J2EE for GLAST

A Lightweight Service Oriented Architecture for GLAST Data Processing

Matthew D. Langston
Stanford Linear Accelerator Center
November 23, 2004

Outline

1. Introduction to GLAST Data Processing
   - Major components
   - Requirements, constraints, resources, schedule
2. Proposed Solutions
   - Perl architecture (Perl scripts + CGI)
   - Classic J2EE (Java + EJB containers)
   - Lightweight container (Java)
3. Lightweight container solution
   - Container requirements specific to GLAST
   - Spring Framework
   - Transparent Object ⇔ Relational Database persistence (O/R Mapping)
4. Processing Pipeline 2.0
   - Status
   - Existing components
   - Moving data in and out of Oracle
   - XML-based pipeline configuration
   - Monitoring
4. Development process
   - Project management
   - Release Manager
     - builds, unit tests, documentation
   - extlib manager
   - Test Driven Development
   - Example
5. Dashboard: web front-end with Macromedia products
   - ColdFusion MX 6.1
   - Dreamweaver MX
   - FLEX
6. Conclusion

GLAST Data Processing

- Serve GLAST's data processing and infrastructure needs for 10+ years
- Major Components
  - Monitoring and Reporting
    - Data quality
    - Software quality (physics output, nightly builds, etc.)
    - Data processing, re-processing, simulation, etc.
    - Computing resources (server health, processing status, batch farm, NFS space, etc.)
    - Problem notification (email, pager, etc.)
    - Historical tracking of all of the above
  - Processing Pipeline
    - General purpose rule engine
    - Automate and manage simulation, reconstruction, builds, etc.
  - Data Server
    - General purpose query engine and data assembler
    - query physics event properties from ROOT data library and assemble into synthetic bite-sized pieces for individual analysis
- Implicit component: Framework and development approach
  - tying everything together
  - common enterprise services: security, persistence, transactions, pooling, remoting and web services, web-framework, job scheduler, email notification

Requirements

- 24x7 uptime
- 10+ year shelf life
- Support Linux and Windows Platforms
  - Many (but not all) components must run on both platforms
- Developed and maintained by small group (of order 5 people) of disparate talents (engineers, web developers, interested scientists)

Proposed Solutions

- Perl + CGI
  - Difficult to maintain
  - Limits involvement
  - SLAC Security concerns
  - Limited enterprise services
  - Limited tool and project support
- Classic J2EE (EJB)
  - Complex programming model
  - Restricted access to Java APIs
  - Monolithic
  - Difficult to test

Is there something in between?

XP mantra: “the simplest solution that can possibly work”

Lightweight Container

- "J2EE without EJB"
  - Part of emerging post-EJB consensus
  - Driven by practical Open Source benefits (not ideological ones)
- You program in Plain Old Java Objects (POJO)
  - Nothing fancy
  - Nothing new to learn
  - Easily testable
- Declaratively provides best parts of EJBs (and **only** those required by GLAST)
  - Transaction management
  - Security
  - Remoting
  - Cross cutting concerns in general
- No API
  - Not a class library
  - No inheritance
  - Non-invasive
- No restrictions on use of 3rd party APIs
  - Full access to richness of Java/J2EE open source products (JAS, Tomcat, Hibernate, etc.)
  - Full access to commercial products (ColdFusion MX, GLUE)
- Light footprint
  - Useful in standalone applications: `public static void main(String[] args)`
  - Web container (for example, Tomcat)
  - Full blown J2EE container

Spring Framework

- Mission Statement (from http://www.springframework.org/)
  - J2EE should be easier to use
  - It's best to program to interfaces, rather than classes. Spring reduces the complexity cost of using interfaces to zero.
  - JavaBeans offer a great way of configuring applications.
  - OO design is more important than any implementation technology, such as J2EE.
  - Checked exceptions are overused in Java. A framework shouldn't force you to catch exceptions you're unlikely to be able to recover from.
  - Testability is essential, and a framework such as Spring should help make your code easier to test.

Spring Framework

- From the Spring manual (180 pages)
  - Bean Factory
    - Java beans replace EJB
  - Aspect Oriented Programming
    - “Configure when you can, program when you must”
    - Transactions
    - Security
  - Data Access
    - JDBC
    - Object Relational Mapping (Java Beans ⇔ RDBMS)
  - Transaction Management
  - Security Framework
    - Never touch the password
  - Web Framework
    - Beans as Servlets
  - Java Message Service
    - Distributed Asynchronous and Synchronous Events
  - Remoting
    - Web Services (SOAP + many others)
  - Sending Email
  - Job Scheduling

Spring Framework

- Configure Java beans using setters in simple xml configuration file

```
<bean id="taskDao" class="glast.TaskDao"/>

<bean id="pipeline" class="glast.Pipeline">
    <property name="taskDao"><ref local="taskDao"/></property>
</bean>
```

```
public class Pipeline {
    private TaskDao taskDao;

    private TaskDao getTaskDao() {return taskDao;}
    private void setTaskDao(TaskDao taskDao) {this.taskDao = taskDao;}

    public void run(Task task) {/*submit Task to batch farm*/}
```

Spring Framework

- The container is a Java bean factory

```
public static void main(String[] args) {
    ApplicationContext applicationContext =
    new ClassPathXmlApplicationContext("/grits/gino/applicationContext.xml");
    Pipeline pipeline = (Pipeline) applicationContext.getBean("pipeline");

    TaskDao taskDao = pipeline.getTaskDao();
    List taskList = taskDao.loadAll();
    Iterator iterator = taskList.iterator();
    while (iterator.hasNext()) {
        Task task = (Task) iterator.next();
        String taskName = task.getName();
        if (taskName.startsWith("dc1")){
            taskDao.delete(task);
        }
        else {
            task.setName("dc2" + taskName);
            taskDao.saveOrUpdate(task);
        }
    }
}
```

1. Ask Spring for a Pipeline.
2. Spring creates and returns a Pipeline configured to talk to Oracle.
3. Both “singleton” and “create on demand” beans are supported (the latter being almost always what you want).

Requirement check

- Do we need a bean factory?
- Yes

- A bean factory removes configuration from code - all configuration stored in configuration files
  - Application objects are “wired up” using simple bean setters
  - All GLAST software **and** all 3rd party libraries are configured identically
  - No proliferation of proprietary configuration files
  - Database connection settings, connection pool size, LSF queues, etc.
- Out-of-the-box implementations for
  - `FileSystemApplicationContext`
  - `ClassPathApplicationContext`
  - `XmlWebApplicationContext` (`web.xml` for Tomcat, ColdFusion MX, etc.)
- Don’t have to use JNDI (although you can)
- Objects remain loosely coupled
- Objects are easy to test

Spring Framework

```
public class Task {
    private Long id;
    private String name;

    private Task() {}

    public Long getId() {return id;}
    private void setId(Long id) {this.id = id;}

    public String getName() {return name;}
    public void setName(String name) {this.name = name;}
}
```

1. Task is a simple **POJO** Java bean
2. Property **id** is primary key (set by Oracle; never set in Java)
3. Private constructor – bean can only come from Oracle; never created in Java.

```
public interface TaskDao {
    public List loadAll() throws DataAccessException;
    public Task loadTask(Long id) throws DataAccessException;
    public void saveOrUpdate(Task task) throws DataAccessException;
    public void delete(Task task) throws DataAccessException;
}
```

1. Task DAO is an interface (JDBC? Hibernate?)
2. Spring translates all vendor-specific **checked** exceptions into generic **unchecked** exceptions.

