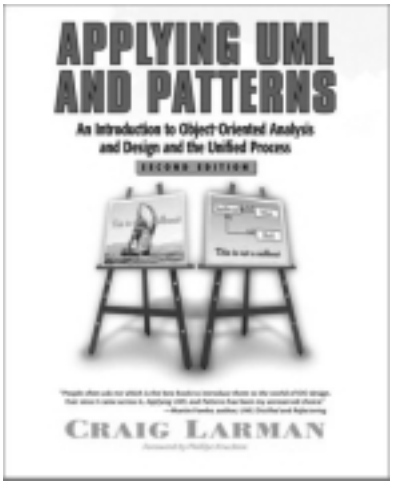
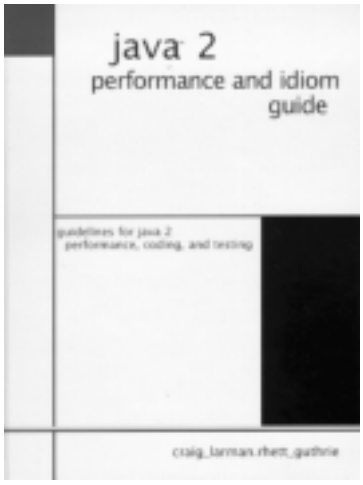
J2EE & EJB Design Patterns

Craig Larman
www.craiglarman.com

Speaker Biography

- www.craiglarman.com





Objectives

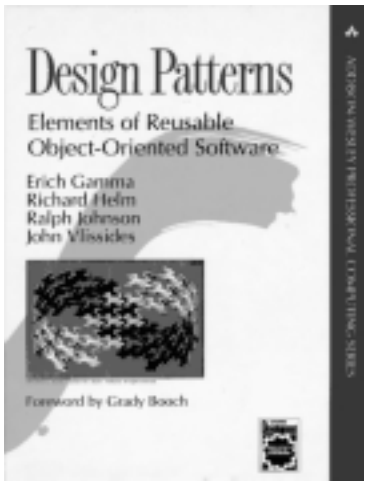
- See and apply J2EE and EJB design patterns.
- Read and apply *correct* UML.

Contents

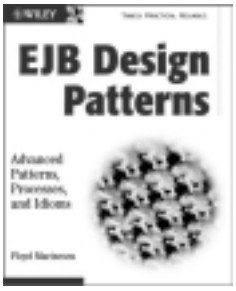
- General EJB Patterns
- Client-Side EJB Patterns
- Application Patterns
- Presentation Patterns
- Persistence Patterns

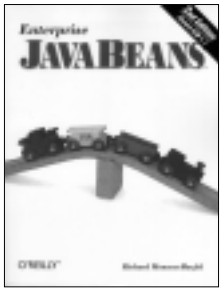
Patterns?

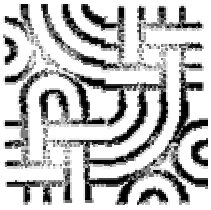
- Tried-and-true best-practices.
- Not new ideas, tried on one project.



Our Pattern Sources



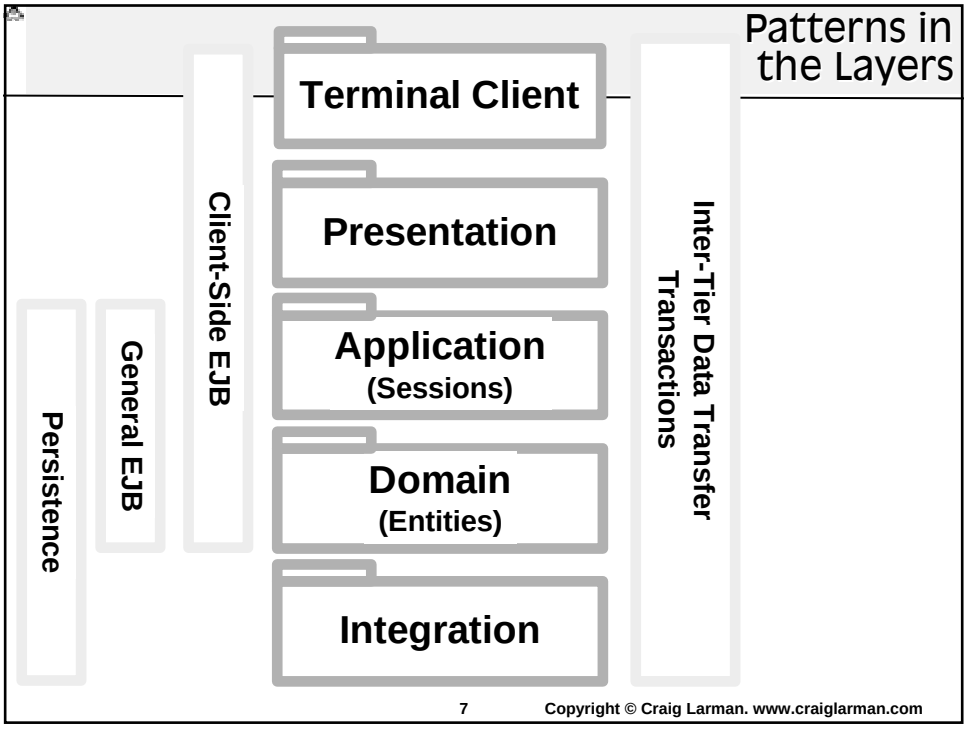


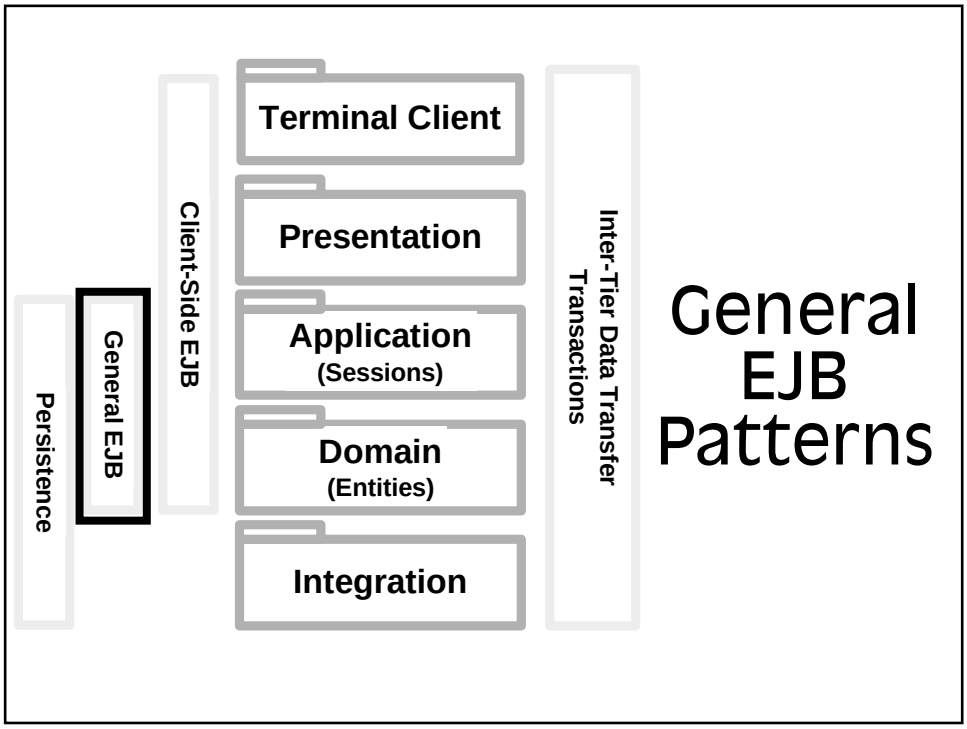


Portland Pattern Repository

c2.com/ppr

Fowler, . . .

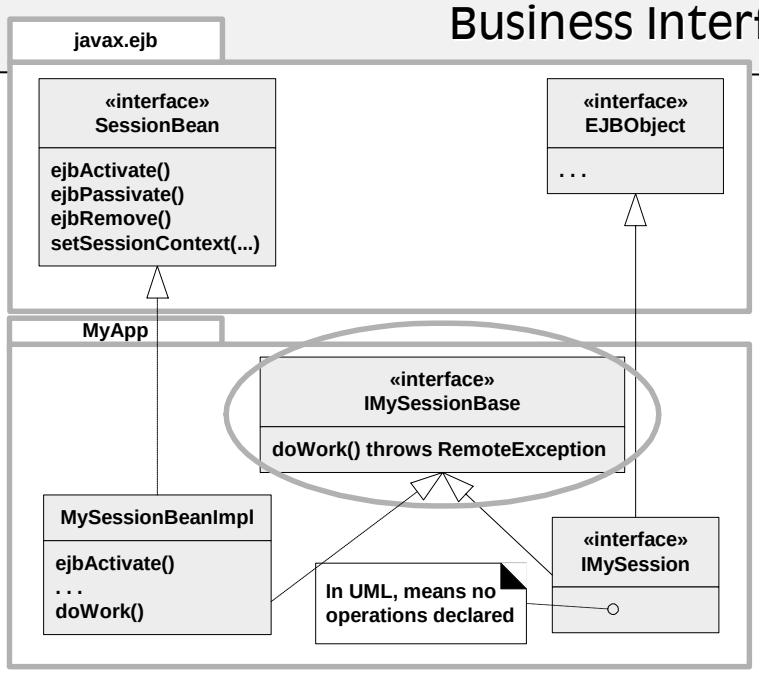

Patterns in the Layers
Terminal Client
Presentation
Application
(Sessions)
Domain
(Entities)
Integration
Client-Side EJB
General EJB
Persistence
Inter-Tier Data Transfer
Transactions
7
Copyright © Craig Larman. www.craiglarman.com


