
Agile development for Java Enterprise Applications

Prerana Patil

Agenda

- ❑ Agile- What, Why, When (-not)
 - ❑ Agile vs. Traditional
 - ❑ Agile + Java
 - ❑ Planning
 - ❑ Modeling
 - ❑ Coding (using light weight technologies)
 - ❑ Some features – CI, TDD
 - ❑ IDE tips
-

Agile software development

Adaptive

Goal Driven

Iterative

Lean

Emergent Approach

- Adaptive
 - Iterative
 - Active Customer involvement
 - Incremental development
 - Feature driven
 - Priority based delivery
 - Rapid delivery
 - Empowered Teams
 - Disciplined**
 - Is not new; Used in other industries
-

Why Agile

- Customer
 - Has higher priority
 - Gets to know regular and frequent status of the application
 - Requirements are accepted after each iteration
 - Rapid delivery; Short time-to-market esp. for key functionalities
 - Fixed timescale to deliver
 - More Testing, so better quality

 - Project Teams
 - Empowered
 - Incremental development
 - Can focus on developing the application only
 - Apply the 80/20 rule to design the workable solution
 - Frequent feedback; less rework
 - Fixed timescale so generally fixed revenue
 - Collaborative & cooperative development
-

Agile vs. Traditional

Parameter	Traditional	Agile
Requirements	Fixed	Evolve
Time & People	May vary	Fixed
Customer Involvement	Before, After	During
Negotiable	Estimates	Schedule
Testing	After code	Integrated
Feedback	After	During
Concentration on	Processes; reviews	Workable software
Focus	Plan driven	Value driven
Stages	Requirements, Design, Code, Test, Feedback	(Plan-do-adapt)*

AgileManifesto.org

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Agile Challenges

- Agile is difficult
 - Requires **TESTING & CUSTOMERS**
 - Impact management more than developers
 - **More disciplined**
 - Plan , Test , code with continuous integration (can be many times a day)
 - Higher(?) technical expertise
 - *Complete* each feature before moving on to the next
 - Needs self-discipline
 - More Planning needed
 - Frequent, focused planning
 - Adaptive plan
-

When (not!) Agile

- Isn't solution for everything
 - Organization Support:
 - Culture: open to change
 - Hierarchy: ready to empower
 - Bureaucratic : processes, steps
 - Infrastructure Support:
 - Team co-located (with customer);
 - Hardware support for CI
 - Team Support:
 - Highly skilled, flexible people
 - More testing
 - Is functionality split-able
 - Each subsection visible at UI and testable
 - Is customer available
 - Are requirements flexible
 - Is really time -constrained
 - Passion!!!
-

Agile Process

- ❑ Kick-off meetings;
 - ❑ Prioritize requirements
 - Must, Should, Might, NITI
 - ❑ Estimate relative complexity of each requirement
 - ❑ Sufficient design: using simple diagrams
 - ❑ TDD : write test first and then write code to pass the test. That's it, don't over-code
 - ❑ Shared ownership of code :
 - pair programming (do not stay in same pairs always!)
 - status meeting
 - ❑ Refractor
 - Add interface, superclass; Move class
-

Introducing Agile in Project

- Check options
 - XP(CI, TDD, AT)
 - Scrum
 - Crystal
 - DSDM
 - RUP
 - Check project : not business critical
 - Check people : mindset, skills, can ask right questions
-

Agile + Java (EE)

- Uniqueness of this combination
 - Both are evolving fast, People centric
 - Java means Complexity ; Agile means making things Simple
 - Challenges
 - Insufficient Technical & Managerial expertise
 - Java
 - slower dev., complex architecture
 - testing not easy
 - Solutions
 - Start now! Iterate & evolve.
 - Start simple!
-