Requirement check

- Do we need unchecked data access exceptions?
- Yes

- We currently use at least two database vendors
  - Oracle
  - MySQL
  - More may follow? (Richard Mount's in-memory terabase)
- Spring translates vendor-specific error codes (in JDBC `SQLException`) into specific DataAccessExceptions.
  - For example, `TypeMismatchDataAccessException`
- Spring translates exceptions from different data access strategies (for example, JDBC, Hibernate, etc.) into a generic DataAccessException hierarchy.
- GLAST code stays decoupled from specific database vendors and specific data access strategies
  - Easy maintenance and allowing migration
  - Use case: wire up a Goddard Pipeline

Spring Framework

Database Transactions

Arguably the best part of EJB was CMT (Container Managed Transactions)

- Declarative
- JTA (span multiple databases)
- Remote Transaction Propagation (span multiple JVMs)

Complete but heavy-handed

Spring provides declarative transactions to POJOs

- Specified in configuration file (the *lightweight container* way)
- **or** using source-level meta attributes (ala .NET, *jakarta-commons attributes* and JDK 1.5)
- Pluggable transaction strategies
- Can use JTA, but don't have too

Common to all Transaction Managers

- Propagation behavior
  - required
  - supports
  - mandatory
  - requires new
  - not supported
  - never
- Isolation level
  - default
  - read uncommitted
  - read committed
  - repeatable read
  - Serializable
- Timeout
- Read-only

Spring Framework

Same as before

```
<bean id="pipelineTarget" class="glast.Pipeline">
    <property name="taskDao"><ref local="taskDao"/></property>
</bean>

<bean id="transactionManager"
    class="org.springframework.transaction.jta.JtaTrqansactionManager"/>

<bean id="pipeline"
    class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean">
    <property name="transactionManager"><ref local="transactionManager"/></property>
    <property name="target"><ref local="pipelineTarget"/></property>
    <property name="transactionAttributes">
        <props>
            <prop key="saveOrUpdate*">PROPAGATION_REQUIRED</prop>
            <prop key="delete">PROPAGATION_REQUIRED</prop>
            <prop key="*">PROPAGATION_REQUIRED,readOnly</prop>
        </props>
    </property>
</bean>
```

Instantiate transaction manager

1. Plug in your POJO
2. Plug in your Transaction strategy
3. **Bam**. Pipeline now protected by Transactions

Simple patterns matching member functions (**Perl-style regxps** also supported)

Spring Framework

- **Important**: Java code did not change. Transactions were specified declaratively in configuration file.

```
public static void main(String[] args) {
    ApplicationContext applicationContext =
    new ClassPathXmlApplicationContext("/grits/gino/applicationContext.xml");
    Pipeline pipeline = (Pipeline) applicationContext.getBean("pipeline");

    TaskDao taskDao = pipeline.getTaskDao();
    List taskList = taskDao.loadAll();
    Iterator iterator = taskList.iterator();
    while (iterator.hasNext()) {
        Task task = (Task) iterator.next();
        String taskName = task.getName();
        if (taskName.startsWith("dc1")){
            taskDao.delete(task);
        }
        else {
            task.setName("dc2" + taskName);
            taskDao.saveOrUpdate(task);
        }
    }
}
```

All database access automatically enlisted in Transactions

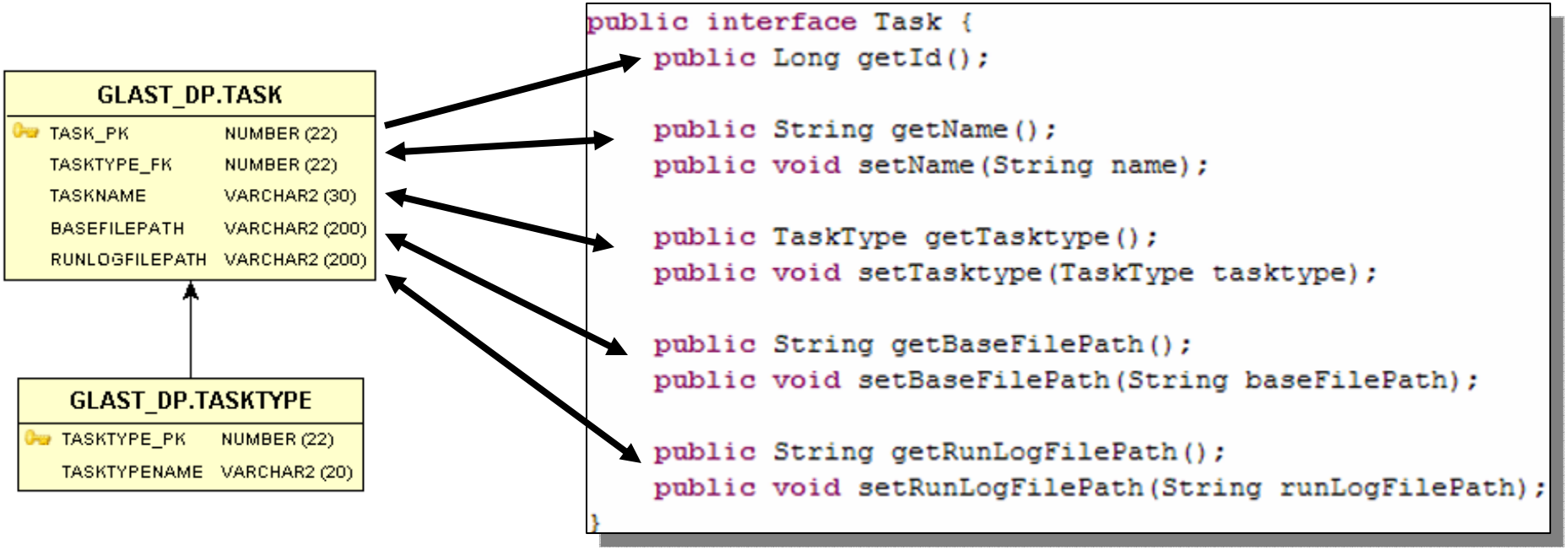
Requirement check

- Do we need Transactions? Do we need declarative transactions?
- Yes and Yes

- **Use case**: Editing Pipeline configurations (using web interface)
  - User think-time easily exceeds connection time boundaries.
  - Data is disconnected and may have become inconsistent.
  - Transactions protect data integrity.
- **Use case:** Pipeline XML file upload utility
  - Makes tremendous number of changes to the database all at once
  - many deletes, inserts and updates
- Transactions are a cross-cutting concern
  - Should therefore not be done programmatically (besides, none of us are probably qualified anyway)
  - Applying transactions to POJOs in a configuration file keeps code from changing and eases maintenance.

