- Assume generated code is right
  - ✓ Have a class with two One-To-Many relations with another class
    - Called setFooA method in both setBarA and setBarB
    - Never called setFooB



Questions

Questions?