General
EJB
Patterns
Terminal Client
Presentation
Application
(Sessions)
Domain
(Entities)
Integration
Client-Side EJB
General EJB
Persistence
Inter-Tier Data Transfer
Transactions

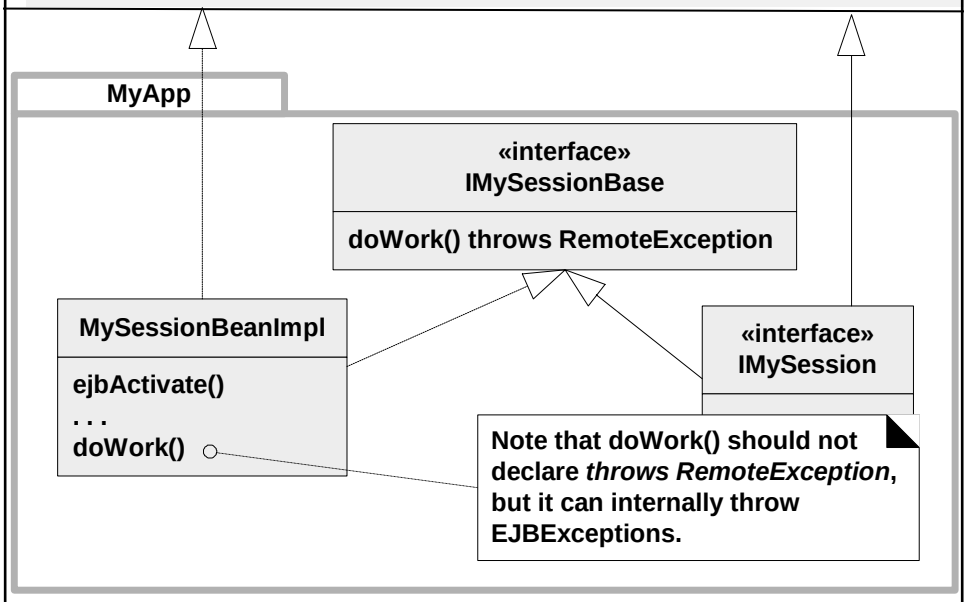
Business Interface

- Problem:
  - How to ensure the bean implements the correct methods, with correct signatures?
- IDEs provide wizards, but then one is dependent on an IDE.

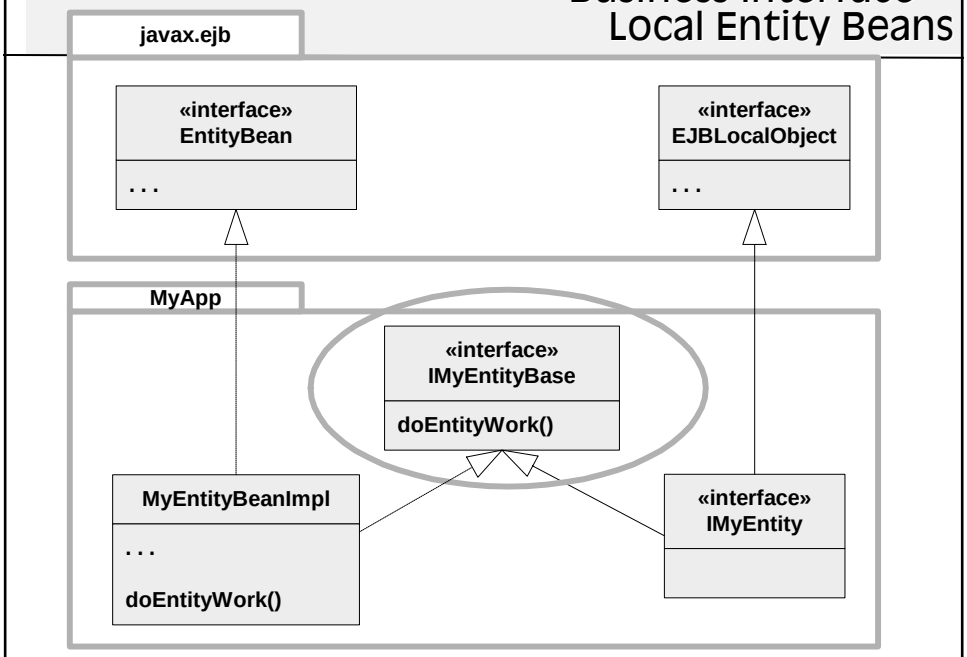
Business Interface



Business Interface—Session Beans



Business Interface—Local Entity Beans



Coarse-Grained Remote Interface

- Problem:
  - Fine-grained remote calls over a network impact performance.
- See also: Session Façade

Coarse-Grained Remote Interface

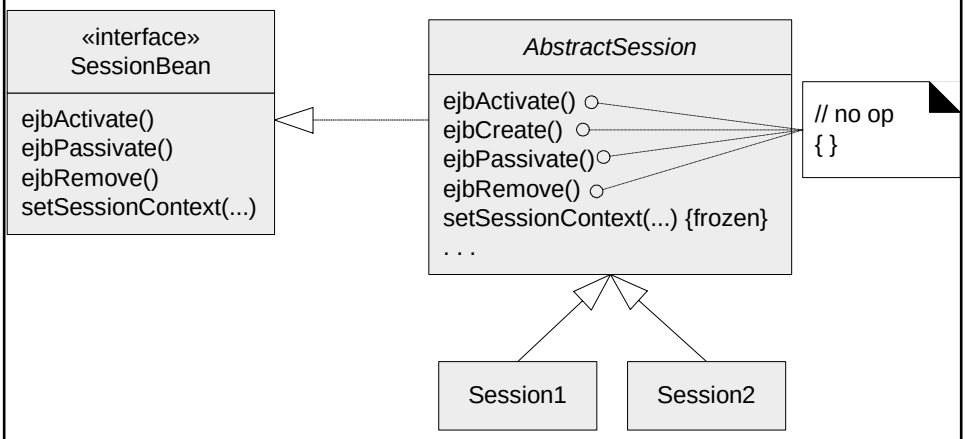
- Create "batch" operations.
- May also use a reflective *setData(Map)*
  - Note that there is a loss of security and transaction support related to particular values in this approach.

MyStockSessionBean
. . .
setData(Map) **setDJIQuotes(List)** **getDJIQuotes() : List** **getQuotes(symbols : List) : List**

Layer Supertype

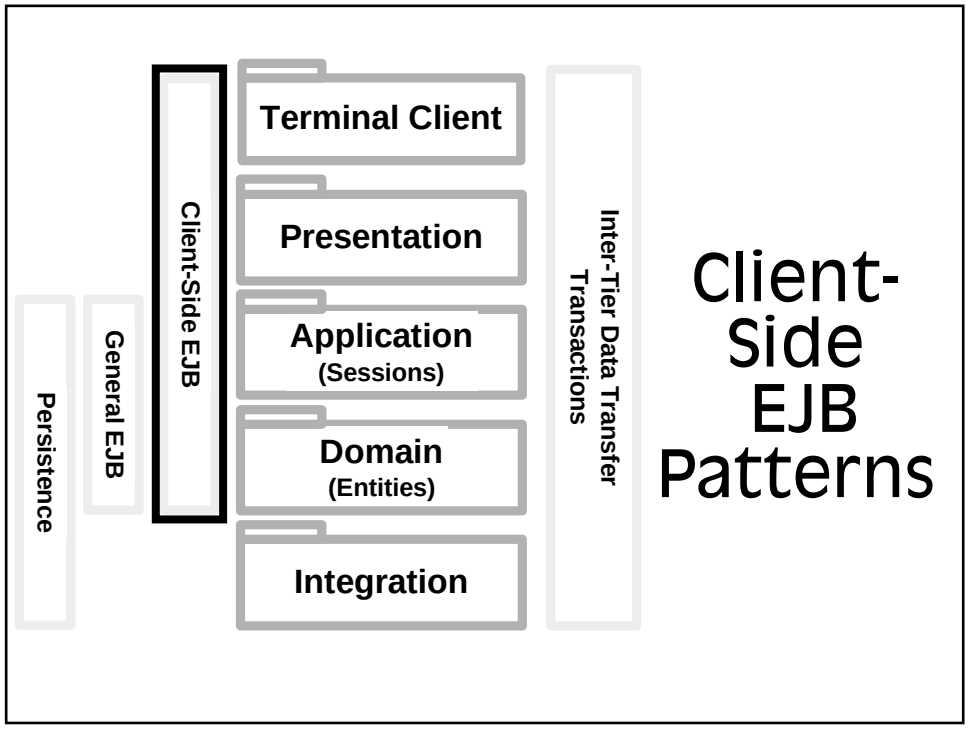
- Problem:
  - Session and entity beans must implement certain methods defined by their interfaces.
  - There are predictable services and behaviors required by these beans.