Agile Modeling/ AMDD

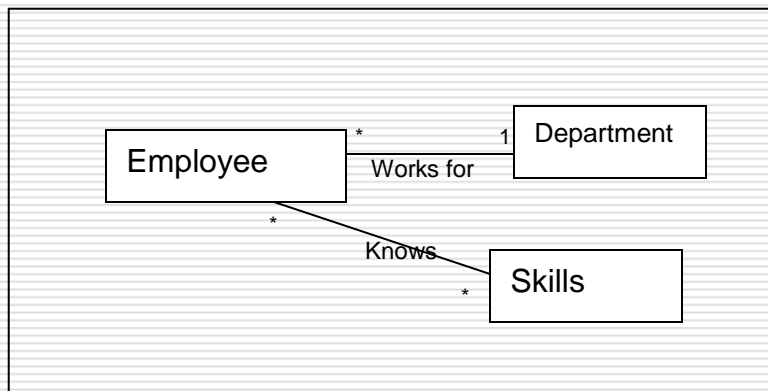
Exploration

- Domain Model
- UI Prototype
- Story Board/ User Stories
- Scope

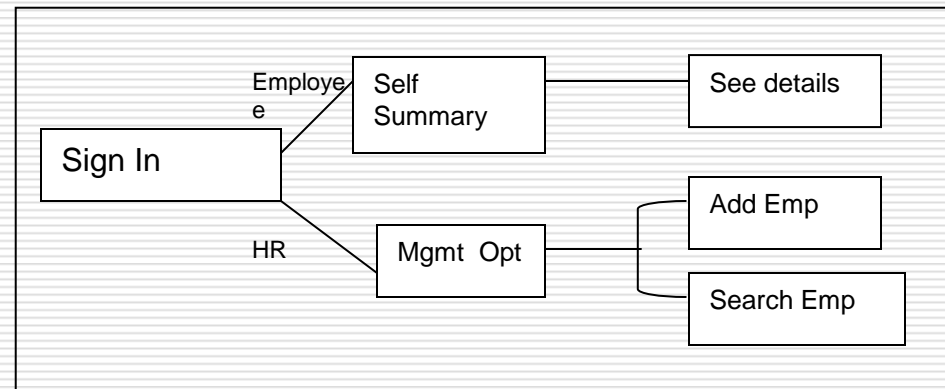
Design

- CRC Cards– Class responsibilities collaboration
 - Class diagram
 - Some UML Diagrams
-

Domain Model



Story Board (UI Flow diagram)



User Stories, Priorities & Estimates

User Story	Priority	Points
Add Employee	1	1
Edit Employee	2	1
Delete Employee	3	1
Search Emp by ID	1	2
Search by Dept	3	3
Search all in a dept knowing a skill	4	5
	Total	13

Release Plan

Iteration	Features	Release Date
0	Setup envi.	10-June-08
1	All priority 1 stories	18-June-08
2	All priority 2 stories	23-June-08
3	All priority 3 stories	..
4	All priority 4 stories	..

CRC Card

Employee	
Know Empid Know EmpName Know EmpDept	

Employee DAO	
Add Emp Search Emp	DBConnection

Agile Planning

- ❑ Ongoing
 - ❑ Iterative
 - ❑ Spread out much more evenly over the project
 - ❑ Uncertainty is key factor
 - ❑ Encourages change
 - ❑ High priority first
 - High value
 - Financial benefits to customer
 - ❑ Consider
 - Cost, Innovation/Knowledge, Risk
 - ❑ Derive Release plan
 - Precise Iteration plan
-

Iteration

□ Includes

- Development task, Estimates by developers, Plan for next iteration
- Design
- Code
- UAT

□ Scope indicates: what is included & what is deferred

Agile Estimation

- Deciding the size of the artifact
 - Relative
 - Team can come up with estimates
 - Can be derived in the meetings
 - Guestimate
 - Sample Estimate
-

Agile –Coding Process

- Prioritize task
 - Write test
 - Write code
 - Refractor
-

Java EE Rapid development using

- Ant
 - JUnit
 - Spring Framework
 - Hibernate
 - IDE
-

Ant- Build the application

- Target, property
- classpath/path
- Javac
- Mkdir/cd/copy/move
- War/jar/ear
 - Exclude
- Deployment
- Preprocess
- Mail, Remote Access

```
<ftp server="deploymentsrver"
  action="get"
  remotedir="/gnu/empappl"
  userid="anonymous"
  password="guest@guest.com"
  verbose="yes"
  binary="yes">
  <fileset file="README.emp"/>
</ftp>
```

```
<mail tolist="friend@somehost.com"
  subject="Hello!"
  from="me@myhost.com"
  mailhost="myhost.com"
  user="myuserid"
  password="mypassword"/>
```

JUnit - Erich Gamma, Kent Beck

□ *Various Assert methods*

- assertEquals
- assertFalse
- assertNotNull
- assertNotSame
- assertNull
- assertSame
- assertTrue

