SE 550: Lecture 10

Overview

Object persistency

Java persistency support

Transactions

JDBC example

JDO comparison

JDO gruntwork

Object persistency

What is object persistency? Why do we care?

What is orthogonal persistency?

What is a database? A relational database? An object database?

Java persistency support

How can object serialization be used to implement persistency?

What is JDBC? How can it be used to implement persistency?

What is JDO? How can it be used to implement persistency?

What is EJB CMP? How can it be used to implement persistency?

What are the tradeoffs?

See JSR12 and JSR20.

Transactions

An issue with concurrent databases:

Databases should be *ACID*: Atomic, Consistent, Isolated, Durable.

What do these mean?

What is a transaction? How doe transactions help for ACID properties?

Transactions support begin/commit/abort operations - what are these?

How can a database maintain ACIDity? How does a concurrent Java program maintain thread safety? How do these two mechanisms interact?

Take a concurrent database course to find out more!

JDBC example

```
class BookStoreImpl ... {

    final Connection conn;
    final Statement s;

   public void addBook (final Book book) throws BookStoreException, RemoteException {
        Debug.out.println("BookStoreImpl.addBook: Adding " + book);
        try {
            this.s.execute ("INSERT INTO Books values (’" +
                    book.title () + "’, ’" + book.author () + "’, " +
                    book.price () + ")");
            Debug.out.println("BookStoreImpl.addBook: Returning");
        } catch (final SQLException ex) {
            throw new BookStoreException (ex);
        }
    }
    ...
}
```

JDO comparison

```
class BookStoreImpl ... {

    final PersistenceManager pm;
    final Transaction tr;

    public void addBook (final Book book) throws RemoteException {
        this.tr.begin ();
        this.pm.makePersistent (book);
        this.tr.commit ();
    }
    ...
}
```

JDBC example

```
public Book[] findBooks (final String author) throws BookStoreException, RemoteException {
    Debug.out.println("BookStoreImpl.findBooks: Finding " + author);
    try {
        final ResultSet rs =
            s.executeQuery ("SELECT title, price FROM Books " +
                        "WHERE author ='" + author + "'");
        final Vector tmp = new Vector ();
        while (rs.next ()) {
            final String title = rs.getString ("title");
            final int price = rs.getInt ("price");
            final Book book = Book.factory.build (title, author, price);
            tmp.add (book);
        }
        rs.close ();
        final Book[] result = new Book[tmp.size ()];
        tmp.toArray (result);
        return result;
    } catch (final SQLException ex) {
        throw new BookStoreException (ex);
    }
}
```

JDO comparison

```
public Book[] findBooks (final String author) throws RemoteException {
    this.tr.begin ();
    Query query = this.pm.newQuery(BookImpl.class, "this.author == author");
    query.declareImports("import ajeffrey.teaching.jdo.bookstore.BookImpl");
    query.declareParameters("String author");
    Collection rs = (Collection)query.execute(author);
    Book[] result = (Book[])(rs.toArray (new Book[rs.size ()]));
    query.close (rs);
    this.tr.commit ();
    return result;
}
```

JDO gruntwork

JDO takes care of object data bindings for you, but needs help:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jdo SYSTEM "file:/jdo.dtd">
<jdo>
  <package name="ajeffrey.teaching.jdo.bookstore">
    <class name="BookImpl" identity-type="datastore">
      <field name="title" persistence-modifier="persistent"/>
      <field name="author" persistence-modifier="persistent"/>
      <field name="price" persistence-modifier="persistent"/>
    </class>

  </package>
</jdo>
```

This is run through a *class enhancer*: what does this do?

The whole thing is in ajeffrey/teaching/jdo/bookstore/.

Summary

Object persistency is useful for applications which need to survive reboot: Sun support many different persistency models.

Persistency should be ACID: this is often implemented using a transaction model.

Next week: final exam.