Layer Supertype



Layer Supertype

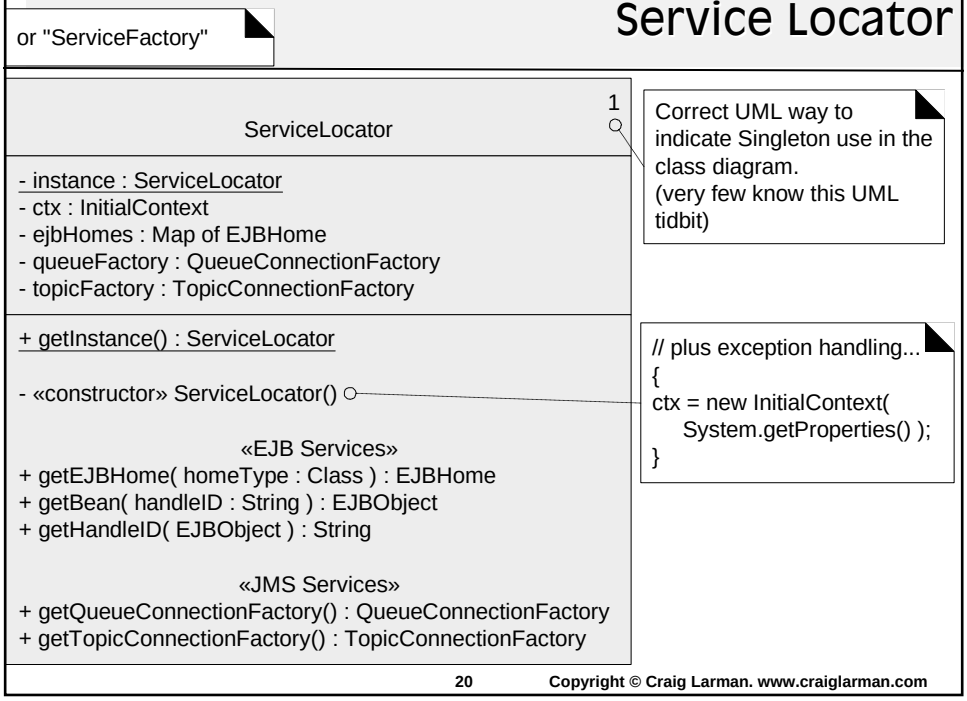
```
public abstract class AbstractSession implements SessionBean {
   private SessionContext ejbContext;

  . . .

      // restrict as only way to cache or return the context
   public final void setSessionContext( SessionContext ctx ) . . .
   protected final SessionContext getSessionContext() . . .

      // common helper services
   protected void log( String message )  . . .
   protected Connection getConnection( String url ) throws SQLException . . .
}
```

Service Locator

- Problem:
  - How should a client get access to remote services, to reduce coupling, complexity, and code duplication?
  - EJBHomes, Remote Proxies, JMS Factories, . . .
  - JNDI lookup is done in all these cases.

Service Locator



Service Locator

```
class MyClient {
  public void foo() {
    ForumServicesHome forumServicesHome =
          (ForumServicesHome) ServiceLocator.getInstance().
                getEJBHome(ForumServicesHome.class );

    IForumServices forumServices = forumServicesHome.create();

     forumServices.createForum( "J2EE and JMS" );
  }
}
```


Service Locator

```
class ServiceLocator {
  public EJBObject getBean( String handleID ) {
    try {
          byte[ ] bytes = handleID.getBytes();
          InputStream is = new ByteArrayInputStream( bytes );
          ObjectInputStream = ois = new ObjectInputStream( is );
          Handle handle = (Handle) ois.readObject();
          return handle.getEJBObject();
    }
    . . .
  }
}
```

Service Locator

- Note the Protected Variation regarding the JNDI key for a Home:
  - getEJBHome(homeType : Class)
- How to get the JNDI key?
  - Preferred: Use the EJB-REF tag in web.xml

Service Locator

- JNDI key via the EJB-REF tag in web.xml, and application server configuration file (e.g., weblogic.xml):
- Place ServiceLocator in WEB-INF\lib

Service Locator

```
// web.xml
<ejb-ref>
    <ejb-ref-name> com.foo.forumapp.ForumServicesHome </ejb-ref-name>
    <ejb-ref-type> Session </ejb-ref-type>
    <home> com.foo.forumapp.ForumServicesHome </home>
    <remote> com.foo.forumapp.IForumServices </remote>
</ejb-ref>

// weblogic.xml
<reference-descriptor>
    <ejb-ref-name> com.foo.forumapp.ForumServicesHome </ejb-ref-name>
    <jndi-name> ForumServicesHomeActualJNDIKey </ jndi-name >
</reference-descriptor >
```


Service Locator

```
// "java:comp/env/ejb/" is used for all JNDI lookups that have been made
// available in the environment

class ServiceLocator {
  public EJBHome getEJBHome( Class homeType ) {
    try {
    Object ref = ctx.lookup("java:comp/env/ejb/" + homeType.getName() );
    return (EJBHome) PortableRemoteObject.narrow( ref, homeType );
    }
    . . .
  }
}
```

Service Locator

- Cached home proxy (stub) dangerous if server fails?
- Not likely. Most EJB servers (e.g., Weblogic, WebSphere, JBoss?) use the "Smart Proxy" pattern, so that the home proxy has fail-over logic to another server in a cluster.

Service Locator

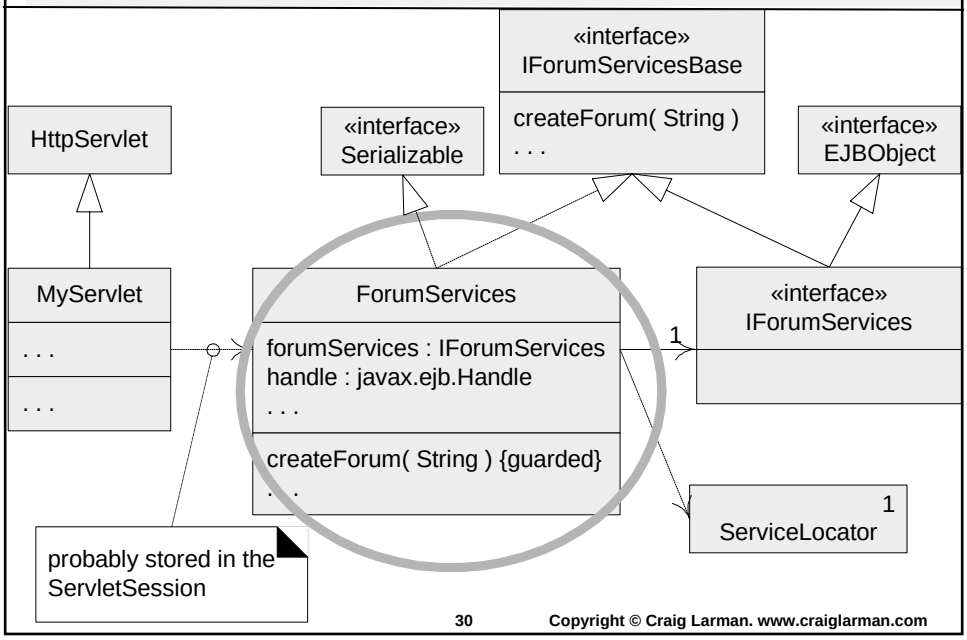
- AKA; Similar to; Variations:
  - Encapsulate Obtaining References
  - Proxy Factory
  - EJBHome Factory

This name captures the core pattern applied in many contexts

Business Delegate

- Problem:
  - How to decouple clients from the complexities of distributed components?
- If EJBs are directly used by clients:
  - Client developers require finished, working EJBs.
  - Their testing requires remote communication to the EJBs.
  - Common EJB-related error handling code is duplicated throughout the clients.
  - Exception handling is not "application level."
  - Makes refactoring to a different service layer solution more complex.
  - Caching is more difficult.

Business Delegate



Business Delegate

◆ **Also supports Convert Exceptions.**

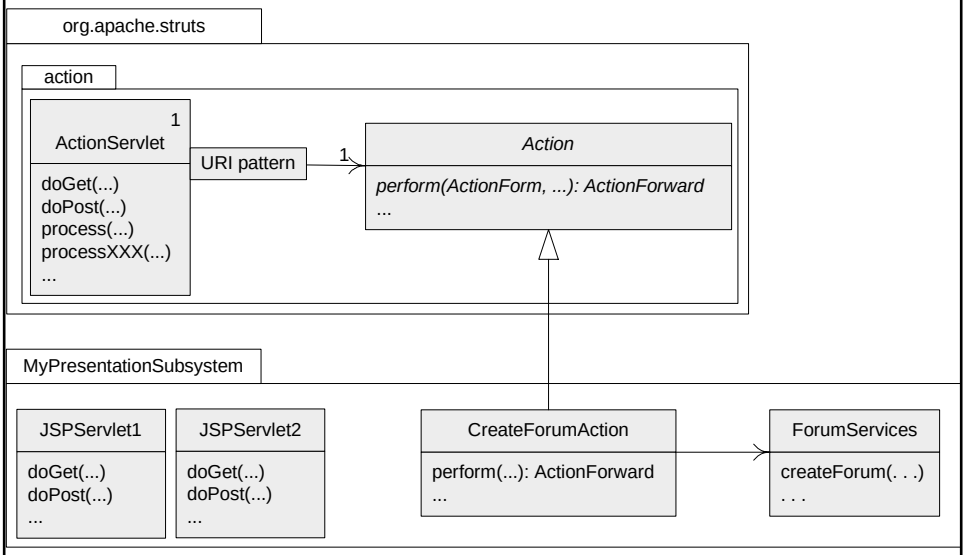
```
class ForumServices implements IForumServicesBase, Serializable {
 private IForumServices forumServices;
 . . .

 public void synchronized createForum( String forumName ) {
   try {
         forumServices.createForum( forumName );
   }
   catch( RemoteException ex ) {
      // crack open and throw appropriate ForumException. . .
      if ( . . . ) throw new DuplicateForumNameException( . . . );
   }
```


