Java Data Objects (JDO)

Paul McKinney
EDS / Sun Microsystems

Java Data Objects (JDO)

- Overview
- Interfaces and helper class
  - PersistenceManager
  - PersistenceMangerFactory
  - Transaction
  - Extent
  - Query
  - JDOHelper

Java Data Objects (JDO)

- Datastore Mapping
- Configuration - Setup
- Class Enhancement
- Transactions
- Queries
- Identity
- Lifecycle states

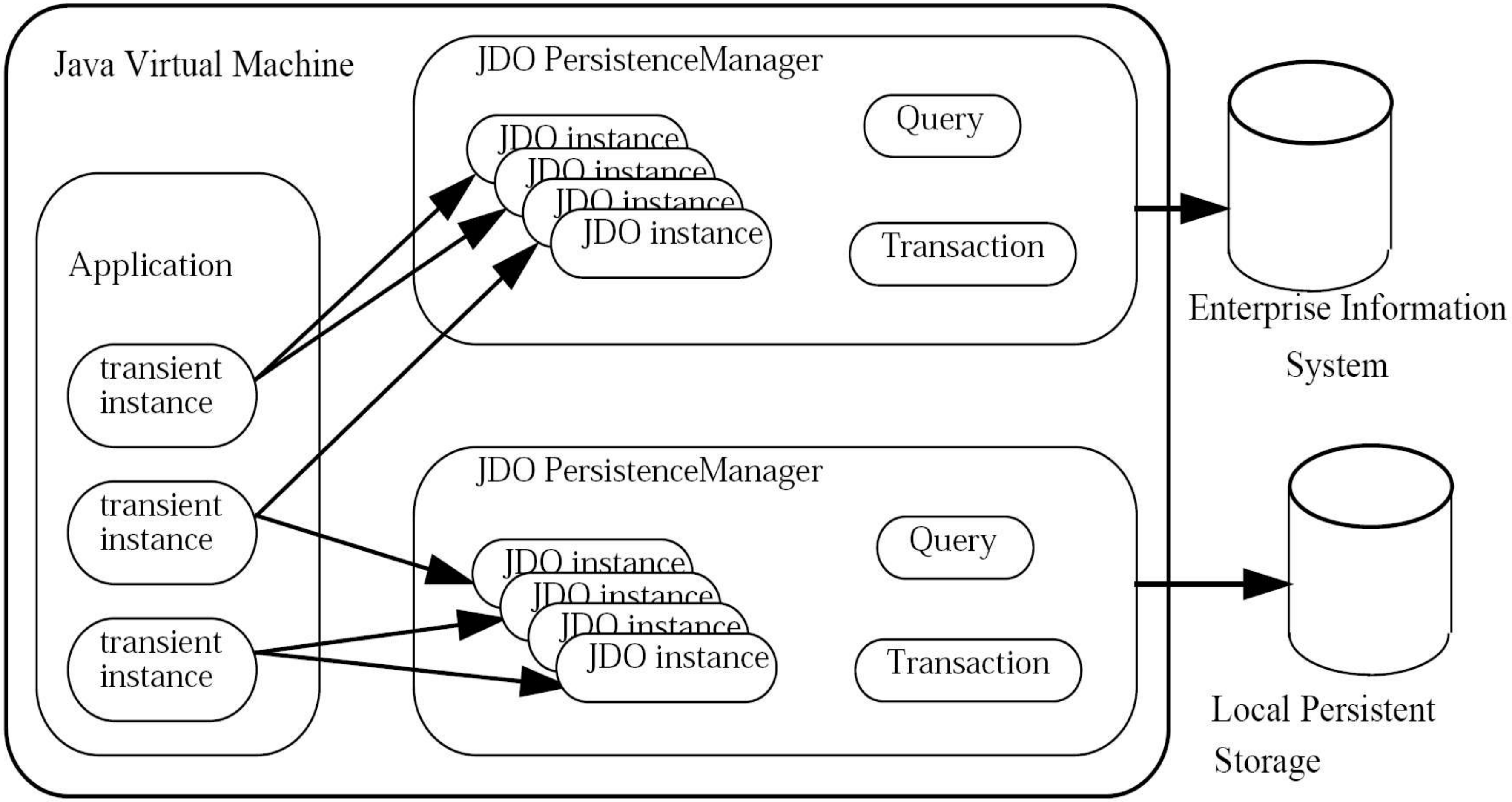
Java Data Objects (JDO)

