MPV Lessons Learned

IntercontinentalExchange (ICE)

David Burton – Dir. of Web Development

Chase Stephens – Developer

Michael Boni – Developer

Roadmap

- MPV Application Overview
- Architecture
- Lessons Learned

Overview – ICE

- Electronic Commodity Exchange
- Energy Markets
  - ✓ Oil, Power, Gas, Metals & others
  - ✓ Futures & OTC (Financial & Physical)
  - ✓ US, Europe, Asia
- Anonymous Trading
- Volatility & Liquidity
- ICE Data – Indexes and Analysis

Overview – Valuations/Validation

- FASB/SOX—Mark their book
- Liquid Market – Use Published Index
- Less Liquid Market – Find Consensus
- Customers
  - ✓ Producers/Consumers
  - ✓ Banks/Investors
- Competitive Intelligence
  - ✓ Privacy Between Companies
  - ✓ No ‘fishing’ for data

Overview – Submit/Return

- Monthly Cycle
- Customers Submit Current Valuations
- Internal Analysts Cleanse for Errors
- Customers Resubmit as Needed
- Internal Analysts Cleanse for Narrow Range
- Return Consensus Valuations to Customers
- 1-2 Day Turnaround

Overview – Techniques

- Extreme/Agile Programming
- Pair Programming
- Incremental (monthly) Cycle
- 2 Developers & 3 Internal Users

Products

- JBoss AS
- JBoss TreeCache
- JBoss Transaction Manager
- Hibernate
- JBoss Messaging
- EJB 3.0 (beta)
- Struts

Architecture

- Original
- Prototype/Interim
- Migration
- Current
- Future

Architecture – Original

- Excel, VBA, Access (outsourced)
- Hard to Maintain
- Slow, Eats Memory
- Email for File Transfer
- No Collaboration Internally
- User Feedback by Phone

Architecture – Prototype/Interim

- Tomcat, Hibernate, EHCache, Struts
- XDoclet annotated POJOs
- POJO Data Access Objects
- Cache vs Query
- Web Forms for Submit/Returns
- Internal Analysis still uses Excel
- Hibernate Session per Request (using Filter)

Architecture - Migration

- JBoss, Hibernate, JBoss Cache
- Hibernate Session per Request
  - ✓ Failed to link Oracle Transactions
- Applet for Charting & Cleansing

Architecture - Current

- JBoss, EJB3, JMS, JBoss Cache
- Cache size limits
- Managed JTA Transaction per Request (using Filter)
- EJB3 Stateless Session DAOs
- JMS Queue for Worker Threads

Architecture - Clustering

- 2 Machines, 2 Instances of JBoss
  - ✓ Front JBoss for Web
  - ✓ Back JBoss for Computations (Grid?)
- Clustered JBoss Cache
- Shared JMS Queue
- Shared File System
  - ✓ NAS/NFS
  - ✓ JBoss HA Filesystem
- Reliability

Architecture – Cluster Workers

- JMS Work Queue
- Message Driven Bean
  - ✓ Rollback on Failure
- JMS Listener Thread/Worker
  - ✓ Started with Startup Servlet

Lessons Learned

- Object Behavior
- Cache vs Query
- Get/Set Parity
- JDBC Settings
- Transactions
- Applet Serialization
- Web Tier Rendering
- JBoss Support

Lesson – Basic Object Behavior

- *Hibernate in Action*
- Synthetic ID & Business Keys
- hashCode() and equals()
  - ✓ Use Business Key, not ID
- compareTo()
- toString()
- toXML() / fromXML()

Lesson – Basic Object Behavior

```
public class User {
  public long getId() ...
  public String getUserName() ...

  public boolean equals(Object o) {
    /* BAD */
    return ((User) o).getId() == getId();
    /* BETTER */
    return ((User)
   o).getUserName().equals(getUserName());
  }
}
```

Lesson Learned – Basic Object Behavior

- Read *Hibernate in Action*
- IDs are for databases
- Business keys are for objects

Lesson – Cache vs Query

- Hibernate 2nd Level Cache
  - ✓ findById(), walking lazy pointers
  - ✓ Sharing in Memory
  - ✓ Distributed Cache
  - ✓ High Memory Usage
- HQL Query
  - ✓ Sharing in Database
  - ✓ Low Memory Usage
  - ✓ Slower

Lesson Learned – Cache vs Query

- Evaluate object usage v. memory requirements
- Second-level cache isn't *always faster*
- *Consider caching frequently-used queries*

Lesson – Get/Set Parity

- Hibernate Persisted Objects
- Get & Set accessors must match
- Query followed by Update
- Update Locks Rows for Read
- Test: SQL Debug On