Business Delegate

◆**Notes on Java interface implementation:**

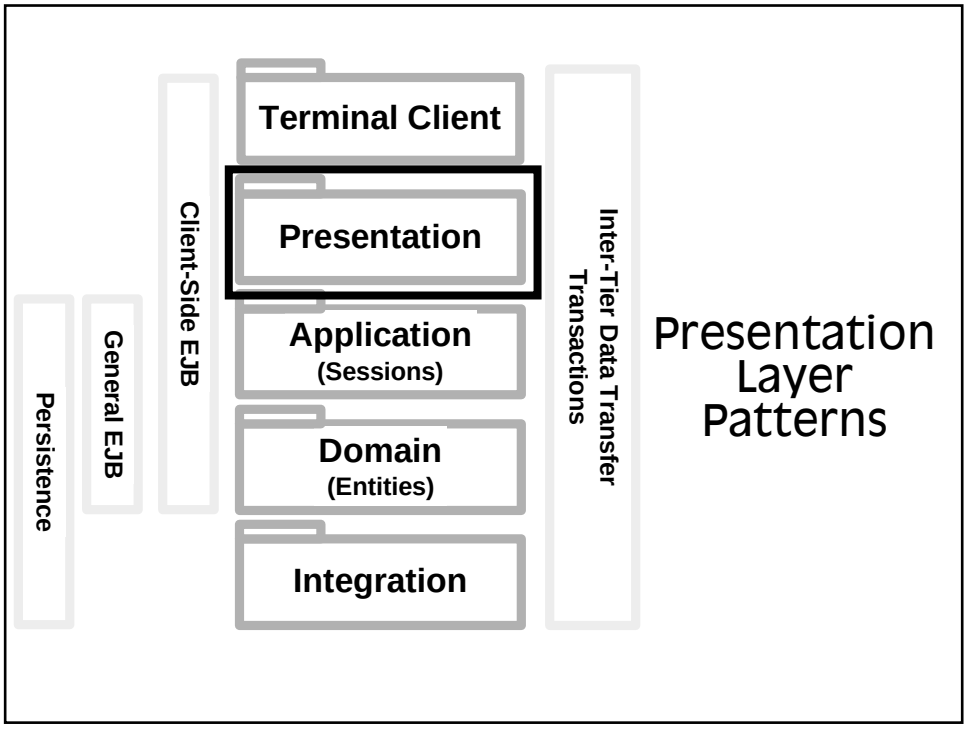
```
interface IForumServicesBase {
 public void createForum ( String forumName ) throws RemoteException;
}


class ForumServices implements IForumServicesBase {

 // legal to add “synchronized” and remove “throws”
 public void synchronized createForum( String forumName ) {
}
```


Business Delegate—Struts



Business Delegate

- AKA; Similar to; Variations:
  - Bridge
  - Proxy
  - Adapter



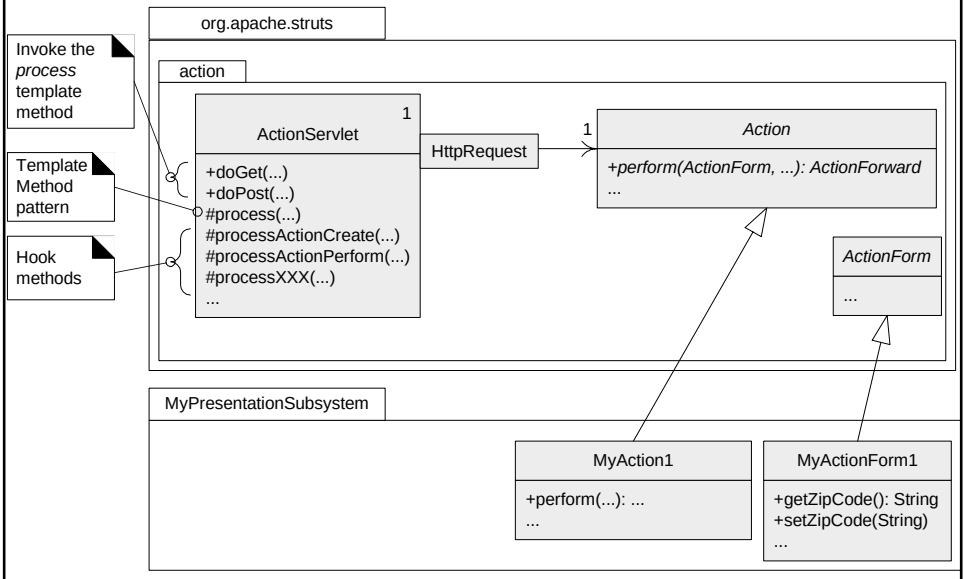
Front Controller

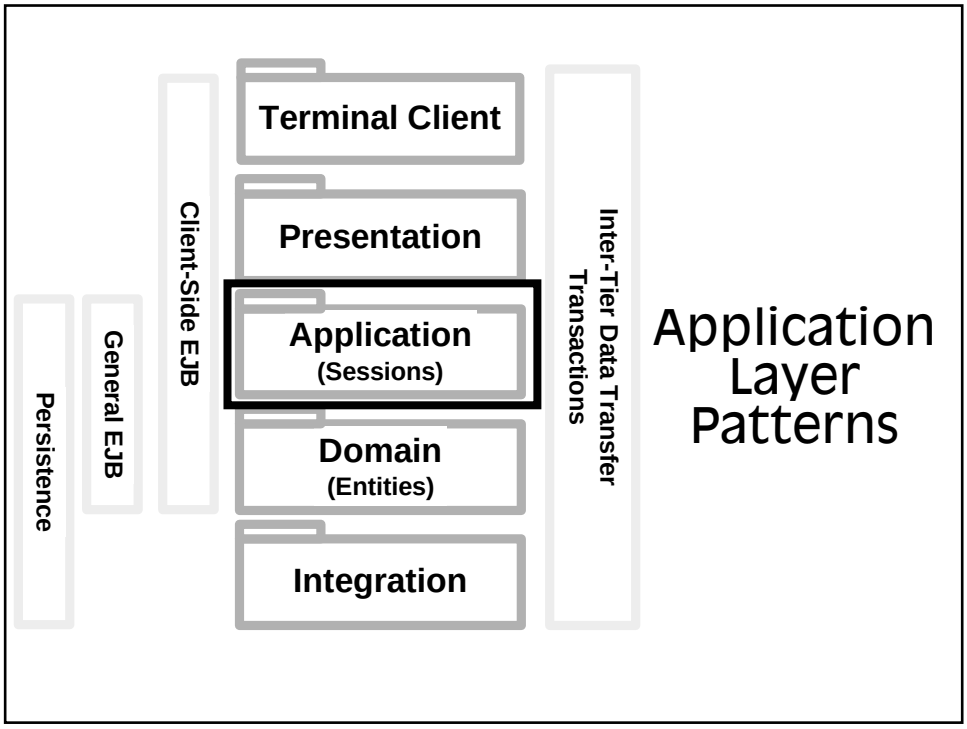
- Problem:
  - In the server-side web presentation layer, need a common point of access for services, content retrieval, view management and navigation.
- Solution:
  - Use a singleton controller as the initial point of contact.

Front Controller—Command and Controller Strategy

```
class CommandControllerServlet extends HttpServlet {

 // called from doGet and doPost

 protected void processRequest(
        HttpServletRequest request, HttpServletResponse response ) {

   Command cmd = CommandFactory.getInstance().getNewCommand(
        request.getParameter( "op" ) );

    resultPage = cmd.execute( request, response );
 }
. . .
```


Front Controller—Struts





Application Layer

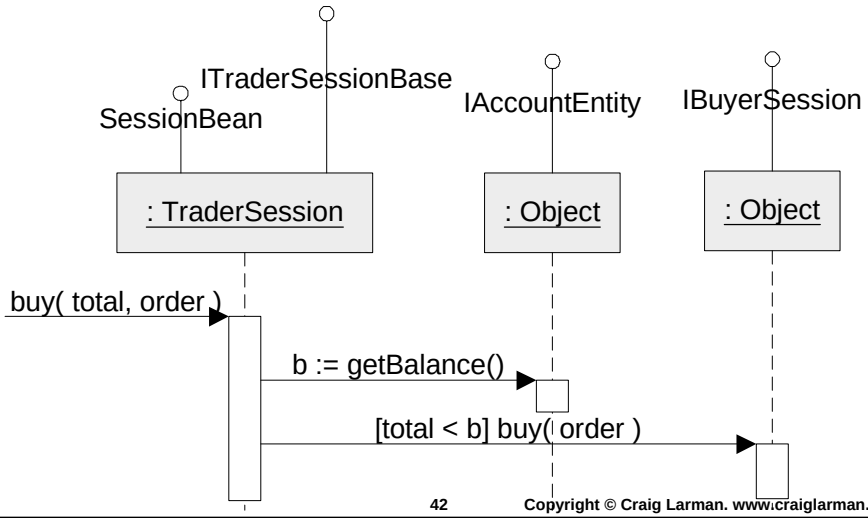
- AKA; Similar to:
  - Application +
    - Control/Controller/Coordination
    - Mediator/Mediation
    - Services
  - Workflow
  - Processes
- In other schemes, a sub-layer of:
  - Business Tier (*Core J2EE Patterns*)
  - Domain (Fowler)

Session Façade

- Problem:
  - How minimize remote proxies on client, coordinate multi-bean interaction within one transaction, and reduce coupling to the EJB components?