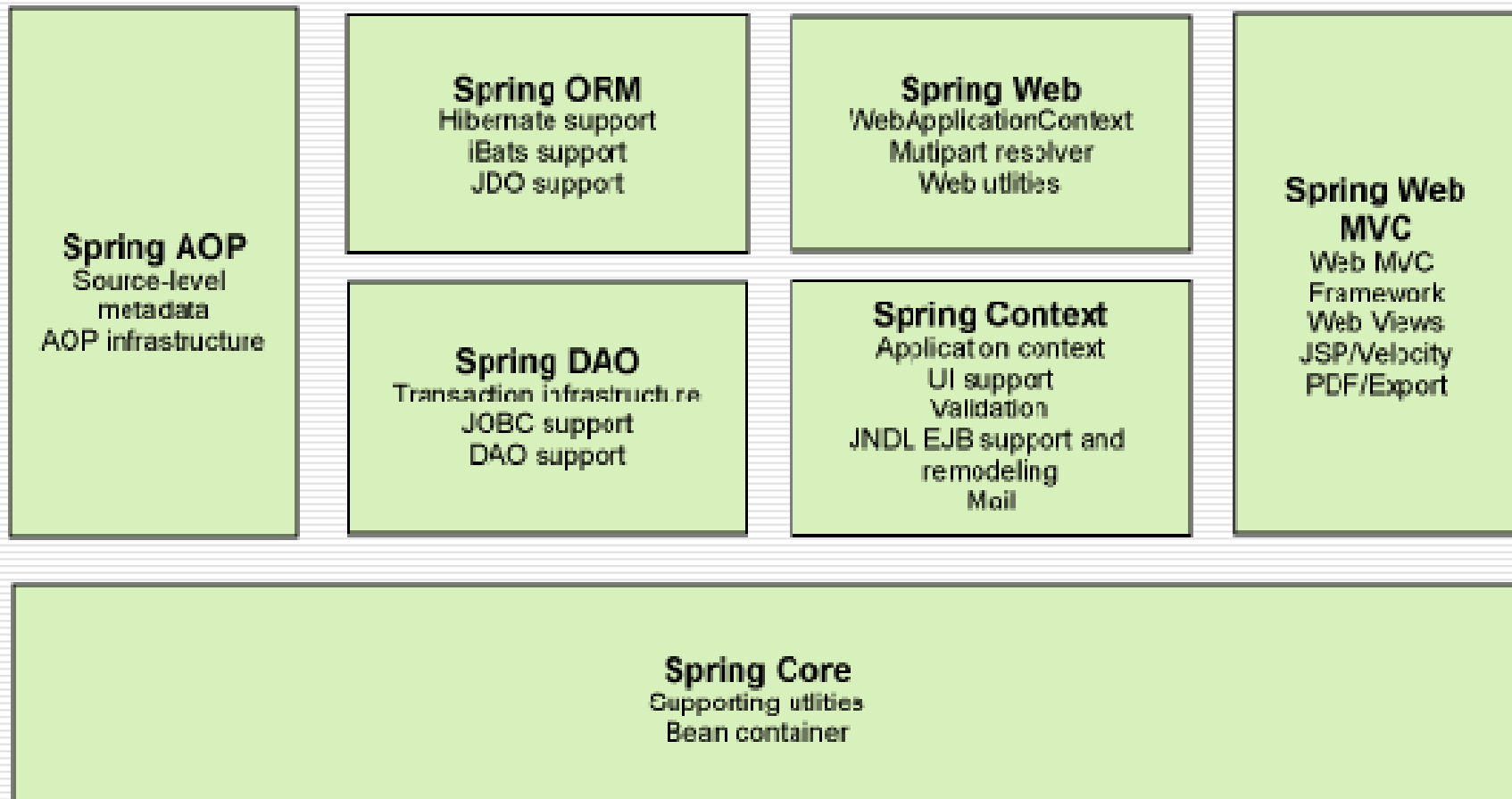
```
public class SimpleTest extends
junit.framework.TestCase
{
    int value1 = 2, value2 = 3, expectedResult = 5;

    public static void main(String args[])
    {
        junit.textui.TestRunner.run(suite());
    }

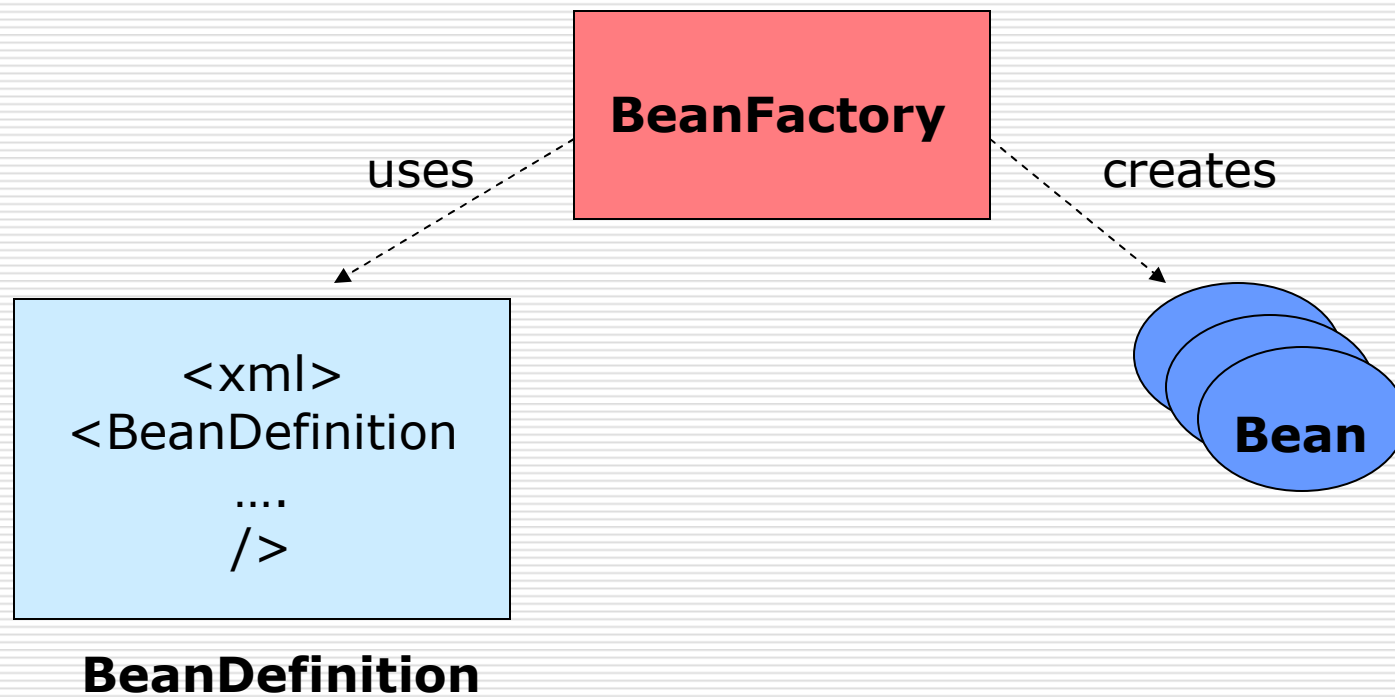
    public static Test suite()
    {
        return new TestSuite(SimpleTest.class);
    }

    public void testAddSuccess()
    {
        assertTrue(value1 + value2 == expectedResult);
    }
}
```

