J2EE for Glast

Matthew D. Langston (SLAC)
4/25/2004
What is J2EE?

- Java 2 Platform, Enterprise Edition
  - Current specification is J2EE version 1.4
- A platform-agnostic operating system for developing component-based applications
- Provides common “middleware” or “plumbing services” common to all applications:
  - Database persistence
  - Declarative transactions
  - Role-based security
  - Legacy application integration
  - Component interoperability
  - transparent fail-over, logging and auditing, etc.
- Designed to run mission-critical, 24x7 data-processing-intensive applications that interface and integrate with legacy non-Java based applications and systems
J2EE Technologies

version 1.3 and later

- EJB 2.0
  - Enterprise Java Beans
- JDBC
  - Java Database Connectivity
- JTA and JTS
  - Java Transaction API
  - Java Transaction Service
- JAAS
  - Java Authentication and Authorization Service
- RMI and RMI-IIOP
  - Java Remote Method Invocation
  - CORBA integration
- JCA
  - J2EE Connector Architecture
- JNDI
  - Java Naming and Directory Interface
- Java Servlets
- JSP
  - Java Server Pages
- JAXP
  - Java API for XML Parsing
- JavaMail
- Java IDL

Lots of acronyms. This is the middleware, or common plumbing, provided by the Application Server (“J2EE Container”, or just “Container”).
Enterprise Java Beans

EJB 2.0

• Of all of the middleware, EJB 2.0 is most important to Glast Infrastructure Projects
  – Container Managed Persistence (CMP)
    • Automatic DB table creation, transactions, inserts, updates, queries, deletes
    • Automatic transactions
  – Container Managed Relations (CMR)
    • One-to-many or many-to-many relationships among objects (i.e. just like a relational database)
  – EJB-QL (EJB Query Language for finder methods)
    • `SELECT OBJECT (h)
      FROM SystemTests AS s, IN(s.histograms) AS h
      WHERE h.subsystem = 'CAL' AND h.entries = 0`
Glast Infrastructure Projects
Functionality Overlap

- **Processing Pipeline**
  - Submit batch jobs
  - Track batch jobs
  - Track datasets
  - Send Notifications
    - email, pager, etc.
  - Generate reports
  - Web front end

- **Release Manager**
  - Submit batch jobs
  - Track batch jobs
  - Track builds
  - Notifications
    - email, pager, “run System Tests”
  - Generate reports
  - Web front end

- **Data Server**
  - Submit batch jobs
  - Track batch jobs
  - Track datasets
  - Notifications
    - email
  - Web front end

- **System Tests**
  - Submit batch jobs
  - Track batch jobs
  - Track builds
  - Track datasets
  - Notifications
    - plot out of spec, etc.
  - Generate reports
  - Run system tests for each new release
Glast’s Infrastructure Choices

1. Do nothing
   • continue down script-based Perl + Oracle path
2. CORBA
3. Microsoft .NET
4. J2EE
Perl + Oracle

– Pros
  • Leverage existing programming knowledge
  • Current Infrastructure Projects already exist

– Cons
  • Must write middleware by hand
    – DB persistence, transactions, failover and fail recovery, etc.
    – Easy to get wrong, lots of effort, so why?
  • Doesn’t scale
  • No reuse in practice (too much effort)
  • Little to no collaboration
  • Non-portable
Why J2EE for Glast

- “Mission critical, 24x7 data-processing-intensive applications that interface and integrate with legacy non-Java based applications and systems”. Sound like us?
- A programming language (Java) that supports larger scale team-oriented server and service based projects.
  - type-safety
  - native exception handling
  - native support for object-oriented programming
  - native support for interfaced based programming
  - native support for code structuring
  - excellent development and runtime support for Linux and Windows
- A rich suite of libraries and frameworks (the middleware) that supports server and service based projects.
- Strong support for integrating third-party and/or legacy non-Java based applications that run on disparate platforms.
- Portable – tell Goddard to install a J2EE Application Server and e-mail them pipline.ear.
EJB 2.0
The Triad of Beans

• Session Beans
  – Provide the logic and algorithms of the application

• Entity Beans
  – Provide database persistence

• Message-Driven Beans
  – Lightweight vehicle for object communications
  – Guaranteed message delivery
EJB Objects