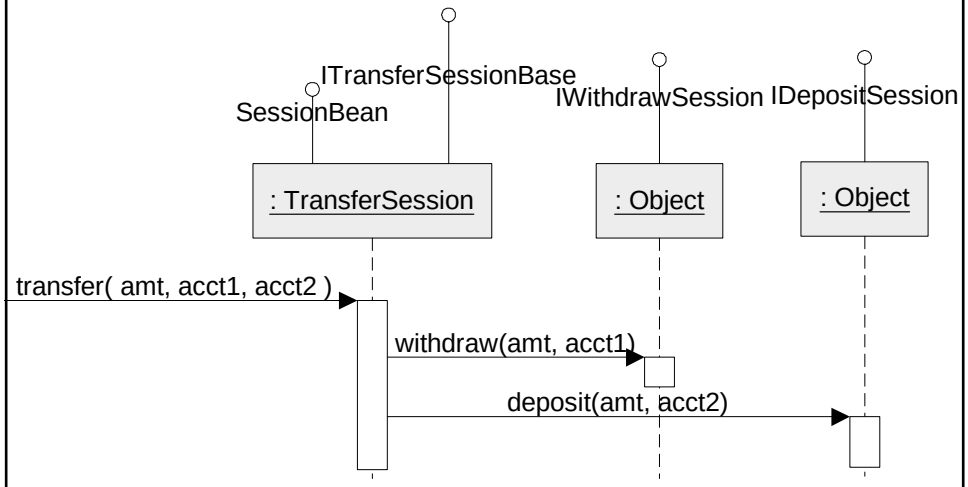
Session Façade

- Coarser-grained operations; composition of steps; wrapping Entity access.



Session Façade

- Composing multiple transactions into one.



Session Façade—Local Interfaces

```
class TraderSession implements ITraderSessionBase, SessionBean {
  private AccountLocalHome acctHome;
  public void buy( . . . ) {
    try {
          initializeHomes();
          IAccount acct = acctHome.findByPrimaryKey( . . . );
          Money balance = acct.getBalance();
          . . .
    }
  }
 private void initializeHomes( . . . ) {
    try {
          acctHome = (AccountLocalHome) ServiceLocator.getInstance().
                        getEJBHome( AccountLocalHome.class );
       }
    . . .
```

Session Façade—Consequences

- Adds a "workflow" application controller layer.
- Uniform, simpler, coarser interface.
- Reduced coupling.
- Lower network overhead.
- Centralized security and transaction management.

Session Façade

- See also; Variants
  - Session Wraps Entity
  - Use Case Session Façade
  - Application-Independent Entity
  - Façade
  - Broker

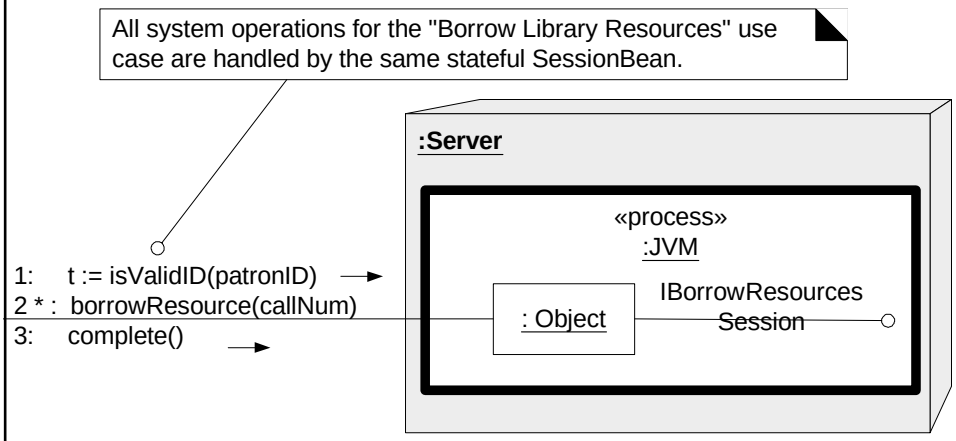
Use Case Session Façade

- Problem:
  - One approach to designing the responsibilities in a Session Bean Façade?
- Specialization of Session Façade.

Use Case Session Façade

- Solution: Group all system operations for 1 use case (or a scenario of a use case) into the same session bean Façade.
  - This is an example of the GRASP Controller pattern.
  - A different session bean for each use case.
    - Use case: Borrow Library Resources
    - SessionBean: IBorrowResourcesSession
- Session bean usually stateful.

Use Case Session Façade



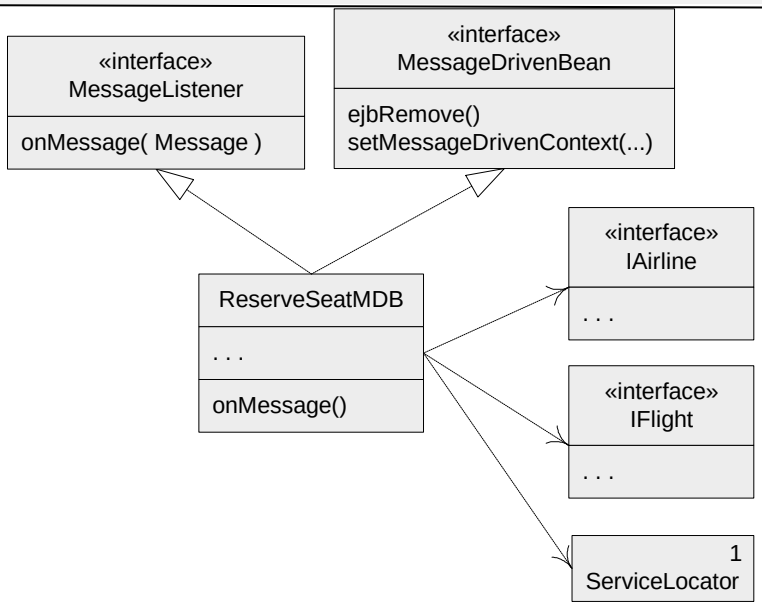
Use Case Session Façade

- Contraindications:
  - If there are too many fine-grained use cases, the Session Facades will be too fine-grained, with related performance and management issues.
    - Thus "coarsen" beyond one use case into functionally related sets, often by data operated on.
  - Some experts categorically reject this pattern, but on the (faulty) assumption that there are always too many use cases.

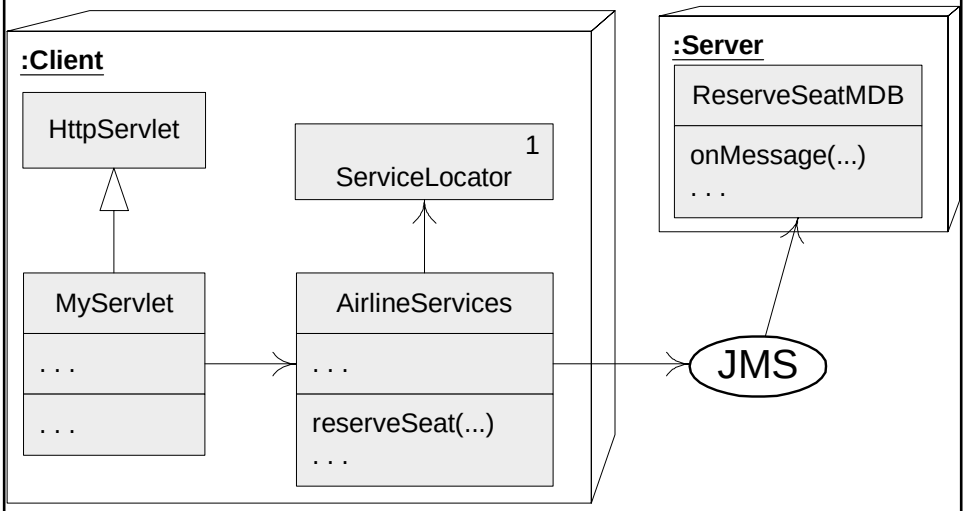
Message Facade

- Problem:
  - How can the client invoke a set of EJB operations within one transaction, yet not block?

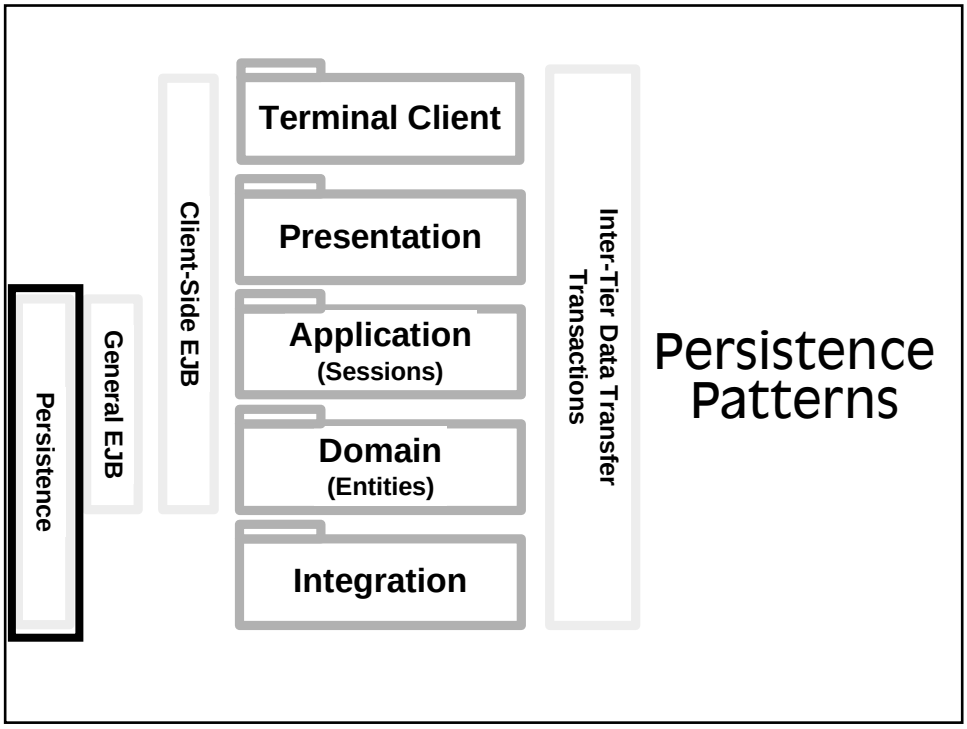
Message Facade



Message Façade—Business Delegate



Message Facade



Session Wraps Entity

- Problem:
  - When there are entity beans, how to minimize remote proxies held by the client, improve performance, and offer lower-coupled application-specific views on the data required for an application?
- Specialization of Session Façade.