- JDO 1.0.1 Opinion
- JDO 2.0 Features
- Un-architected JDO Demo
- Spring
  - IoC – Dependency Injection
  - Quick framework overview
- Architected JDO Example

Overview

- JDO is a specification that describes a way to persist objects in a datastore independent way.
- Java developers are allowed to create their domain model in a fully object oriented (composition, inheritance) way and JDO provides a way to persist that domain model.

Overview - Architecture



Interfaces

- PersistenceManager
  - Primary interface when using JDO
  - Used to create query and transaction objects.
  - Manages the lifecycle of persistent instances.
- PersistenceManagerFactory
  - Creates and configures PersistenceManager.
  - Helps create JDO runtime environment

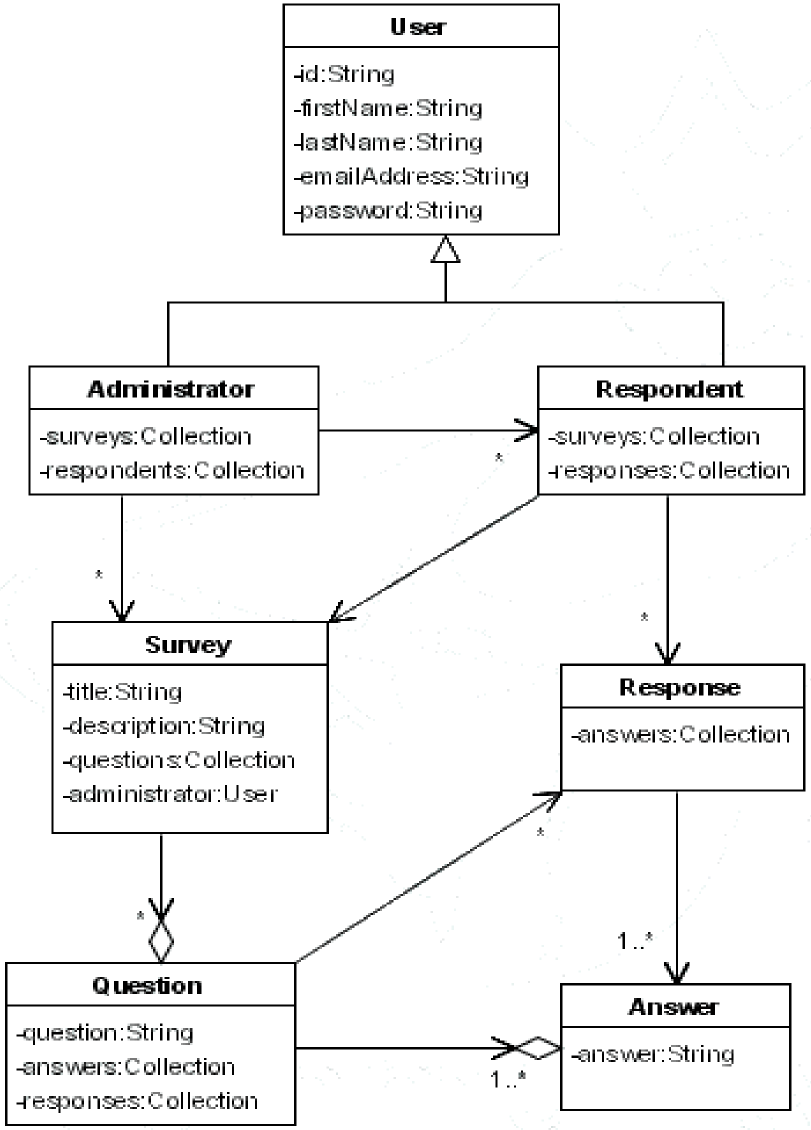
Interfaces and Helper Class

- JDOHelper
  - Provides static utilty methods.
  - Creates PersistentManagerFactory
- Transaction
  - Provides methods to manages the demarcation of transactions
- Extent
  - Used to access all instances of a class

Intefaces

- Query
  - Evaluates a filter expression when querying for persistent instances.