Spring Framework



Spring Core –IOC Container



Application Context

- Spring ApplicationContext is a subinterface of BeanFactory, which provides support for:
 - **Message lookup**, supporting internationalization
 - **Event propagation** to beans implementing the ApplicationListener interface
 - **Access to resources**, such as URLs and files
 -

```
ApplicationContext ctx = new  
    ClassPathXmlApplicationContext("emp.xml");
```

```
Employee emp = (Employee)ctx.getBean("prerana");
```

Instantiating using XML

Get XML file

```
Resource namerXmlFile = new FileSystemResource("namer.xml");
```

Load in BeanFactory

```
BeanFactory factory = new XmlBeanFactory(namerXmlFile);
```

Get bean

```
Namer namer = (Namer)factory.getBean("namerId");
```

Entry in XML

```
<bean id = "namerId"  
      class = "Namer">  
  <property name = "name">  
    <value>Prerana</value>  
  </property>  
</bean>
```

Providing bean definitions

□ Including bean definition files

```
<beans>  
  <import resource = "player.xml"/>  
  <import resource = "team.xml"/>  
</beans>
```

□ Controlling the Order of Creation of Beans

```
<bean id = "joseph" class = "Employee"  
depends-on = "admin">  
</bean>
```

□ p: namespace

```
<bean id="prerana" class="com.mycompany.Employee" p:name="Prerana"  
p:dept-ref="techprac"/>
```

□ Compound property names

```
<bean id="prerana" class=" com.mycompany.Employee ">  
  <property name="dept.location.pincode" value="411027" /> </bean>
```

Bean definition – contd.

- init-method , destroy-method
 - Singleton/ Prototype
 - Using collection elements
 - `java.beans.PropertyEditor` interface
-

JNDI Lookup

```
<bean id="dataSource"  
class="org.springframework.jndi.JndiObjectFactoryBean">  
<property name="jndiName" value="jdbc/testdatasource"  
</>  
</bean>
```

OR

```
<jee:jndi-lookup id="dataSource" jndi-  
name="jdbc/testdatasource"/>
```

Validations

```
public class PersonValidator implements Validator {  
  
    public void validate(Object obj, Errors e) {  
        ValidationUtils.rejectIfEmpty(e, "name", "name.empty");  
        Person p = (Person) obj;  
        if (p.getAge() < 0) {  
            e.rejectValue("age", "negativevalue");  
        }  
        else if (p.getAge() > 110) {  
            e.rejectValue("age", "veryold");  
        }  
    }  
}
```

Declarative Tx Mgmt

```
<tx:advice id="txAdvice" transaction-manager="txManager">
  <tx:attributes>
    <!-- all methods starting with 'get' are read-only -->
    <tx:method name="get*" read-only="true" />
    <!-- other methods use the default transaction settings (see below) -->
    <tx:method name="*" rollback-for="NoNameDetailsException" />
  </tx:attributes>
</tx:advice>
```

//linking tx:advice to class

```
<aop:config>
  <aop:pointcut id="empOperation"
    expression="execution(* x.y.Employee.*(..))"/>
  <aop:advisor advice-ref="txAdvice" pointcut-ref="empOperation" />
</aop:config>
```

Data Access

□ Defining Data Source

```
<bean id="myDataSource"  
  class="org.apache.commons.dbcp.BasicDataSource" destroy-  
  method="close" p:driverClassName="com.mysql.jdbc.Driver"  
  p:url="jdbc:mysql://localhost:3306/test" p:username="someone"/>
```

□ Linking the Data Access Object

```
<bean id="exampleDataAccessObject"  
  class="example.ExampleDataAccessObject"  
  p:dataSource-ref="myDataSource"/>
```

□ Use in Business Object

```
<bean id="exampleBusinessObject"  
  class="example.ExampleBusinessObject"  
  p:dataAccessObject-ref="exampleDataAccessObject"  
  p:exampleParam="10"/>
```

RMI using Spring

- For publishing the service

```
<bean class="org.springframework.remoting.rmi.RmiServiceExporter">  
  <property name="serviceName" value="AccountService"/>  
  <property name="service" ref="accountService"/>  
  <property name="serviceInterface"  
    value="example.AccountService"/>  
  <property name="registryPort" value="1199"/>  
</bean>
```