Session Wraps Entity

- Solution:
  - Client has visibility to a Session Façade that:
    - provides a limited view onto the data in the entities
    - Use the local interfaces of the entities
    - adds application-specific responsibilities related to working with the data

Session Wraps Entity

- **Use local interfaces for other sessions and especially entities.**

```
class TraderSession implements ITraderSessionBase, SessionBean {
  private AccountLocalHome acctHome;
 . . .
 }
private void initializeHomes( . . . ) {
   try {
         acctHome = (AccountLocalHome) ServiceLocator.getInstance().
                              getEJBHome( AccountLocalHome.class );
     }
   . . .
```


Session Wraps Entity—Consequences

- Improved entity bean reuse/generality — Sessions can factor out any application-specific logic, leaving the entity bean with application-independent data-related responsibilities.

Session Wraps Entity—Consequences

- Lower coupling to persistence mechanism — If it is desirable to defer commitment to a persistence approach (e.g., JDBC in the session, entity beans, …), this adds the necessary level of indirection.

Session Wraps Entity—Consequences

- More flexible transaction boundary — Calls to a set of entity bean update methods can be grouped within one transaction, defined by the session.

Session Wraps Entity—Consequences

- Much Improved performance: Using the local interfaces avoids remote proxies and network communication.

Session Wraps Entity

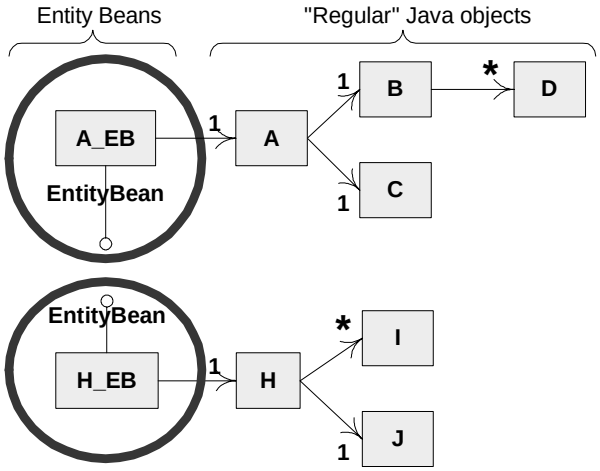
- See also:
  - JDBC for Reading
  - Data Access Object
  - Composite Entity
  - Data-Oriented Entity

Composite Entity

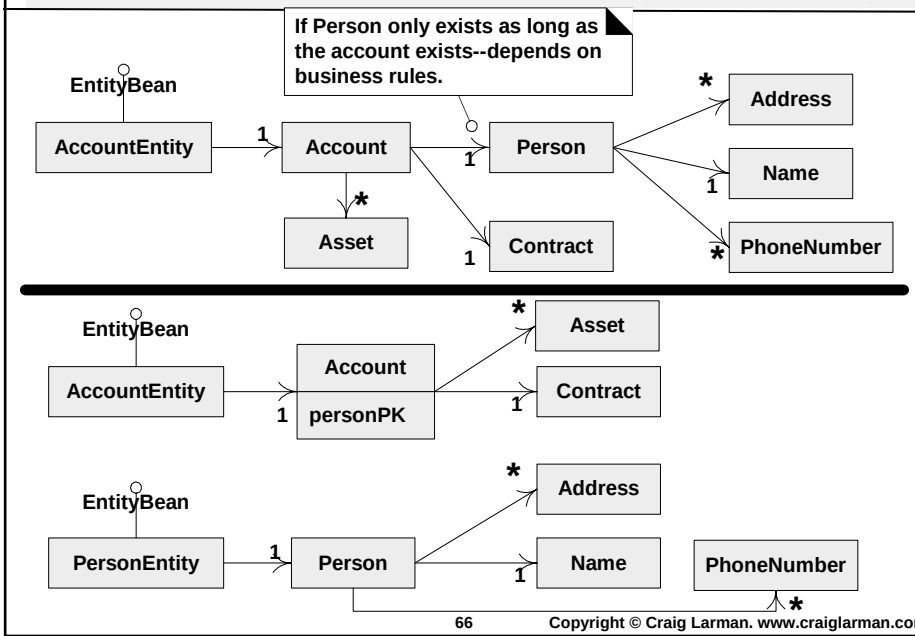
- Problem
  - You are pre-EJB 2.0 and 2.0 CMP, or using BMP. How to use design entity beans in this case?
- Motivation and Applicability
  - In general, fine-grained entity beans which represent individual persistent types (isomorphic schemas) are problematic.
    - e.g., Table <—> entity bean is not desirable
  - Performance…
  - Maintainability and management ….
  - Awkward programming…

Composite Entity

- Define an entity bean for the root of a composition hierarchy of dependent persistence types.
  - For "sub-domains" of related, dependent objects.



Composite Entity—Dependent Objects

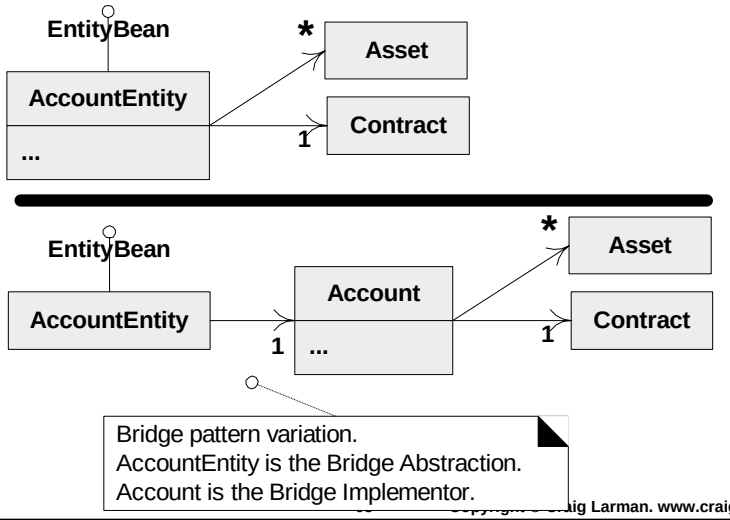


Composite Entity—EJB 2.0 CMP

- 2.0 CMP offers the opportunity (if implemented well by vendors) of high-performance fine-grained entities, based on local interfaces.
- If using 2.0 CMP, experiment with a simple "naïve" "isomorphic" entity bean schema, and see if it performs well. It may!

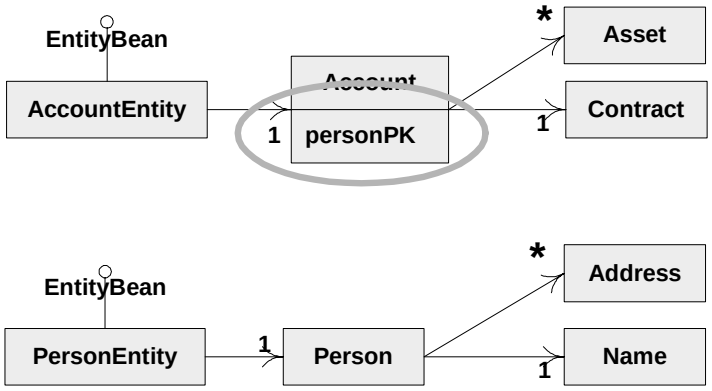
Composite Entity—Structural Variations

- The Bridge version allows deferral of deciding what will/won't be an entity bean.



Composite Entity—Relationships

- For "inter-sub-domain" relationships, the persistent types may record entity bean primary keys.
  - No direct EJB connectivity below the EntityBean level.



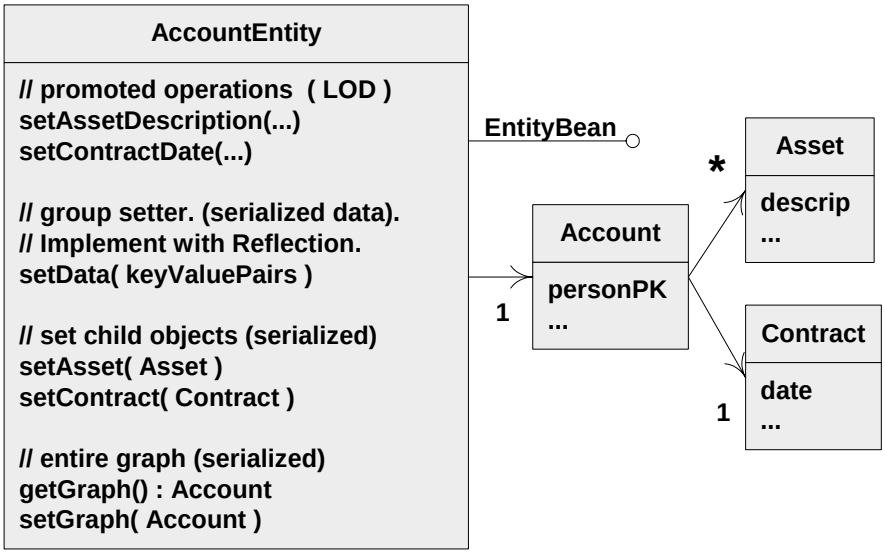
Composite Entity—EJB 1.1 Sun Conformance

- "…an entity bean should represent an independent object…" (EJB Spec 1.1)
- "A dependent object should not be implemented as an entity bean."
- A dependent object B's lifecycle and access is bound within an independent object A.