Domain Model – Survey System



Datastore Mapping

```
<jdo>

   <package name="survey.domain">

      <class name="User"
             identity-type="application"
             objectid-class="survey.domain.keys.UserKey">

         <field name="id" persistence-modifier="persistent"
                         primary-key="true">
          <extension vendor-name="jpox" key="length" value="max 40"/>
         </field>

         <field name="firstName" persistence-modifier="persistent">
          <extension vendor-name="jpox" key="length" value="max 20"/>
         </field>

         <field name="lastName" persistence-modifier="persistent">
          <extension vendor-name="jpox" key="length" value="max 40"/>
         </field>
      .
      .
      .
```

Datastore Mapping

```
<class name="Administrator"
     objectid-class="survey.domain.keys.AdministratorKey"
     persistence-capable-superclass="survey.domain.User">

   <field name="surveys">
      <collection element-type="Survey"/>
   </field>

   <field name="respondents">
      <collection element-type="Respondent"/>
   </field>

</class>

      .
      .
      .
```

Datastore Mapping

```
<class name="Survey" identity-type="datastore">

   <field name="title" persistence-modifier="persistent">
      <extension vendor-name="jpox" key="length" value="max 40"/>
   </field>

   <field name="description" persistence-modifier="persistent">
      <extension vendor-name="jpox" key="length" value="max 200"/>
   </field>

   <field name="questions" persistence-modifier="persistent">
      <collection element-type="Question"/>
   </field>

</class>

      .
      .
      .
```

Configuration - Setup

```
Properties properties = new Properties();

// Set the PersistenceManagerFactoryClass.
properties.setProperty(
   "javax.jdo.PersistenceManagerFactoryClass",
   "org.jpox.PersistenceManagerFactoryImpl");
properties.setProperty(
    "javax.jdo.option.ConnectionDriverName",
    "com.mysql.jdbc.Driver");
properties.setProperty(
    "javax.jdo.option.ConnectionURL",
    "jdbc:mysql://localhost/demo1");
properties.setProperty("javax.jdo.option.ConnectionUserName", "root");
properties.setProperty("javax.jdo.option.ConnectionPassword", "");
properties.setProperty("org.jpox.autoCreateTables", "true");
properties.setProperty("org.jpox.validateTables", "false");
properties.setProperty("org.jpox.validateConstraints", "false");


pmf = JDOHelper.getPersistenceManagerFactory(properties);
pm = pmf.getPersistenceManager();
```

Class Enhancement

- Classes to be persisted are required to implement the PersistenceCabable interface. The interface defines a set of methods that the JDO implementation uses to manage instances.
- Enhancement can be done manually or by a source or byte code enhancer.
- Adds code to mediate access to fields.

Queries

- Performed using the Query interface
- JDO Query Language (JDOQL) used to access persistent instances based on specified search criteria.
  - provides language neutrality
  - allows implementation to provide datastore-specific query optimizations

Queries - Code

```
// load Administrator
try {
   tx.begin();

   Extent extent = pm.getExtent(Administratorclass, false);
   String filter = "id == parmId";
   Query query = pm.newQuery(extent, filter);
   query.declareParameters("String parmId");
   query.declareImports("import java.lang.String;");
   Collection result = (Collection) query.execute("pm143527");

   Iterator iter = result.iterator();
   while (iter.hasNext()) {
      admin = (Administrator) iter.next();
   }
```