Database Access

- Programmatic data access
  - database data ➔ Java beans
  - Do something useful with beans
    - run a Task
    - create web report
    - edit configuration
    - etc.
  - Java beans ➔ database
- Web-page data access
  - Reports (large lists of information)
    - Failed runs
    - System tests
    - Time histories
  - Form editing (Pipeline configuration)



Database Access

Programmatic data access

- JDBC
  - Powerful API for working with relational databases at SQL level (similar to Perl DBI)
  - Bloated and repetitive infrastructure code (transactions, exceptions, etc.)
  - Manual bean get/set round trips
  - Mapping not done declaratively (done programmatically)
- iBATIS SQl Maps
  - Simple xml “mapping file” for Java beans (declarative mapping)
  - Retain full power of SQL
  - Pluggable cache strategies
  - Change/dirty detection and done manually (same for JDBC)
- Hibernate
  - Simple xml “mapping file” (declarative mapping)
  - This layer over JDBC
  - Doesn’t hide underlying RDBMS
  - Transparent persistence of Java beans and their complex object graphs
  - Disconnect and re-associate persistent objects (ala .NET’s disconnected Dataset)
  - Pluggable cache strategies
- JDO
  - Generic object persistence
  - Agnostic of underlying data store (can use RDBMS, OODBMS, etc.)
  - Does not support relational concepts like joins, aggregate functions, etc.
  - inability to re-associate persistent object with new transaction
- EJB

- Web-page data access
  - Access data from any of the above methods (JSP and ColdFusion MX)
  - JSP: `<sql:query …>`
  - ColdFusion MX: `<cfquery …>`

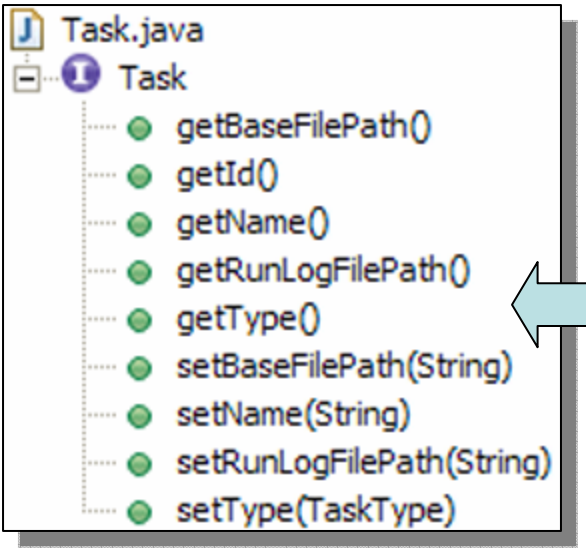
Which Data Access Strategy?

- The simplest solution that can possibly work
  - For web based reports: `<cfquery>`
    - Paging through thousands of records 20 rows at a time (like Google)
  - For simple web forms: `<cfquery>`
  - For complex web forms: Java beans + Hibernate
    - Data integrity
  - Processing Pipelines: Hibernate
    - object graphs
    - High I/O
    - Multiple connections
    - Aggressive caching

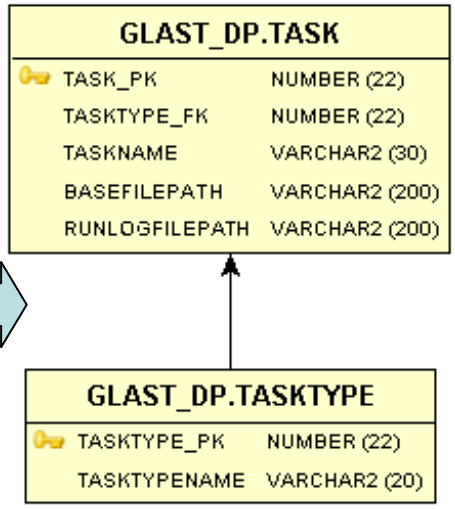
Hibernate

- Create simple “mapping” file
  - Specify which Java bean properties map to which database columns
  - Java bean is never aware it is persistent
    - Configuration done external to code
  - Designed to support legacy databases
    - Database does not have to change
  - Can create schemas on demand
    - Very useful for unit tests

Most important takeaway: Java bean and Database are completely decoupled – neither have to change.




```
<hibernate-mapping>
    <class name="glast.Task">
        <id name="id"/>
        <property name="name"/>
        <many-to-one name="type"/>
        <property name="datasetBasePath"/>
        <property name="runLogPath"/>
    </class>

    <class name="glast.TaskType">
        <id name="id"/>
        <property name="name"/>
    </class>
</hibernate-mapping>
```



Hibernate

TASKTYPE_PK	TASKTYPENAME
1	SimReconDigi
21	test
62	Analysis
41	Reconstruction
42	Digitization
61	Report
63	SystemTest
64	Simulation
66	Reprocessing
67	Conversion

```
public final class TaskType extends Enum {
    public static final TaskType SIM_RECON_DIGI = new TaskType("SimReconDigi");
    public static final TaskType TEST           = new TaskType("test");
    public static final TaskType ANALYSIS       = new TaskType("Analysis");
    public static final TaskType RECONSTRUCTION = new TaskType("Reconstruction");
    public static final TaskType DIGITIZATION   = new TaskType("Digitization");
    public static final TaskType REPORT         = new TaskType("Report");
    public static final TaskType SYSTEM_TEST    = new TaskType("SystemTest");
    public static final TaskType SIMULATION     = new TaskType("Simulation");
    public static final TaskType REPROCESSING   = new TaskType("Reprocessing");
    public static final TaskType CONVERSION     = new TaskType("Conversion");

    private TaskType(String name) {super(name);}
}
```

```
task.setType(TaskType.SIM_RECON_DIGI);
```

- ANALYSIS TaskType - TaskType
- CONVERSION TaskType - TaskType
- DIGITIZATION TaskType - TaskType
- RECONSTRUCTION TaskType - TaskType
- REPORT TaskType - TaskType
- REPROCESSING TaskType - TaskType
- SIM_RECON_DIGI TaskType - TaskType
- SIMULATION TaskType - TaskType
- SYSTEM_TEST TaskType - TaskType

- What just happened?
  - Use **existing** Oracle database created for Perl Pipeline
  - Use 3rd party Enum library with no knowledge of Hibernate
  - Map Perl-style enums to type-safe Java enums
  - Everything done declaratively

An example of HQL

- HQL == “Hibernate Query Language”
- When all you want is data, not objects (which is often)

SQL with Java bean “dot” notation.

```
public Map loadTaskNameAndIdMap() throws DataAccessException {
    StringBuffer sb = new StringBuffer("select task.id, task.name from ");
    sb.append(Task.class.getName());
    sb.append(" as task");
    Iterator iterator = getHibernateTemplate().iterate(sb.toString());

    Map map = new HashMap();
    while (iterator.hasNext()) {
        Object[] row = (Object[]) iterator.next();
        Long id = (Long) row[0];
        String name = (String) row[1];
        map.put(name, id);
    }

    return map;
}
```