- For usage

```
<bean id="accountService"  
  class="org.springframework.remoting.rmi.RmiProxyFactoryBean">  
  <property name="serviceUrl" value="rmi://HOST:1199/AccountService"/>  
  <property name="serviceInterface" value="example.AccountService"/>  
</bean>  
<bean class="example.SimpleObject">  
  <property name="accountService" ref="accountService"/>  
</bean>
```

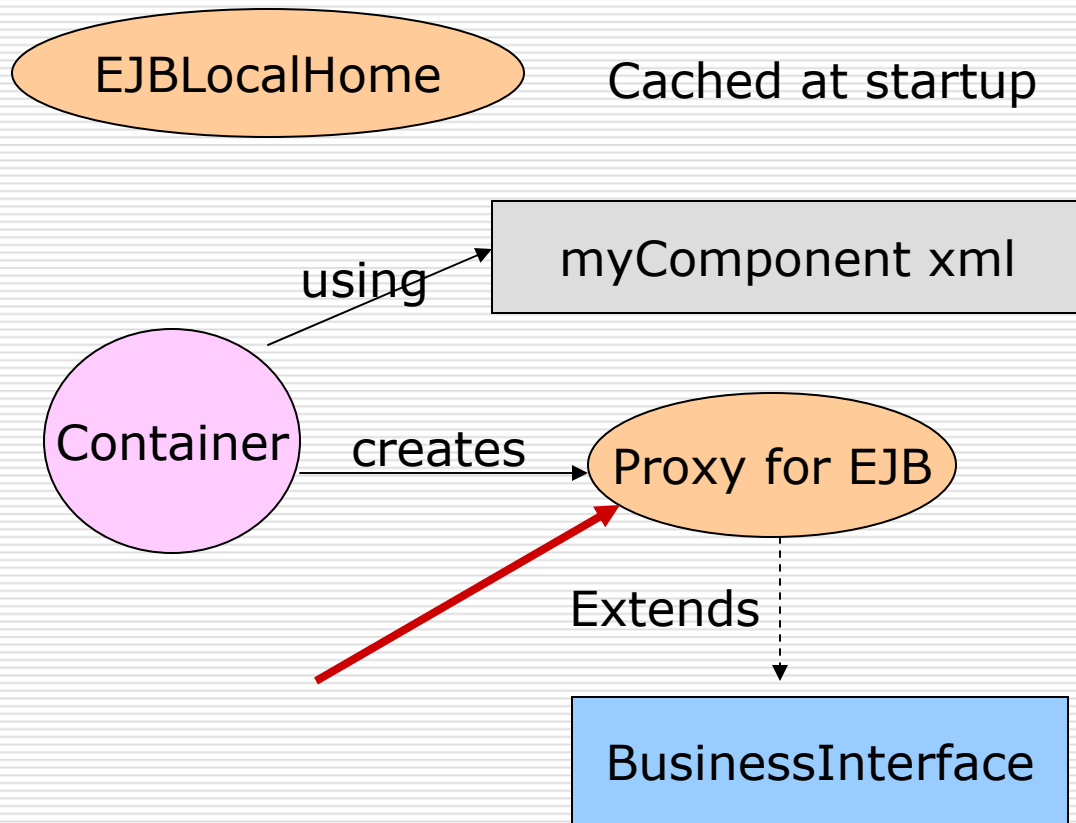
Accessing EJB

```
<bean id="myComponent"  
  class="org.springframework.ejb.access.LocalStatelessSessionProxyFactoryBean">  
  <property name="jndiName" value="ejb/myBean"/>  
  <property name="businessInterface"  
    value="com.mycom.MyComponent"/>  
</bean>
```

OR

```
<jee:local-slsb id="myComponent" jndi-name="ejb/myBean"  
  business-interface="com.mycom.MyComponent"/>
```

EJB Execution



Spring MVC

□ **Controller**

(`org.springframework.web.servlet.mvc.Controller`)

■ ModelAndView **handleRequest**(request, response)

□ **View** (`org.springframework.web.servlet.mvc.View`)

■ void **render**(model, request, response)