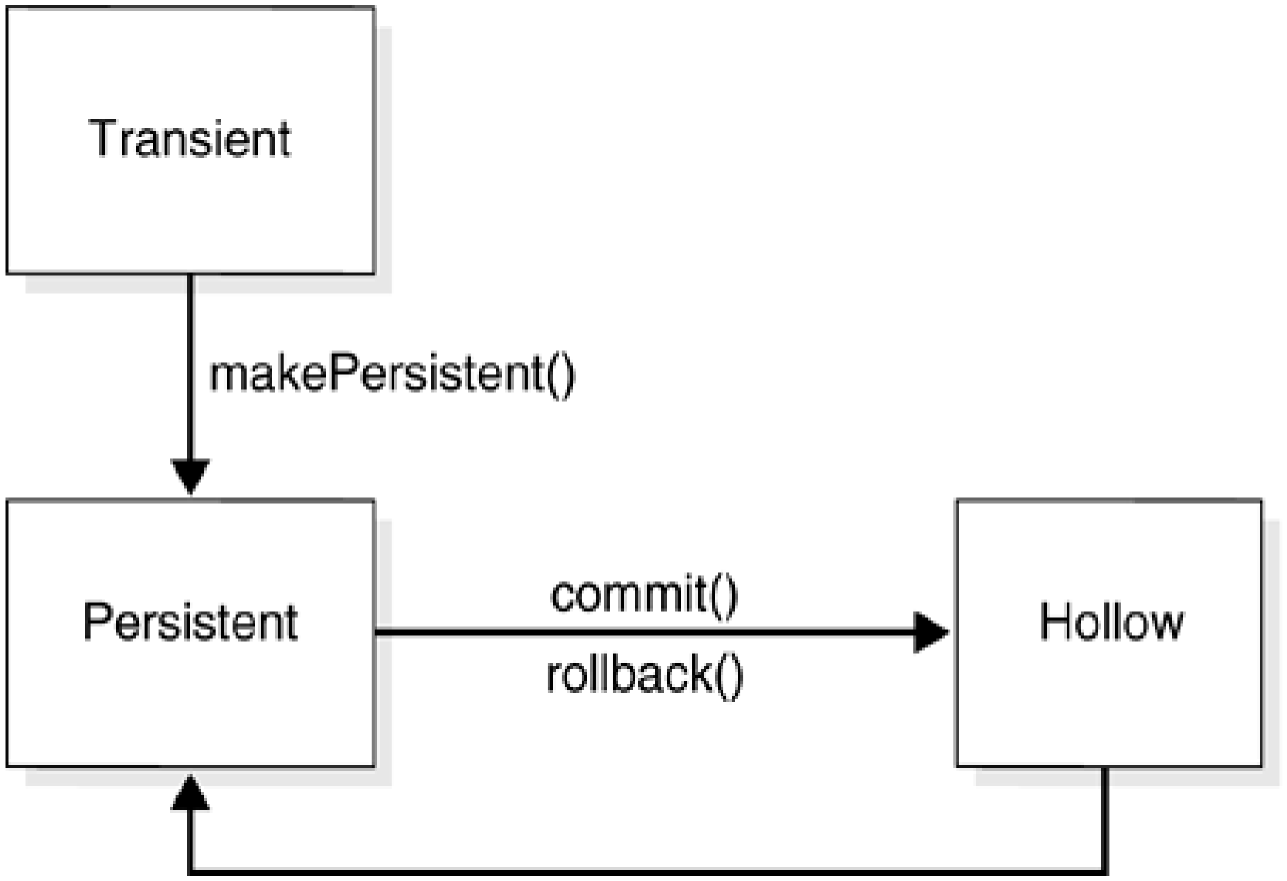
Transactions

- Access and updates are performed in the context of a transaction.
- One-to-one relationship between a PersistentManager and a Transaction.
- begin() to begin a transaction; commit() or rollback() to end a transaction