Security

- Use Spring's declarative security approach

```
/**
 * @@securityConfig("ROLE_PIPELINE_ADMINISTRATOR)
 */
public void delete(Task task) throws DataAccessException;
```

Declarative configuration using "metadata"

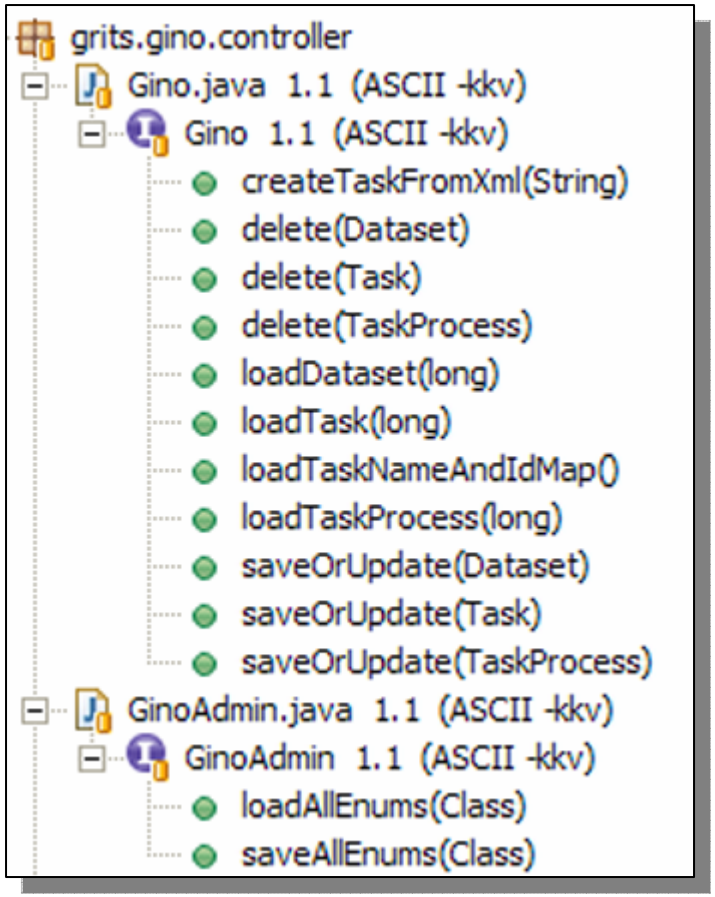
- Microsoft .NET style
- Just to show something different
- Could have used bean factory

- Single Sign On Service
  - Applications should never touch the password
    - DOE requirement
  - Yale's Central Authentication Service (CAS)
  - http://www.yale.edu/tp/auth/
  - Simple .war file
  - Accept credentials over HTTPS
  - Many clients
    - Java, Perl, Python, …
  - Authenticate to
    - Kerberos
    - Simple database tables
    - etc.
  - Max Turi connected CAS to SLAC Kerberos (Windows only)

Pipeline 2.0

- Status
  - OO Pipeline Design **without** regard to database
    - Domain and DAO
      - Design interfaces
      - Implement classes
      - Document implementation (Javadoc)
    - Logic (scheduler)
      - Hibernate
      - Spring + Quartz + JMS
      - Dan's special sauce
    - Launch and track
  - Hibernate entire Pipeline
    - Map this design onto existing Oracle 9i GLAST_DP database
  - Configuration
    - XML file upload
    - Web editing
  - Web reports
    - Aggregate reports
    - Individual reports