Composite Entity—Comments

- Depending on server load, access patterns, and size of graph, may be most efficient to load the entire graph at once.
  - Versus lazy materialization.

Composite Entity—Operations



JDBC for Reading

- Problem:
  - When should the server perform direct database access?
- Motivation to avoid the entity beans:
  - N + 1 DB calls on BMP and some CMP implementations.
  - If pre-EJB 2.0, remote call overhead.
  - More join operations, and relationship traversals.

JDBC for Reading

- Solution:
  - Use JDBC for reading.
- Warning: Use entity beans for inserts and updates.
  - Otherwise the rules, relationships, validations, and business logic services that entities can offer is bypassed.

JDBC for Reading

- Where?
  - In the Session Façades or Data Access Objects.
- Format of returned data?
  - Prefer RowSets.

JDBC for Reading

- Consequences:
  - Query in one read operation.
  - Optimize with related DB views.
  - Can avoid dynamic joins by exploiting DB views and/or stored procedures.
  - Can avoid transaction overhead.
  - Exploit built-in DB caching of bulk reads.
  - Retrieve the *exact* data from the DB.
  - Optimize with DB stored procedures.

JDBC for Reading

- AKA; Similar to; Variations:
  - Fast-Lane Reader

Data-Oriented Entity

- Problem:
  - How to factor application responsibilities between session and entity beans? What should an entity bean do?

Data-Oriented Entity

- Solution:
  - All application-specific responsibilities belong in session beans.
  - This pattern is a corollary of Session Wraps Entity.
  - Entity beans focus on common data-oriented responsibilities, such as:
    - Data validation
    - BMP (if necessary)
    - Structure management of the graph of dependent objects

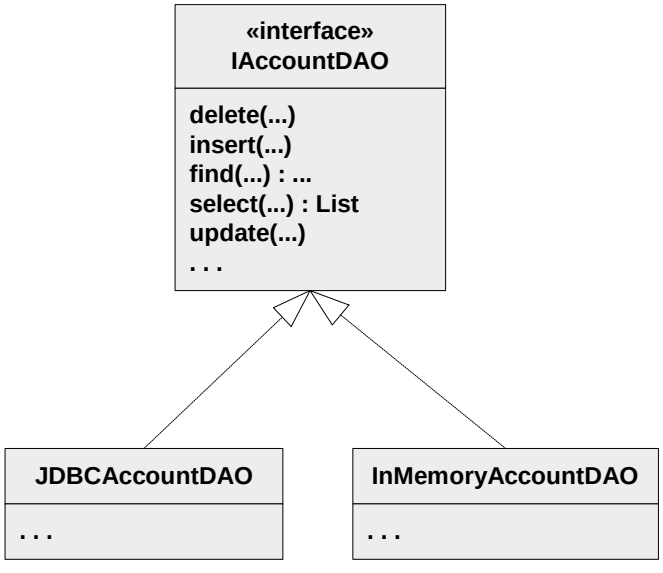
Data Access Object

- Problems:
  - Simple persistence needs; entity beans are overkill.
  - May be changing to/from entity beans, JDO, JDBC, . . .
  - Varying persistent stores for the same data (RDB, ODB, LDAP, flat file, in-memory)
  - Low cohesion and mixture of concerns if mix data and application logic in a session bean (for example).

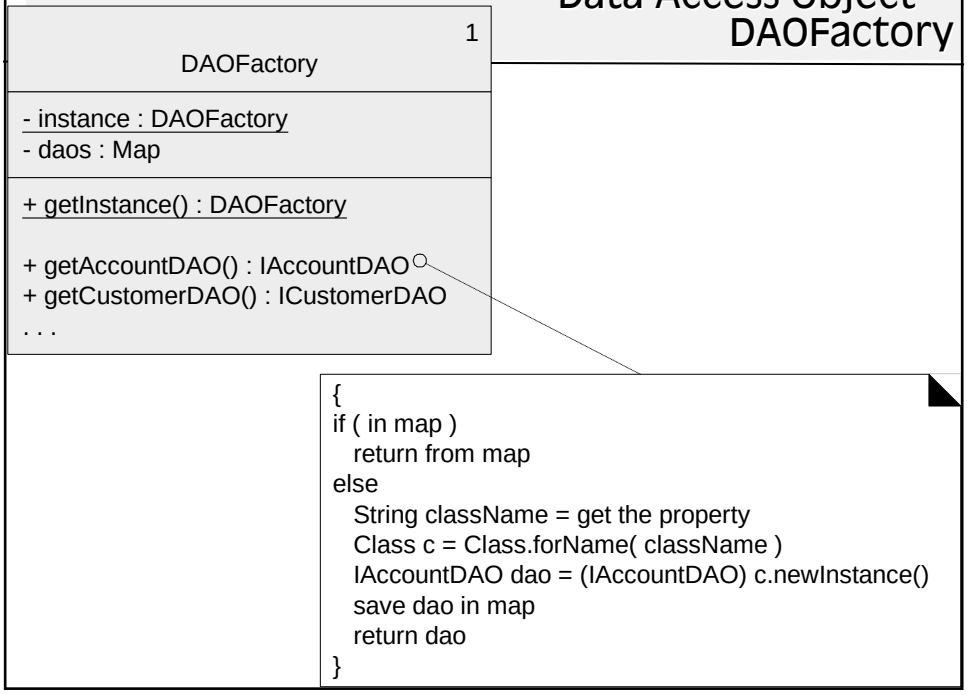
Data Access Object

- Solution:
  - Use DAOs to abstract and encapsulate data access operations.
  - May access entity beans, JDBC, in-memory creation (for testing), . . .

Data Access Object



Data Access Object—DAOFactory



Data Access Object—Type of Data?

- If don't need "object-oriented" data or type safety
  - Rowsets
  - Maps
- Else
  - Data Transfer Objects

Data Access Object

- Clients?
  - Session Facades
  - BMP entity beans
    - note there could be a DAO on both sides of a BMP entity
  - Servlets

Data Access Object

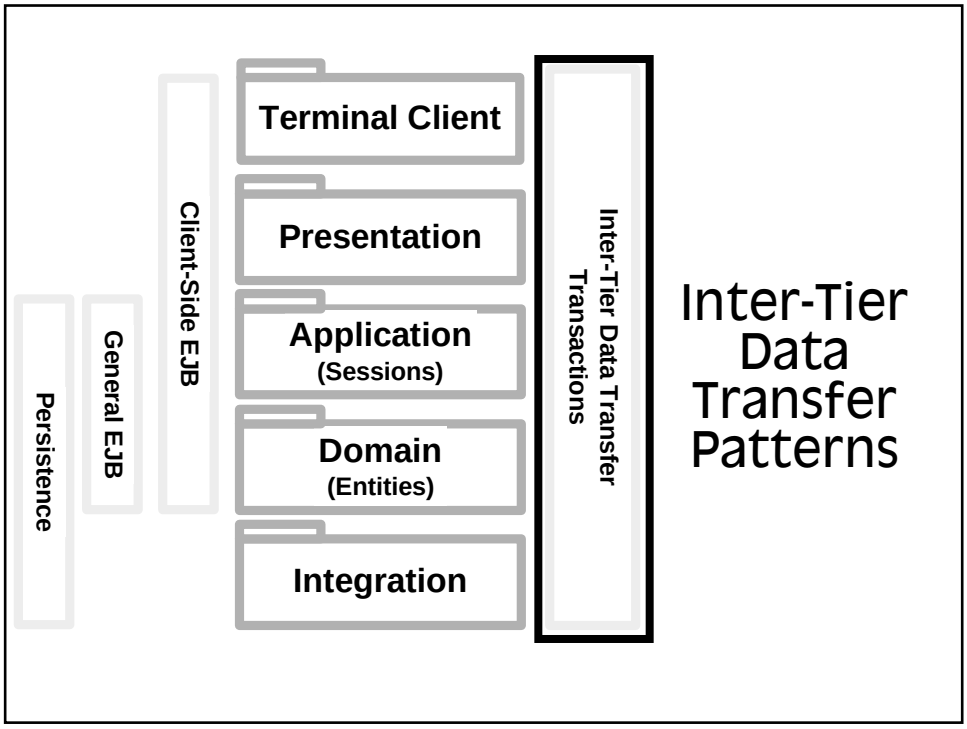
- AKA; Similar to; Variations:
  - Bridge
  - Adapter

Local Entity

- Problem:
  - Remote communication on entity beans is a performance problem, especially if the entity beans are fine-grained.

Local Entity

- Solution:
  - In EJB 2.0, local interfaces offer performant entity beans.
  - Only access entity beans via local interfaces, wrapped with a DAO and/or Session Façade.