Identity

- Datastore identity
  - identity managed managed by JDO or the datastore
- Application identity
  - identity is managed by the application
  - composed of one or more primary-key-fields
  - must define an application identity class with fields that match the primary-key-fields

Lifecycle State Diagram -VERY Simplified



Lifecycle States

- Transient
  - normal non-persistent object
  - how a persistent object starts it life
- Persistent
  - instance made persistent when makePersistent () is called
  - has an associated object identity
- Hollow
  - a persistent instance whose fields have not been retrieved from the datastore

Un-Architected Demo

JDO 1.0.1 - Opinion

- Less mature than Hibernate
- Would not use an open-source solution of JDO at this time.
- JDOQL lacks aggregate functions (min, max, count)
- PIA to perform updates of domain objects in a web application.

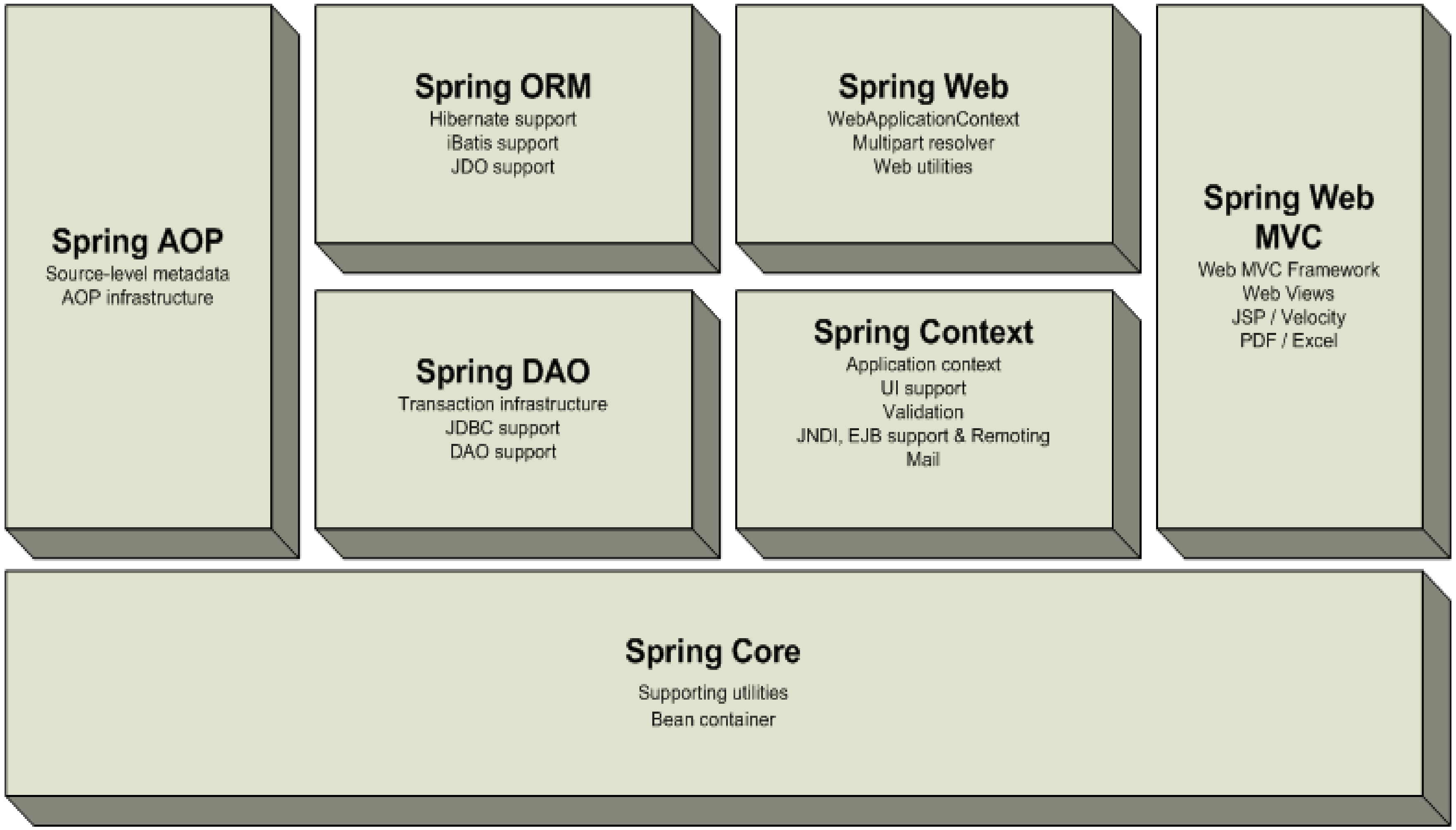
JDO 2.0 – New Features

- Addresses shortcomings in JDO 1.0.1
- Will have attach/detach capability.
- JDOQL will include aggregates and will have named queries.
- Will include an “official” escape hatch for running SQL if need be.
- Standardized O/R mappings.

Spring Framework

- A lightweight J2EE framework container utilizing inversion of control.
- At it's core, utilizes “bean factories” to wire together and manage relationships between objects.
  - singleton
  - prototype
- Promotes the use of well defined layers.