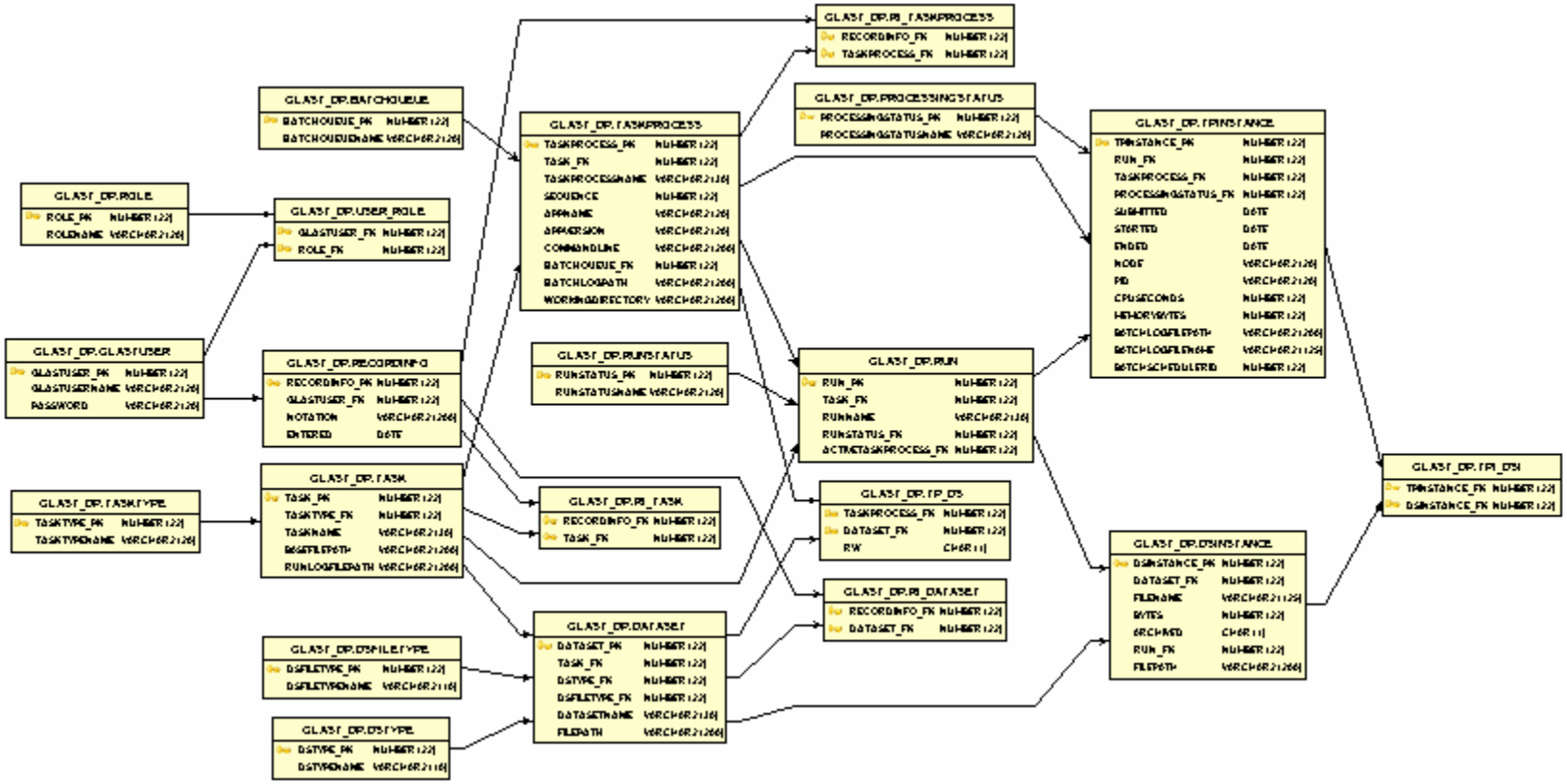
Pipeline 2.0 API



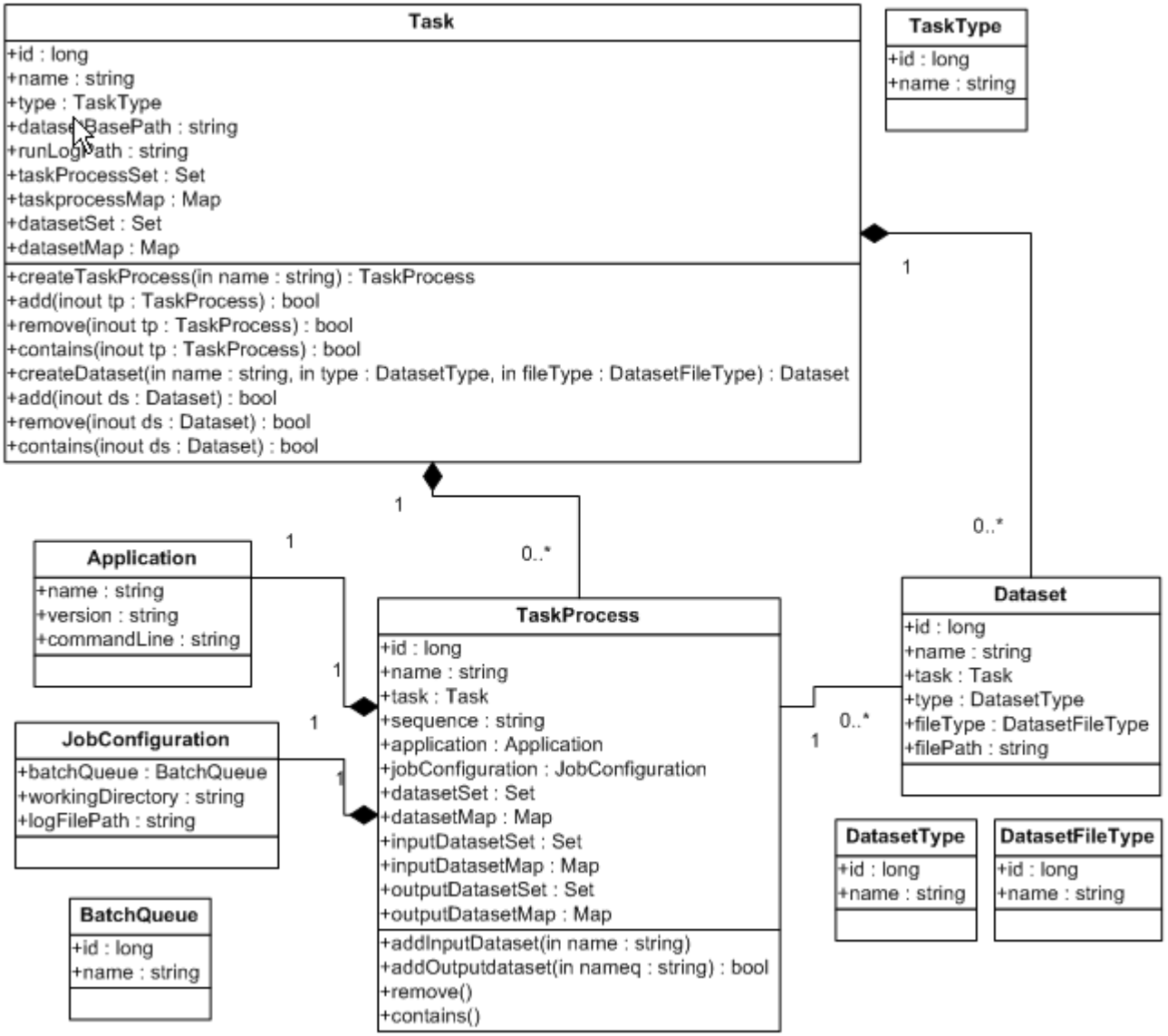
- Primary, “business interface”
  - Ready now
  - Tested
  - Documented
  - 50+ classes (not including unit tests)
- Created for XML file upload and round-trip web editing.
- Designed and implemented with entire Pipeline in mind.