Data Transfer Object

- Problem:
  - How to exchange data with a remote service, avoid expensive fine-grained network communication, and maintain type safety?

Data Transfer Object

- Solution:
  - Create plain Java classes that contain the data, and use a pass-by-value approach that replicates the data to a client.

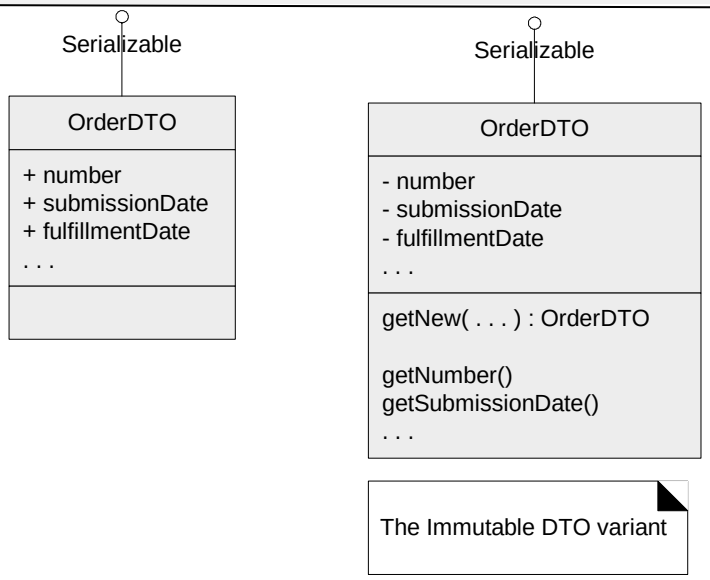
Data Transfer Object

- Advantages
  - Performance — Local invocation on the client side to access data.
  - Type safety
  - . . .

Data Transfer Object

- Limitations
  - Synchronization — Replicated data raises problems in both synchronizing data from the client to server, and vice versa.
  - If many types need DTOs, increased maintenance hassles.
  - . . .

Data Transfer Object



Data Transfer Object—Immutable DTO

- Responsibilities?
  - Limited to data access and validation.
  - There is not consensus on the validation approach, but some of the experts (e.g., Monson-Haefel) recommend the following. . .

Data Transfer Object—Immutable DTO

- No mutators (setters) on the replicate; only accessors (getters).
- There is no chance of errors made in the client resulting from programmers forgetting that the object was obtained from a remote EJB, writing code that updates its, and thinking (due to confusion) that the change is reflected remotely.

Data Transfer Object—Immutable DTO

- Validation is done via a DTO creation operation.
  - If planning to update the DTO, a new one is created

```
OrderDTO o = oldDTO.getNew( newData1, newData2 );
```


Data Transfer Object—Immutable DTO

```
OrderDTO o = oldDTO.getNew( newData1, newData2 );
```

- The new DTO is then sent to the server (e.g., to a session bean that wraps an entity).
- This also ensures the entity bean is receiving correct data, and doesn't need to redo validation.

Data Transfer Object—Validation

- Two types of validation: format and semantic.
  - Format validation can definitely be done in the DTO class.
  - Semantic validation may require collaboration with a server object (e.g., a session bean method).
    - That's OK, but put all server-side validation in 1 place, so that there are at most 2 classes that handle validation.

Data Transfer Object—Validation

- Consider collecting a set of errors in a ValidationException (a Composite exception).

```
try { OrderValues o = new OrderValues ( . . . ) ; }
catch ( ValidationException ex ) { . . . }
```


Data Transfer Object—Validation

- Note that this approach encourages placing all validation logic only in the replicate class, and if necessary, in 1 server side session bean.

Data Transfer Object

- Who creates the DTO?
  - DAO, usually
  - or DAO delegates to a DTOFactory, if DAO is becoming incohesive, or
  - The DTO itself — for the case previously described.

Data Transfer Object

- Automatic generation of DTO classes, and supporting code?
  - Yes, there are approaches. . .

Data Transfer Object

- Warnings.
  - Some discussions (e.g., at TheServerSide) recommend defining the DTO as a superclass, and the entity bean class a subclass.
  - This is not ideal, and it is invalidated in EJB 2.0, as the entity bean class is required to be abstract, and the CMP engine will automatically define the access methods.

Data Transfer Object

- AKA or minor variation of
  - Value Object
  - Info Object
  - Replicated Object
  - State Holder
  - Details Object
  - Lightweight Serialized Object
  - Data Transfer Object
  - Whew!

DTO as Map

- Problem:
  - DTOs are a hassle due to frequently changing data needs, many new DTO, and the tight coupling of the client to the DTOs.

DTO as Map

- Solution:
  - Transfer the data in a Map (e.g., HashMap).

DTO as Map

- Advantages:
  - Ease of maintenance.
  - One data object across all tiers.
- Disadvantages and Consequences:
  - Loss of type safety.
  - A contract for keys is required.
  - Primitives need wrapping.
  - Casting for all reads.

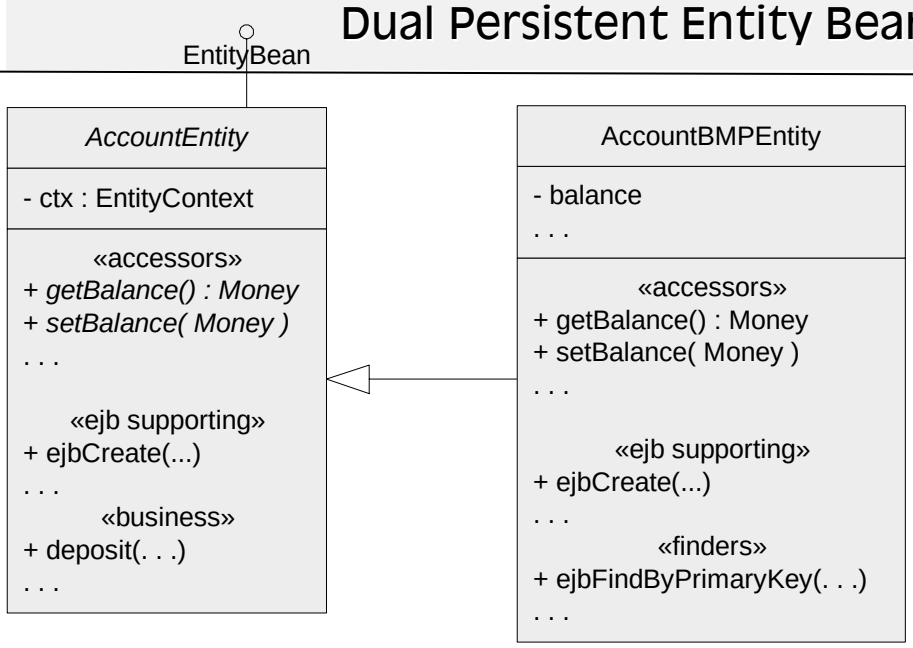
DTO as Map—Safer Keys

```
class Keys implements Serializable {
   public static final String ORDER_ID = "ID";
   . . .
}


. . .

Object value = dtoMap.get( Keys.ORDER_ID );
```


Dual Persistent Entity Bean

- Problem:
  - How to design entity beans for both CMP and BMP support?

Dual Persistent Entity Bean



UUID for EJB

- Problem:
  - Need unique keys, especially for entity beans, or database records in general.
  - Not all DBs have a key generator; there might not be a DB.
  - Want high performance without accessing a remote service (DB key generator, session object, …)

UUID for EJB

- Solution:
  - Create a universally unique ID (16 bytes; 32 digits) based on the standard UUID specification.

UUID for EJB

- Digits 1-8: hex-encoded lower 32 bits of System.currentTimeMillis().
- Digits 9-16: hex-encoded of the 32 bit integer IP address.
- Digits 17-24: hex-encoded System.identityHashCode(Object)
- Digits 25-32: random 32 bit integer from java.security.SecureRandom.

UUID for EJB

```
private int timeLow; //32 bits
 private String hexInetAddress; //32 bits
 private String thisHashCode; // 32 bits
 private String midValue;

public void initialize() {

   // initalise the secure random instance
   seeder = new SecureRandom();

     // get the inet address
   InetAddress inet = InetAddress.getLocalHost();
   byte [ ] bytes = inet.getAddress();
   hexInetAddress = hexFormat( getInt( bytes ), 8 );

     // get the hashcode
   thisHashCode = hexFormat( hashCode(), 8 );
```


UUID for EJB

```
        //...-xxxx-xxxx-xxxx-xxxx.. mid part of the sequence
        StringBuffer buffer = new StringBuffer();
        buffer.append("-");
        buffer.append(hexInetAddress.substring(0,4));
        buffer.append("-");
        buffer.append(hexInetAddress.substring(4));
        buffer.append("-");
        buffer.append(thisHashCode.substring(0,4));
        buffer.append("-");
        buffer.append(thisHashCode.substring(4));
        midValue = buffer .toString();
    }
```


UUID for EJB

```
public String getUUID()
  {
      long timeNow = System.currentTimeMillis();
      timeLow = (int) timeNow & 0xFFFFFFFF;

      int node = seeder.nextInt();

     return
       (hexFormat(timeLow, 8) + midValue + hexFormat(node, 8));
 }
```


EJB?

EJB?

- It is estimated that $1Billion USD was spent on unneeded application servers 1998-2001 (Gartner).
- What's the value of EJB?
- What the value of entity beans?
- Should JDO replace entity beans?

Contact Information

- Craig Larman
  - www.craiglarman.com
  - High-impact skills transfer for
    - Leadership
    - Developers
    - In agile processes, OOA/D, patterns