□ **Model**

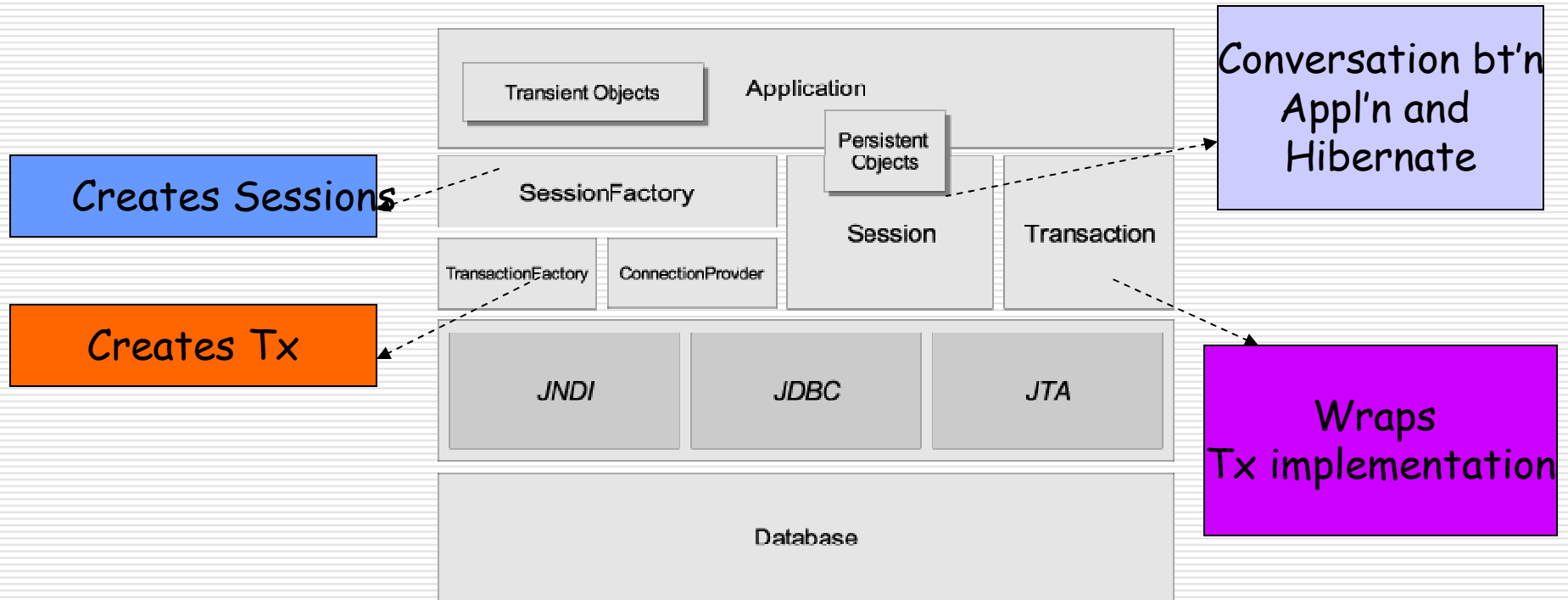
■ Typically handled as a `java.util.Map` which is returned with the view

■ in a JSP, using a `<jsp:useBean/>` where the id corresponds to the key value in the Map

Hibernate Features

- ❑ Automatic Dirty Checking , Lazy initialization
 - ❑ One-to-one, **one-to-many**, many-to-one, many-to-many
 - ❑ **Cascade** save, delete
 - ❑ Multiple-objects to single-row mapping; vise versa
 - ❑ Supports Collections objects
 - ❑ **Bulk** update, delete
 - ❑ Support for **hand-written SQL**
 - ❑ Query supports SQL **joints**, unions, functions
 - ❑ Support for update by ROWID on Oracle
-