1: Call a Method
2: Call Middleware APIs
3: Call a Bean
4: Method Returns
5: Return Result

Client Code, such as Servlets or Applets

EJB Object

Remote Interface

EJB Container/Server

Transaction Service, Security Service, Persistence Service, etc

Enterprise Bean
Home Interfaces and Objects

1: Create a New EJB Object

2: Create EJB Object

3: Return EJB Object Reference

Client Code, Such as Servlets or Applets

Home Object

Home Interface

Remote Interface

EJB Object

EJB Container/Server

Enterprise Beans
Physical Components of an EJB
EJB Object Model

- **java.rmi.Remote**: Comes with Java 2 platform
- **java.io.Serializable**: Comes with Java distribution
- **javax.ejb.EJBLocalObject**, **javax.ejb.EJBObject**: Supplied by Bean provider (we will write)
- **javax.ejb.EJBHome**, **javax.ejb.EJBLocalHome**: Generated for us by container vendor's tools
- **javax.ejb.EnterpriseBean**, **javax.ejb.SessionBean**: Additional interfaces for enterprise beans

- **Hello World Local Interface**, **Hello World Remote Interface**, **Hello World Home Interface**, **Hello World Local Home Interface**: Interfaces for the Hello World Bean
- **Hello World EJB Local Object**, **Hello World EJB Object**, **Hello World Home Object**, **Hello World Local Home Object**: Implementations for the Hello World Bean
Is it too complicated?

• You mean I have to write 5 Java classes just to get persistence?
  – No. In practice you write one file as you would expect and use attributed programming to declaratively specify your requirements (J2EE uses XDoclet + ant for attributed programming).

• Only if Linux and Windows are too complicated for us. J2EE is a platform-agnostic OS.

• Concepts are different, but not difficult.

• Wide assortment of Tools.

• My favorite is IDE is Eclipse, but there are may others.

• Large body of excellent literature. Many have gone before us.
Pet Store
Use Cases

When the customer enters into the Pet Store web site, she can:

- Sign into the site
- Sign off from the site
- Browse the catalog
- Update her shopping cart
- Place her order
- Receive an email notification
- Manage her account
Pet Store
Page Flow

Diagram:

- **Signon**
- **Register**
- **Main**
- **Category**
- **Product**
- **Item**
- **MyAccount**
- **Cart**
- **Checkout**
- **Order**
Pet Store
Database

The database is composed of the following tables:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_ACCOUNT</td>
<td>Login table for the customers</td>
</tr>
<tr>
<td>T_CUSTOMER</td>
<td>Customer' information</td>
</tr>
<tr>
<td>TCATEGORY</td>
<td>The catalog categories (i.e. Fish, Dogs, Cats, etc.)</td>
</tr>
<tr>
<td>T_PRODUCT</td>
<td>Catalog products. Each product may have one or more variants (Items). A typical variant is usually male or female</td>
</tr>
<tr>
<td>T_ITEM</td>
<td>Individual product details</td>
</tr>
<tr>
<td>T_ORDER</td>
<td>Customers' orders</td>
</tr>
<tr>
<td>T_ORDER_ITEM</td>
<td>Items ordered by the Customers</td>
</tr>
</tbody>
</table>
The domain contains all the business objects of the application. The business objects maintain and persist the state of the application into the database.
Pet Store Services

The services manages the logic of the application.
Pet Store

Web
J2EE Code Example

• Pet Store
  – http://glast05.slac.stanford.edu:8180/xpetstore-ejb

• JBoss 3.2.3
  – http://www.jboss.org/index.html

• Eclipse IDE
  – http://www.eclipse.org/
Matt’s Reading List

- There is a vast amount of literature on the subject of J2EE and web applications. After culling through literally dozens of books on these subjects over the course of several months, I selected a subset which I felt were particularly excellent at teaching the core concepts from a pragmatic perspective, was well written, and was not regurgitations of existing reference material. These books are in my office, and I encourage anyone to borrow them at any time.
  - Walls C., Richards N., *XDoclet in Action* (2004); Manning Publications Co.
Summary

• J2EE was designed with Glast in mind. We only have to write those parts unique to our domain, and Container provides everything else.
• Perl isn’t going away anytime soon