Lesson – Get/Set Parity

```
public class User {
   private Address _address;
   public setAddress(Address address) ...

   public Address getAddress() {
      /* This will cause Hibernate to do a write
         for every read of a User */
      return getAddress() == null ? new Address() : getAddress();
      /* Better to deal with nulls elsewhere */
      return _address;
   }
}
```

Lesson – Get/Set Parity

```
public class User {
  private Address _address;
  public setAddress(Address address) {
    /* This will also cause Hibernate to do a write
       for every read of a User */
    _address = (address == null) ? new Address() : address;
    /* Better to deal with nulls elsewhere */
    _address = address;
  }

  public Address getAddress() ...
}
```

Lesson Learned – Get/Set Parity

- Get & Set accessors must match
- Turn on SQL output for development

Lesson – JDBC Settings

- JDBC Fetch Size
  - ✓ 1,000+
  - ✓ Available Heap
- Hibernate Batch Size
  - ✓ N+1 Query Problem
  - ✓ Aggressive Loading of Collections

Lesson – JDBC Settings

```
<hibernate-configuration>
    ...
  <property name="show_sql">false</property>
  <property name="hibernate.cache.use_second_level_cache">true</property>
  <property name="hibernate.default_batch_fetch_size">16</property>
  <property name="hibernate.generate_statistics">true</property>
  <property name="hibernate.connection.autocommit">false</property>
  <property name="hibernate.jdbc.fetch_size">1000</property>
    ...
</hibernate-configuration>
```

Lesson Learned – JDBC Settings

- Hibernate works on top of JDBC – same principals apply
- Larger JDBC fetch size means better performance (to a point)
- Consider queries for populating many levels of an object tree

Lesson – Transactions

- Long Transactions (10+ minutes)
  - ✓ Oracle Timeout
  - ✓ JTA Timeout
  - ✓ TreeCache Lock Timeout
  - ✓ Browser Timeout
- Intermediate Commits
  - ✓ Chained Actions
- Container Managed vs Bean Managed
- J2EE filter for controlling transactions

Lesson – Transactions

```
import javax.transaction.TransactionManager;


public static Transaction beginTransaction(int timeoutSeconds)
throws InfrastructureException {
   ...
      TransactionManager manager = (TransactionManager) (new
InitialContext()).lookup("java:TransactionManager");
      if (manager.getStatus() == Status.STATUS_NO_TRANSACTION)
{
         manager.setTransactionTimeout(timeoutSeconds);
         manager.begin();
      }
      return manager.getTransaction();
   ...
}
```

Lesson Learned – Transactions

- Hibernate transactions attach to existing JTA transactions
- Bean-managed transactions offer finer control
- javax.transaction.TransactionManager
  - ✓ Allows more programmatic control
- javax.transaction.UserTransaction
  - ✓ Simple interface
- org.hibernate.transaction.*
  - ✓ Abstracts access; container independent

Lesson – Applet Serialization

- Prep Data for Serialize to Applet
  - ✓ Walk Lazy Proxies
  - ✓ Close Hibernate Session
  - ✓ Replace Proxy Collections
  - ✓ Stub upward Pointers in Tree
  - ✓ readObject() and writeObject() for order
- Lightminds Technique
  - ✓ Servlet call vs EJB3 call
- Security

Lesson – Applet Serialization

```
/* If we want all users to be populated before serialization,
we must call one accessor to populate each proxy object */
private void walkPointers(Company company) {
   for (User user : company.getUsers()) {
      user.getName();
   }
}

/* If we don't want users, we must clear the proxies before
serialization, otherwise we get a proxy error in the applet */
private void disconnectPointers(Company company) {
   company.setUsers(new HashSet<User>());
}
```

Lesson Learned – Applet Serialization

- EJB3 beta remoting had security issues (likely fixed now)
- Up-pointers can cause serialization deadlocks
- www.lightminds.com

Lesson – Web Tier Rendering

- Hibernate Session
  - ✓ Disconnect at Transaction End
  - ✓ Lazy Collection Proxy
- Transactions across servlet and EJB3
- Rendering in JSP pages
- Servlet Filter

Lesson – Web Tier Rendering

```
public void doFilter(...) {
   TransactionManager.beginTransaction();
   HibernateUtil.beginTransaction();
   try {
      filterChain.doFilter(request, response);
      HibernateUtil.commitTransaction();
   }
   catch (Exception e) {
      HibernateUtil.rollbackTransaction();
      TransactionManager.rollbackTransaction();
   }
   finally {
      TransactionManager.finish();
   }
}
```

Lesson Learned – Web Tier Rendering

- Lazy-loaded objects need an open Hibernate Session
- Using a filter handles most cases but leaves the transaction open longer than absolutely necessary
- Extra work needs to be done to ensure a single transaction across WAR and EJB3 archives

Lesson Learned – JBoss Support

- Mixed results
- Need reduced example
- Need detailed log files
- Point to documented answer
- Eventually get good answer if you ask good question

End – Questions?