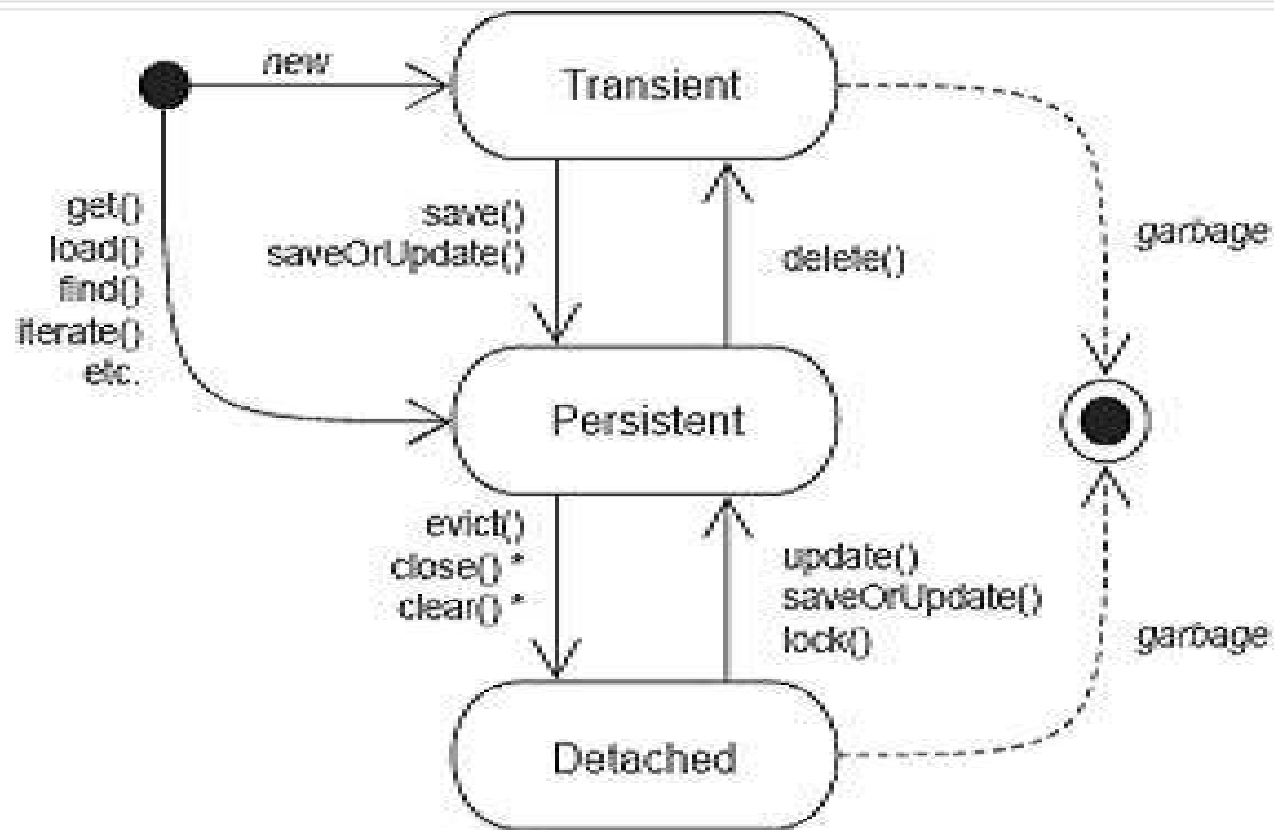
Architecture



Session

- Session - unit of work ;
 - data access operations + cache + primary db API
 - operations
 - openSession, close
 - load, persist, flush
 - get, save, update, saveOrUpdate, refresh, delete
 - CRUD
 - Session.save(object)
 - Session.update(object)
 - Session.delete(object)
 - (User)session.get(User.class, new Long(userId));
-

Object Life Cycle



Typical Hibernate Code

```
//session factory
sessionFactory = new
    Configuration().configure().buildSessionFactory();
//open session
Session session = sessionFactory.openSession();
//begin Tx
Transaction tx = session.beginTransaction();
//create object
Customer newCustomer = new Customer();
    newCustomer.setName("New Customer");
    newCustomer.setAddress("Address of New Customer");
    newCustomer.setEmailId("NewCustomer@NewCustomer.com");
//save data
session.save(newCustomer);
//commit tx , close session
tx.commit();
session.close();
```

XML Files

```
<session-factory>
  <property name="connection.username">preranadb</property>
  <property
name="connection.url">jdbc:derby://localhost:1527/myeclipse</property>
  <property name="dialect">org.hibernate.dialect.DerbyDialect</property>
<property
name="connection.driver_class">org.apache.derby.jdbc.ClientDriver</prope
rty>
  <mapping resource="demo/Emp.hbm.xml" />
</session-factory>
```

```
<hibernate-mapping>
  <class name="demo.Employee" table="EMP" schema="MYBLOG">
    <id name="id" type="java.lang.Integer">
      <column name="ID" />
      <generator class="assigned" />
    </id>
    <property name="name" type="java.lang.String">
      <column name="NAME" length="128" not-null="true" />
    </property>
  </class>
</hibernate-mapping>
```

HQL

- ❑ Select clause
- ❑ Join
- ❑ Subqueries
- ❑ Avg(),min(),max()
- ❑ Group by, Order by

Select from

```
String SQL_QUERY = "from Insurance";  
Query query = session.createQuery(SQL_QUERY);  
for(Iterator it=query.iterate();it.hasNext();){  
    Insurance insurance=(Insurance)it.next();  
    //get values }  
}
```

Criteria Query

- List all

Criteria crit =

```
session.createCriteria(Insurance.class);
```

```
List insurances = crit.list();
```