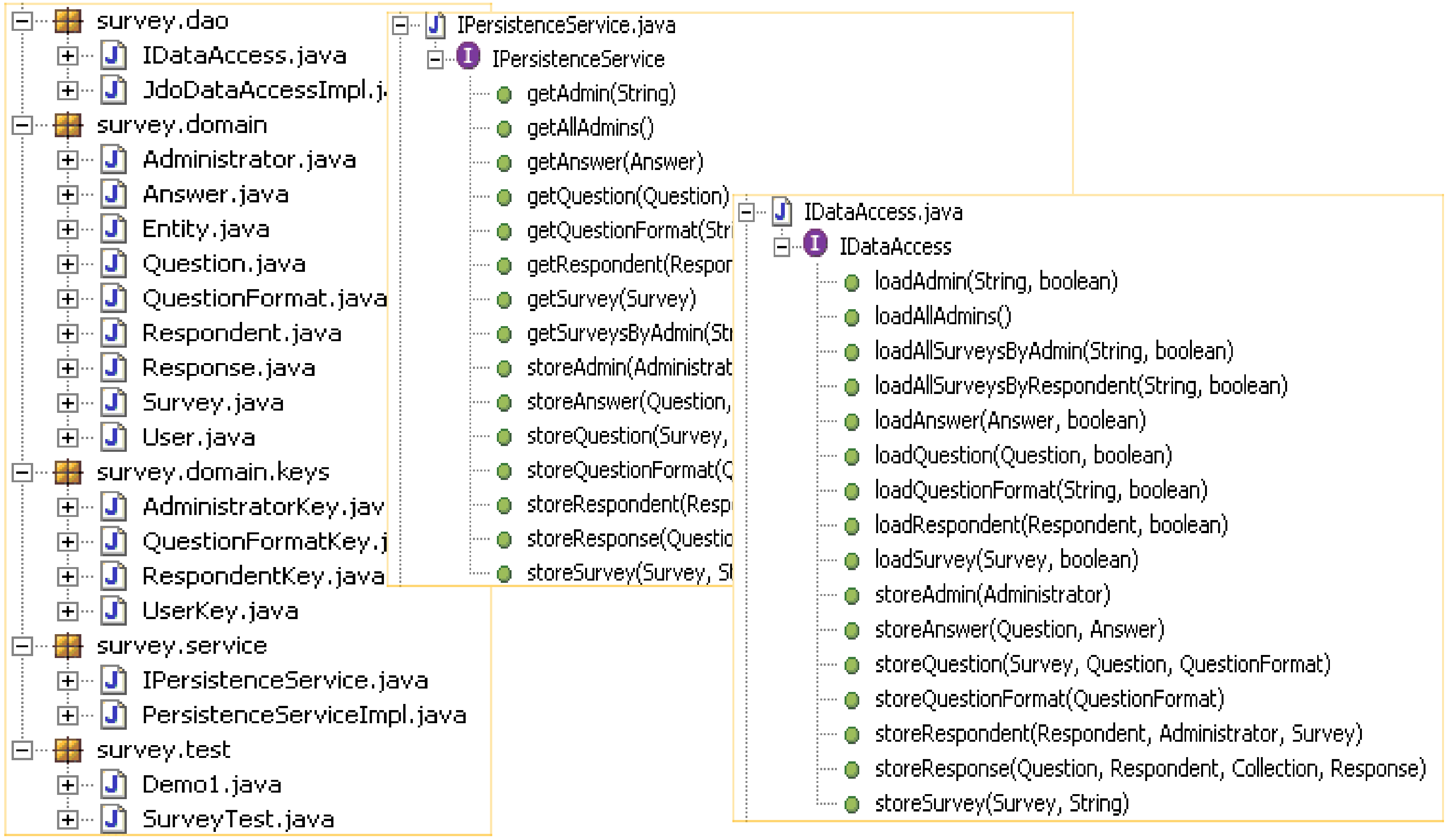
Spring Framework - Continued



Persistence Layer Demo
Survey System

- Built using a service and a DAO layer.
- The application layer, or JUnit tests in this case, talk to the service layer and the service layer talks to the DAO layer.
- The service layer is used to coordinate transactions.