Pipeline Database Schema

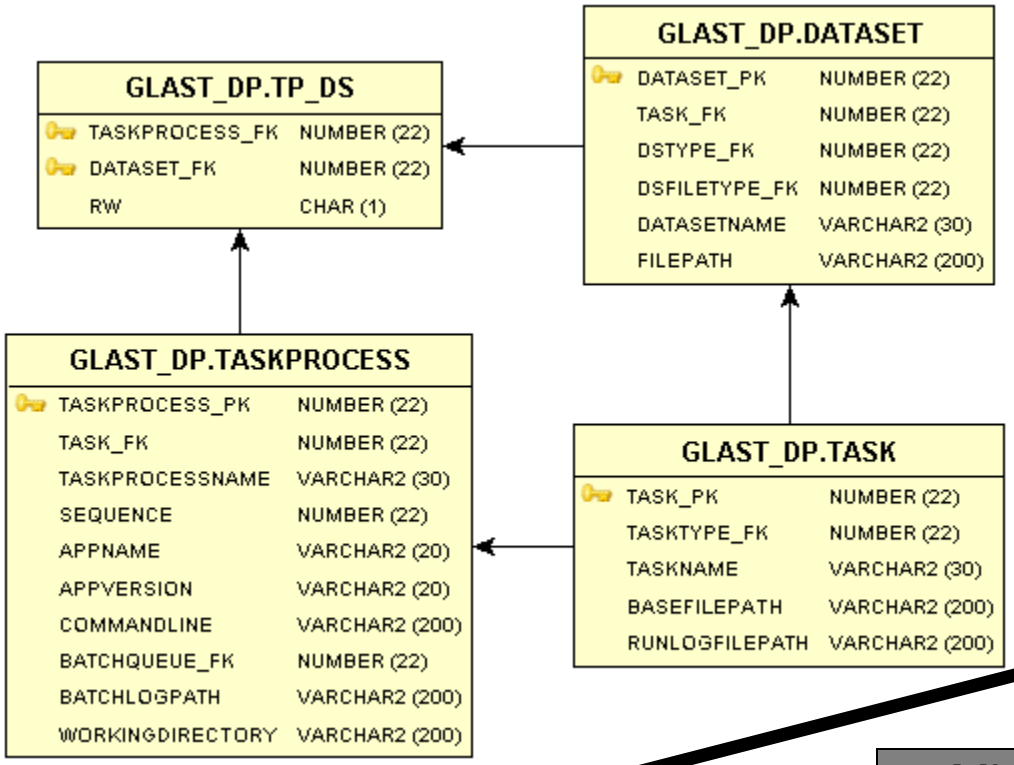
21 Tables
27 relations
and growing…



Pipeline 2.0 UML



Even with complex DB relations…



TASKPROCESS_FK	DATASET_FK	RW
2	2	R
2	3	W
77	60	W
78	60	R
78	61	W
208	224	W
208	225	W
154	129	W
154	128	W
154	126	W

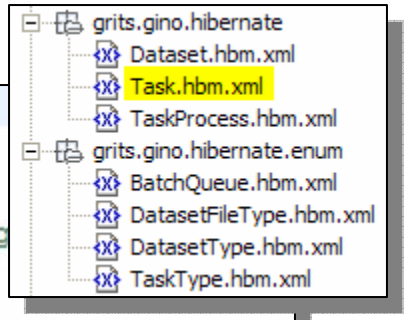
Integration Tier

Middle Tier

Program using Java without concern for underlying DB

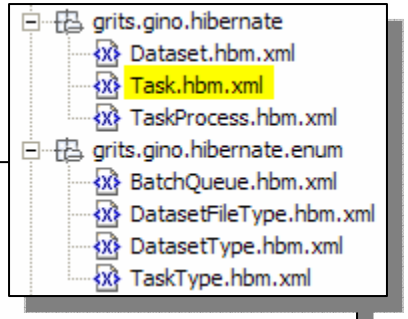
```
public Set getInputDatasetSet();
public void setInputDatasetSet(Set datasetSet);

public Set getOutputDatasetSet();
public void setOutputDatasetSet(Set datasetSet);
```

Mapping a Pipeline Task




```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping

<hibernate-mapping package="grits.gino.domain">

    <class name="TaskImpl" table="TASK">

        <id name="id" type="long" unsaved-value="null">
            <column name="TASK_PK" />
            <generator class="native">
                <param name="sequence">TASK_SEQ</param>
            </generator>
        </id>

        <property name="name" type="string">
            <column name="TASKNAME" length="30" unique="true"
                not-null="true" />
        </property>

        <property name="datasetBasePath" type="string">
            <column name="BASEFILEPATH" length="200" not-null="false" />
        </property>
```

Mapping a Pipeline Task




```
<many-to-one name="type">
    <column name="TASKTYPE_FK" not-null="true" />
</many-to-one>

<set name="taskProcessSortedSet" inverse="true"
    cascade="all-delete-orphan" sort="natural" order-by="SEQUENCE asc">

    <key column="TASK_FK" />
    <one-to-many class="TaskProcessImpl" />
</set>

<set name="datasetSetInternal" inverse="true"
    cascade="all-delete-orphan">

    <key column="TASK_FK" />
    <one-to-many class="DatasetImpl" />
</set>
```

Pipeline XML File Upload

```
langston@trinity /cygdrive/c/java/eclipse/workspace/grits-gino/example
$ java -cp ../target/grits-gino-0.1.jar ./recon-EM2-v1r0-raw.xml
```

Upload Configuration File

Upload a Configuration File

File to upload

C:\java\eclipse\workspace\grits-gino\example\recon-E [Browse...]

[Upload Configuration File]

Pipeline configuration file (XML)

TASK_PK	TASKNAME	BASEFILEPATH	RUNLOGFILEPATH	TASKTYPE_FK
3	allGamma-GR-v4r6-test1	/nfs/farm/g/glas...	/temp/	1
4	recon-EM2-v1r0	/nfs/farm/g/glas...	/temp/	4

TASKPROCESS_PK	TASK_FK	TASKPROCESSN
7	3	gleam
8	4	LaunchReport
9	4	LaunchSVAC
10	4	meritRootFile
11	4	recon
12	4	reconRootFile

TASKPROCESS_FK	RW	DATASET_FK
7	W	10
7	W	11
7	W	12
7	W	13
10	R	14
11	W	14
9	R	15
11	R	15
8	R	15
11	W	16
11	W	17
8	R	18
9	R	18
11	W	18
12	R	18

DATASET_PK	TASK_FK	DATASETNAME	DSTYPE_FK	DSFILETYPE_FK	FILEPATH
10	3	merit	8	2	/
11	3	digi	5	2	/
12	3	recon	15	2	/
13	3	mc	1	2	/
14	4	merit	8	2	calib-v1r0/grRoot
15	4	digi	5	2	grRoot
16	4	script	11	6	calib-v1r0/grRoot
17	4	jobOptions	3	1	calib-v1r0/grRoot
18	4	recon	15	2	calib-v1r0/grRoot

Reads, Inserts, updates and deletes covering 8 tables

```
<name>recon-EM2-v1r0</name>
<type>Reconstruction</type>
<dataset-base-path>/nfs/.../$(RUN_NAME)</dataset-base-path>
<run-log-path>/temp/</run-log-path>

<executable name="reconWrapper" version="v1r0">
  /nfs/.../reconWrapper.pl
</executable>

<batch-job-configuration name="long-job" queue="long">
  <working-directory>/nfs/.../$(RUN_NAME)</working-directory>
  <log-file-path>/nfs/.../$(RUN_NAME)/calib-v1r0/grRoot</log-file-path>
</batch-job-configuration>

<file name="digi"       type="DIGI"   file-type="root">grRoot</file>
<file name="jobOptions" type="text"   file-type="jobOpt">calib-v1r0/grRoot</file>
<file name="merit"      type="merit"  file-type="root">calib-v1r0/grRoot</file>
<file name="recon"      type="RECON"  file-type="root">calib-v1r0/grRoot</file>
<file name="script"     type="script" file-type="csh">calib-v1r0/grRoot</file>

<processing-step name="recon"
  executable="reconWrapper"
  batch-job-configuration="long-job">
  <input-file name="digi"/>
  <output-file name="jobOptions"/>
  <output-file name="merit"/>
  <output-file name="recon"/>
  <output-file name="script"/>
</processing-step>
```