- Like

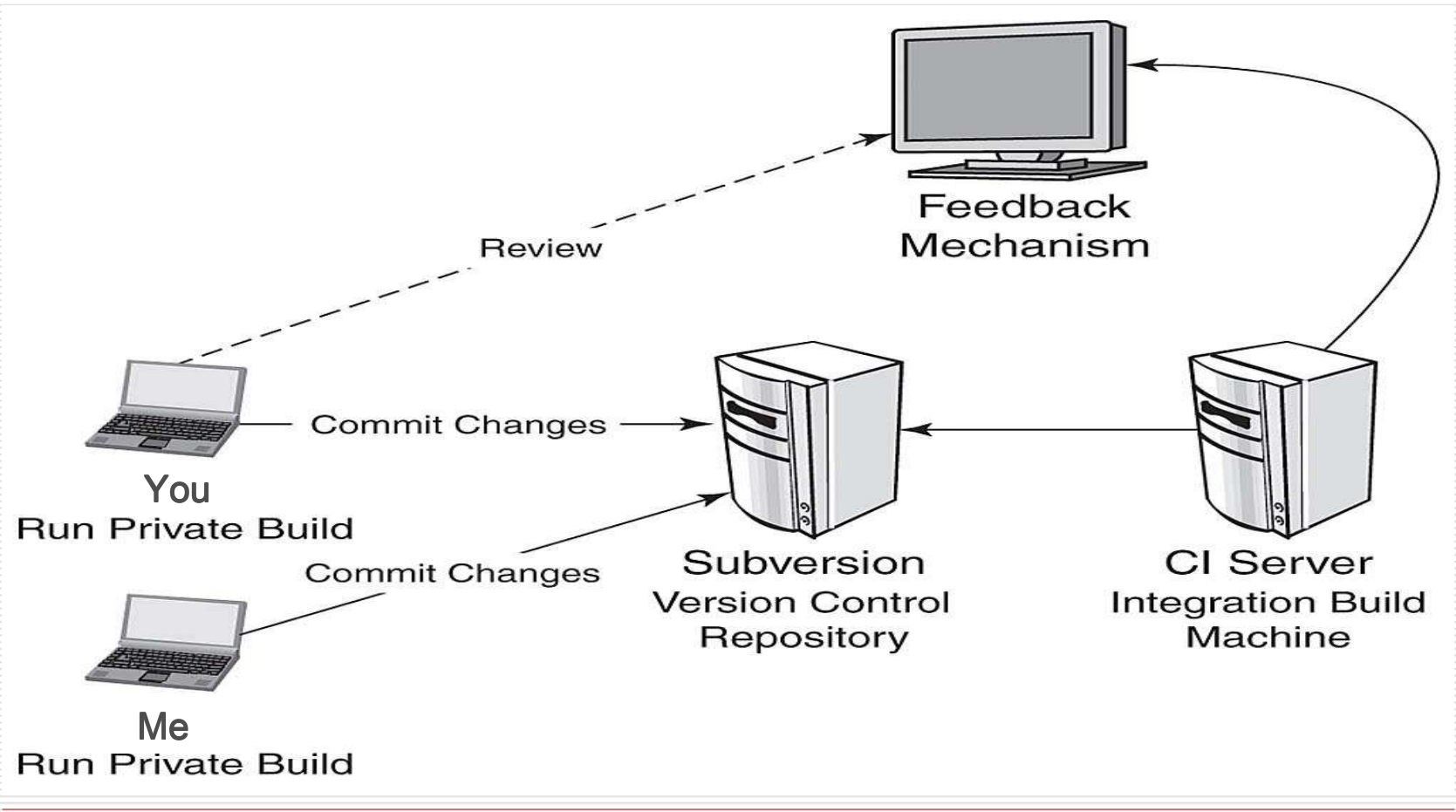
```
Criteria crit = session.createCriteria(Insurance.class);
```

```
crit.add(Restrictions.like("insuranceName", "%a%"));
```

- No of records

```
crit.setMaxResults(5);
```

Continuous Integration



CI

Process

- Follow discipline
- Private build
- Commit
- Check bug report
- Fix most imp bug

Can check

- Coding standards adherence
 - Code duplication
 - Quality metrics
 - Resolve conflicts
 - Assumptions!!
-

Good practices

- Override behavior using abstract functions
 - Sharing functionality using static fields, methods
 - Use standards
 - Bottom up designing (esp. detailed)
 - Prefer Composition over Inheritance
 - Delegate class responsibility
 - Providing Polymorphic behavior
 - Single responsibility to a class
-

Refactoring

Check

- Conditional logic
- Duplication
- Code that needs comments

Cleaning

- Renaming
 - Extract methods & base class
 - Programming by intention
 - Remove conditional logic
-

IDE tips

- Use required perspective
 - Compile immediately; Automatic Build
 - Choose editor
 - Debug
 - Source code completion help
 - Quick fix
 - Refactoring help
 - TODO
 - Reverse engineering
 - Shortcuts
-

IDE tips

- Can Integrate with
 - Spring
 - Mapping files
 - Flow/ Wizards
 - Hibernate
 - Mapping files
 - Struts
 - JUnit
 - Source Control
 - Database
 - Services/ Web Services
-

Eclipse

- ❑ 1 click switching between editors
- ❑ Editors for various file types
- ❑ Configure Output folder
- ❑ Add Libraries
- ❑ Ctrl +1 Quick fix
- ❑ Ctrl + spacebar – Assist
- ❑ Templates
- ❑ Override /Implement Methods:
 - Alt+Shift+S,V
- ❑ Generate Getters and Setters:
 - Alt+Shift+S,R

Toggle Comment	Ctrl+/
Remove Block Comment	Ctrl+Shift+}
Generate Element Comment	Alt+Shift+J
Correct Indentation	Ctrl+I
Format	Ctrl+Shift+F
Add Import	Ctrl+Shift+M
Organize Imports	Ctrl+Shift+O
Sort Members...	
Clean Up...	
Override/Implement Methods...	
Generate Getters and Setters...	
Generate Delegate Methods...	
Generate hashCode() and equals()...	
Generate Constructor using Fields...	
Generate Constructors from Superclass...	
Externalize Strings...	

Eclipse – Surround code

- ❑ Select a piece of code
- ❑ Press Ctrl+Shift+Z
- ❑ Choose how the selected code should be wrapped

```
public static void main(String[] args) {  
    System.out.println("hello, world");  
}
```

Try/catch Block

publ

- 1 do (do while statement)
- 2 for (iterate over array)
- 3 if (if statement)
- 4 runnable (runnable)
- 5 synchronized (synchronized block)
- 6 try (try catch block)
- 7 while (while loop with condition)

od stub

Configure Templates...

Questions??

Thank You!

Contact:

prerana.patil@gmail.com