Survey System Persistence Layer

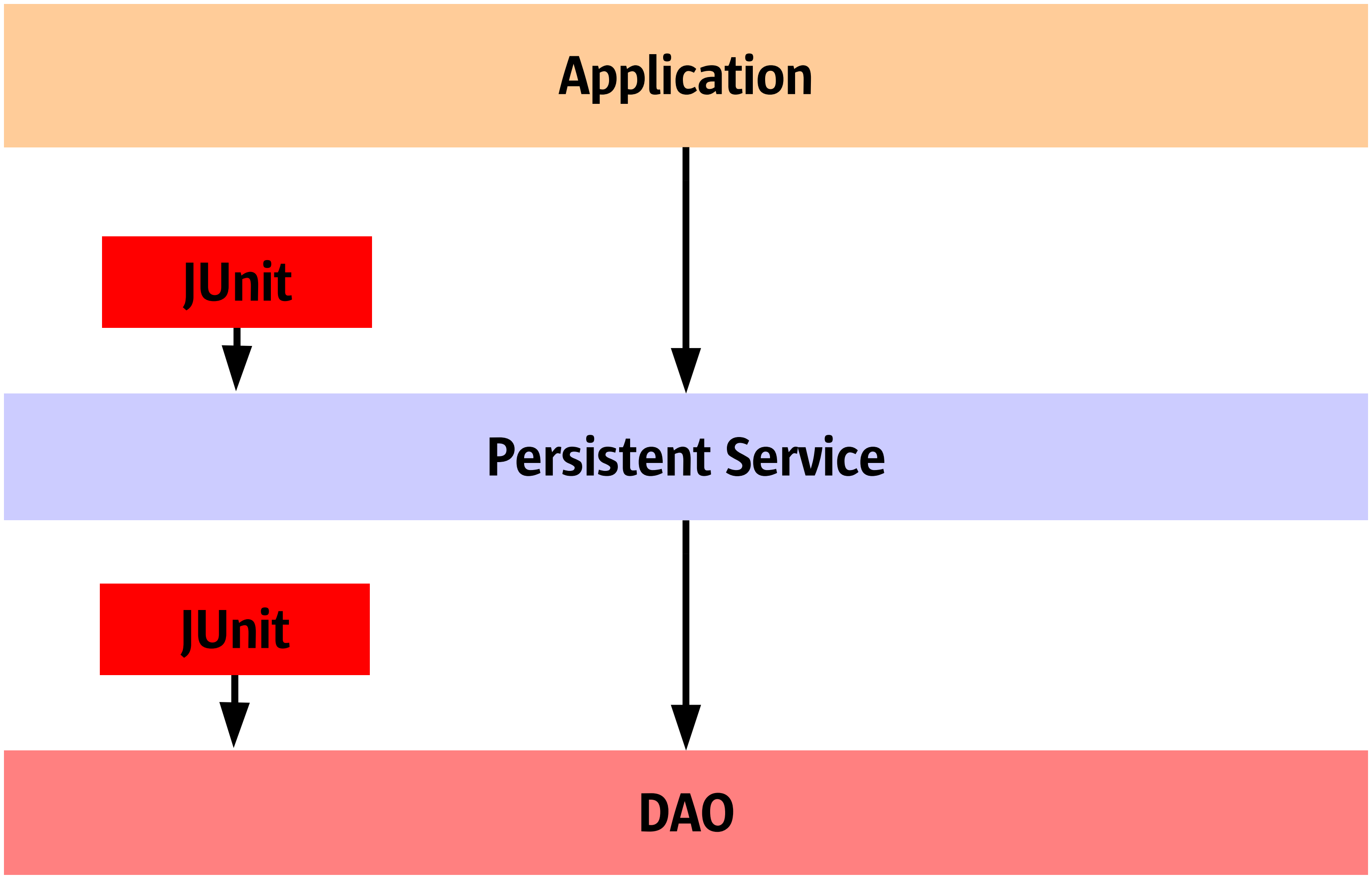


```
survey.dao
    IDataAccess.java
    JdoDataAccessImpl.j
survey.domain
    Administrator.java
    Answer.java
    Entity.java
    Question.java
    QuestionFormat.java
    Respondent.java
    Response.java
    Survey.java
    User.java
survey.domain.keys
    AdministratorKey.jav
    QuestionFormatKey.j
    RespondentKey.java
    UserKey.java
survey.service
    IPersistenceService.java
    PersistenceServiceImpl.java
survey.test
    Demo1.java
    SurveyTest.java
```

```
IPersistenceService.java
    IPersistenceService
        getAdmin(String)
        getAllAdmins()
        getAnswer(Answer)
        getQuestion(Question)
        getQuestionFormat(Stri
        getRespondent(Respor
        getSurvey(Survey)
        getSurveysByAdmin(Str
        storeAdmin(Administrat
        storeAnswer(Question,
        storeQuestion(Survey,
        storeQuestionFormat(Q
        storeRespondent(Resp
        storeResponse(Questio
        storeSurvey(Survey, St
```

```
IDataAccess.java
    IDataAccess
        loadAdmin(String, boolean)
        loadAllAdmins()
        loadAllSurveysByAdmin(String, boolean)
        loadAllSurveysByRespondent(String, boolean)
        loadAnswer(Answer, boolean)
        loadQuestion(Question, boolean)
        loadQuestionFormat(String, boolean)
        loadRespondent(Respondent, boolean)
        loadSurvey(Survey, boolean)
        storeAdmin(Administrator)
        storeAnswer(Question, Answer)
        storeQuestion(Survey, Question, QuestionFormat)
        storeQuestionFormat(QuestionFormat)
        storeRespondent(Respondent, Administrator, Survey)
        storeResponse(Question, Respondent, Collection, Response)
        storeSurvey(Survey, String)
```

Architected Demo



Suggested Development Approach

- Using use cases, design domain model.
- Determine what datastore actions have to be performed to fulfill use cases.
- Build persistent/DAO layers ensuring all use cases are met using JUnit to verify correctness and completeness.
- After persistent layer is complete then lay on a thin UI layer.

Questions

JDO™

Java™ Data Objects

Resources

JPOX – open-source JDO implementation

Spring – open-source IoC container

JDO Central – JDO news, information, and community

JSR 12 – JDO 1.0.1 Specification

JSR 243 - JDO 2.0 JCP

Java Data Objects (JDO)

paul.mckinney@sun.com