Hibernate - under the hood

```
SessionImpl:555 - opened session
JDBCTransaction:37 - begin
JDBCTransaction:41 - current autocommit status:true
JDBCTransaction:43 - disabling autocommit
SessionImpl:1387 - saveOrUpdate() unsaved instance
SessionImpl:825 - saving [grits.gino.domain.TaskImpl#<null>]
SessionImpl:2309 - executing insertions
Cascades:497 - processing cascades for: grits.gino.domain.TaskImpl
Cascades:506 - done processing cascades for: grits.gino.domain.TaskImpl
WrapVisitor:81 - Wrapped collection in role: grits.gino.domain.TaskImpl.taskProcessSortedSet
WrapVisitor:81 - Wrapped collection in role: grits.gino.domain.TaskImpl.datasetSetInternal
Cascades:341 - id unsaved-value strategy NULL
EntityPersister:490 - Inserting entity: grits.gino.domain.TaskImpl (native id)
BatcherImpl:200 - about to open: 0 open PreparedStatements, 0 open ResultSets
SQL:226 - insert into GLAST_DP.TASK (TASKNAME, BASEFILEPATH, RUNLOGFILEPATH, TASKTYPE_FK, TASK_
BatcherImpl:249 - preparing statement
EntityPersister:388 - Dehydrating entity: [grits.gino.domain.TaskImpl#<null>]
StringType:46 - binding 'allGamma-GR-v4r6-test1' to parameter: 1
StringType:46 - binding '/nfs/farm/g/glast/u13/DC2/PipelineTest/allGamma/rootData/$(RUN_NAME)/'
StringType:46 - binding '/temp/' to parameter: 3
Cascades:341 - id unsaved-value strategy NULL
LongType:46 - binding '1' to parameter: 4
BatcherImpl:207 - done closing: 0 open PreparedStatements, 0 open ResultSets
BatcherImpl:269 - closing statement
BatcherImpl:200 - about to open: 0 open PreparedStatements, 0 open ResultSets
SQL:226 - call identity()
BatcherImpl:249 - preparing statement
AbstractEntityPersister:1236 - Natively generated identity: 1
```

Benefits of External Configuration

jdbc-oracle.properties

```
jdbc.driverClassName=oracle.jdbc.driver.OracleDriver
jdbc.username=GLAST_DP
jdbc.password********

db.host=slac-oracle.slac.stanford.edu
db.port=1521
db.sid=SLACPROD

jdbc.url=jdbc:oracle:thin:@${db.host}:${db.port}:${db.sid}

hibernate.dialect=net.sf.hibernate.dialect.Oracle9Dialect
hibernate.default_schema=GLAST_DP
```

Yesterday, oracle-dev was down.

Simple change to Spring configuration file and we are back up.

database properties files

- jdbc-hsqldb-mem.properties
- jdbc-hsqldb.properties
- jdbc-oracle-dev.properties
- jdbc-oracle.properties

```
<bean
    id="propertyConfigurer"
    class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="locations">
        <list>
            <value>classpath:/grits/gino/jdbc-oracle.properties</value>
        </list>
    </property>
</bean>

<bean
    id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
    destroy-method="close">
    <property name="driverClassName"><value>${jdbc.driverClassName}</value></property>
    <property name="url"><value>${jdbc.url}</value></property>
    <property name="username"><value>${jdbc.username}</value></property>
    <property name="password"><value>${jdbc.password}</value></property>
</bean>
```

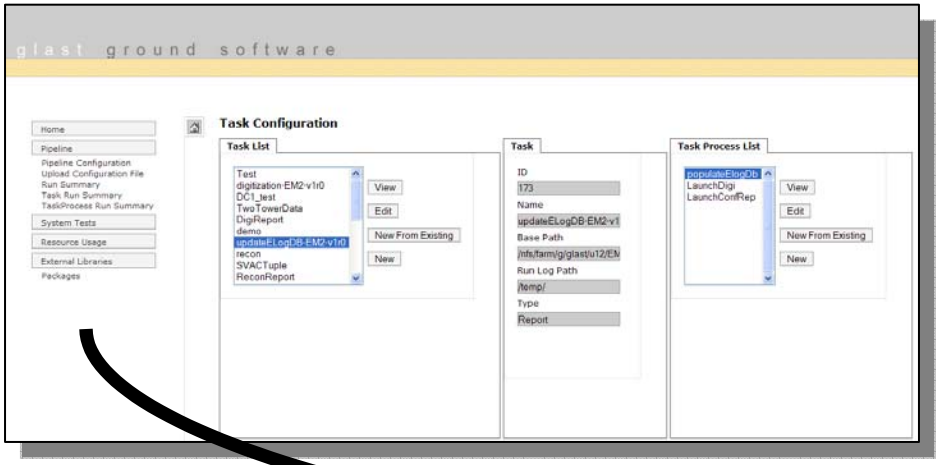
Data Integrity of a Legacy Database

```
public static class RWEnum extends PersistentStringEnum {
    public static final RWEnum READ = new RWEnum("read", "R");
    public static final RWEnum WRITE = new RWEnum("write", "W");

    public RWEnum() {}

    private RWEnum(String name, String persistentValue)
    {super(name, persistentValue);}
}
```

```
public static class RWEnum extends PersistentCharacterEnum {
    public static final RWEnum READ = new RWEnum("read", 'R');
    public static final RWEnum WRITE = new RWEnum("write", 'W');

    public RWEnum() {}

    private RWEnum(String name, char persistentValue)
    {super(name, persistentValue);}
}
```

Pipeline 2.0 Infrastructure

- Main site for users:
  - http://glast-ground.slac.stanford.edu/
- Main site for developers:
  - http://glast-ground.slac.stanford.edu/maven-projects/grits-gino/
  - http://glast-ground.slac.stanford.edu/maven-projects/grits-common/



Task Run Summary Refresh

ID	Name	Type	Done	Failed	Running	Waiting
1	Test	SimReconDig	0	0	0	1
175	digitization-EM2-v1r0	Digitization	1	9	0	0
81	DC1_test	SimReconDig	0	0	0	0
70	TwoTowerData	SimReconDig	0	0	0	0
143	DigiReport	Report	1	0	0	0
90	demo	test	29	1	0	0
173	updateELogDB-EM2-v1r0	Report	4	229	0	16
142	recon	Reconstructio	1	0	0	0
144	SVACTuple	Analysis	1	0	0	0
145	ReconReport	Report	1	0	0	0

Task Process View:

Summary ◉

Detail ○

ID	Name	Sequence	Succeeded	Failed	Prepared	Submitted	Running
187	populateElogDb		9	224	0	0	0
188	LaunchDigi		5	4	0	0	0
189	LaunchConfRep		4	1	0	0	0

Development Process

- Release Manager (Java)
  - Automated builds
  - CVS integration
  - Generate documentation
  - Run unit tests
  - Reports (unit tests, code coverage, metrics, etc.)
  - Can build “anything”
    - .jar
    - .war
- Dependency management (extlibs)
  - External Library Manager
  - Manage and track all versions
- Maven
  - Easily extensible

```
<!-- ========================================
<!-- GRITS commons
<!-- ========================================
<dependency>
    <groupId>glast</groupId>
    <artifactId>grits-common</artifactId>
    <version>0.1</version>
    <type>jar</type>
</dependency>
<!-- ========================================
<!-- Spring Framework 1.1.1
<!-- ========================================
<dependency>
    <groupId>spring</groupId>
    <artifactId>spring</artifactId>
    <version>1.1.1</version>
    <type>jar</type>
</dependency>
```

Development Process

File Edit View Go Bookmarks Tools Help

http://glast-ground.slac.stanford.edu/maven-projects/grits-gir

Zebra Tables: A List ... | CSS Structure and R... | Style Sheets - 183 o... | PrettyPrinter.de, an... | :: Shell-Shocked :: ...

All Classes

Packages

grits.gino.controller
grits.gino.dao
grits.gino.dao.hibernate
grits.gino.dao.hibernate.enum
grits.gino.dao.hibernate.enum.singl
grits.gino.dao.ibatis
grits.gino.dao.xmlbeans

All Classes

Application
ApplicationImpl
BatchQueue
BatchQueue
Dataset
DatasetChangeEvent
DatasetChangeListener
DatasetDao
DatasetFileType
DatasetFileType
DatasetImpl
DatasetType
DatasetType
EnumDao
EnumInterceptor
EnumToOldEnumTranslator
EnumUserType
Gino
GinoAdmin
GinoAdminImpl
GinoImpl
HibernateDatasetDao
HibernateEnumDao
HibernateTaskDao
HibernateTaskProcessDao
JobConfiguration

Overview Package Class Use Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

GLAST Pipeline 0.1 API

Packages	
grits.gino.controller	
grits.gino.dao	
grits.gino.dao.hibernate	
grits.gino.dao.hibernate.enum	
grits.gino.dao.hibernate.enum.singleclass	
grits.gino.dao.ibatis	
grits.gino.dao.xmlbeans	
grits.gino.domain	
grits.gino.domain.enum	
grits.gino.domain.event	

Overview Package Class Use Tree Deprecated Index Help

PREV NEXT FRAMES NO FRAMES

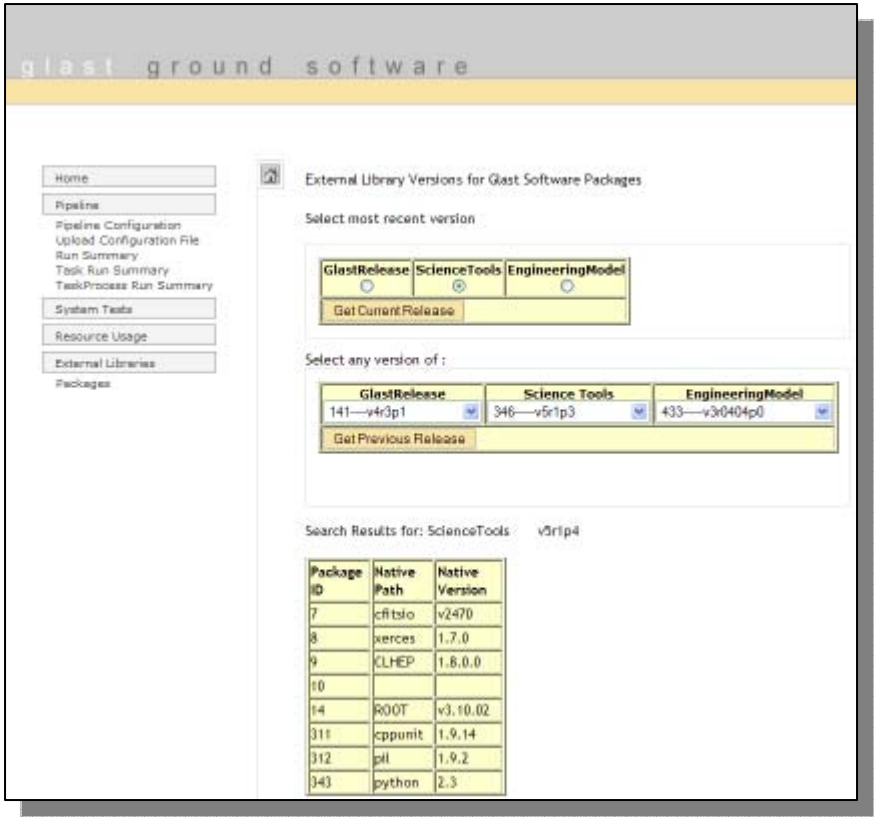
Done

Package	TC
edu.stanford.slac.glastGround.pipeline	54
edu.stanford.slac.glastGround.pipeline.impl	25
grits.gino.controller	4
grits.gino.dao	4
grits.gino.dao.hibernate	5
grits.gino.dao.hibernate.enum	7
grits.gino.dao.hibernate.enum.singleclass	4
grits.gino.dao.ibatis	4
grits.gino.dao.xmlbeans	3
grits.gino.domain	18
grits.gino.domain.enum	4
grits.gino.domain.event	4

Artifact ID	Type	Version
aopalliance	jar	1.0
aspectjrt	jar	1.2.1
cglib-full	jar	2.0.2
commons-collections	jar	3.1
commons-dbcp	jar	1.0
commons-lang	jar	2.0
commons-logging	jar	1.0.4
commons-pool	jar	1.0
dom4j	jar	1.5
ehcache	jar	0.9
grits-common	jar	0.1
hibernate	jar	2.1.6

hsqldb	jar	1.7.2.7
ibatis-common	jar	2.0.6
ibatis-sqlmap	jar	2.0.6
jargs	jar	0.4
jta	jar	1.0
jug	jar	1.1
junit	jar	3.8.1
log4j	jar	1.2.8
mockobjects-core	jar	0.09
odmg	jar	3.0
jdbc	jar	9i
spring	jar	1.1.1
xbean-apache	jar	1.0-DEV

ColdFusion MX and C++ External Libraries



- Karen Heidenreich
  - ColdFusion MX proof-of-concept
  - Used Dreamweaver to create simple “portlet”

ColdFusion MX

```
<cfquery name ="GlastReleases" datasource="Installer">
select  pkgid,pkgname,pkgversion from packages
 where pkgname = 'GlastRelease'
 and tagid = 1 order by pkgname,pkgversion
</cfquery>
```

```
<table width="200" border="1" bgcolor="#FFFFCC">
<tr>
<td><strong> Package ID </strong></td>
<td><strong>Native Path</strong></td>
<td><strong> Native Version</strong></td>
</tr>

<cfoutput query="GlastReleaseSelect">
<tr>
<td>#pkgid#</td>
<td>#natpath#</td>
<td>#natversion#</td>
</tr>
</cfoutput>
```

Example query taken from Karen's code

What I Didn’t Cover

- ColdFusion MX
  - Runs fine on Tomcat
  - Other implementations (BlueDragon) toprotect against vendor lock-in
  - FLEX
- Dashboards with ColdFusion MX
  - `<cfquery>` for Databases
    - query of queries missing from JSP
  - `<cfinvoke>` for Web Services
    - Missing from JSP
- Security
- Remoting
- Email and Scheduling

Conclusion

- Java as an infrastructure platform
  - Lightweight containers make this possible for small groups with disparate talents
    - Make pragmatic use of technologies (not ideological ones)
      - The simplest thing that can possible work
      - Open Source when it makes sense
      - Commercial products when they make sense (Dreamweaver, ColdFusion, etc.)
  - Rich collection of high quality Open Source software
    - Tomcat
    - Spring
    - Hibernate
  - Much GLAST Pipeline “infrastructure” exists
    - Domain model
    - DAO implementations
    - Dashboard
    - Development environment
  - Leverage resources
    - web developers
    - SLAC Java group
    - ISOC