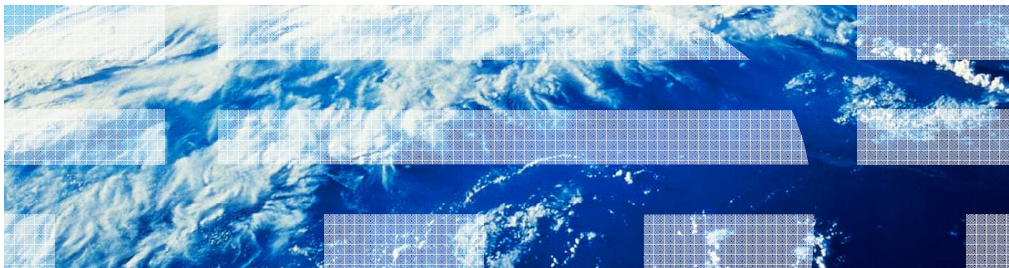




IBM WebSphere Application Server V8

Java Persistence API (JPA) 2.0
JSR 317



WebSphere software

© 2011 IBM Corporation

This presentation describes support for Java Persistence API 2.0 as included in IBM WebSphere Application Server V8.

Table of contents

- Introduction to Java Persistence API (JPA)
- JPA 2.0 Highlights
- OpenBooks Sample Program Description
- References
- Summary

This presentation contains a brief introduction to the Java Persistence API, an overview of what is in the JPA 2.0 feature set in WebSphere Application Server V8, and a description of a sample program available on the web that you can try.

JPA introduction

This section discusses the introduction of JPA.

What is JPA? (1 of 2)

- Standard persistence technology
 - JPA 1.0 introduced in Java EE 5
 - JPA 2.0 is part of Java EE 6 standards
 - Object/Relational Mapping
 - Designed for highly distributed applications
 - Especially web-enabled applications
 - Life cycle manageable by application server to increase quality of service
- Simplifies development
 - Provides framework/tools around providing automatic persistence
 - Objects are mapped to Tables using Metadata
 - Metadata is used to transform data from one representation to another

Here are some of the general JPA concepts.

JPA 1.0 was introduced back in Java EE 5. WebSphere Application Server has had a couple of releases supporting that version of the specification, starting with the EJB3 Feature Pack for WebSphere Application Server Version 6.1 and the support was rolled into Version 7. JPA 2.0 is the next iteration of that specification.

JPA provides a lot of functions, but one of the most important is the object relational mapping capability. If you are familiar with JDBC or CMP Beans or another older technology, then you are likely to see object to relational mapping as a difficult task. JPA provides the capability to do that mapping much more efficiently and in such a way that it is much more applicable to object oriented programmers. JPA greatly simplifies development because it uses standardized metadata to map the objects to the tables. There are standardized mechanisms for accessing these entities whether they are being persisted or queried against. There is now a common query language that can be used called JPQL. Many aspects of JPA are now standardized and therefore portable so moving between vendors is easier than it was in the past. For example, each EJB provider had their own, different, non-standard implementation for CMP beans, so moving from one provider to another was very difficult.

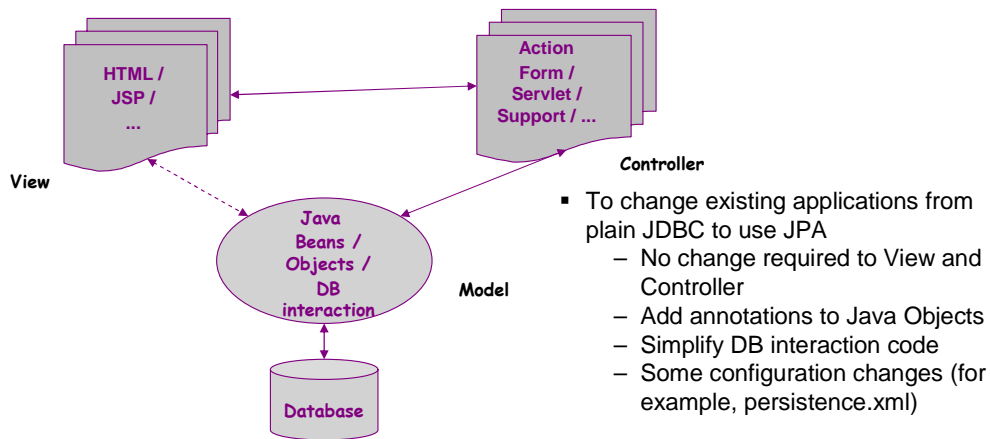
What is JPA? (2 of 2)

- JPA specifications made very significant simplifications:
 - Standardizes O/R mapping metadata (not the case for EJB 2)
 - Java SE 5.0 Annotations can be used instead of XML deployment descriptor
 - No deploy code implementing abstract classes— Entities are POJOs
 - Application Server is not required
 - The Java Persistence API is available outside of Containers
 - Unit testing greatly simplified
 - Removal of checked exceptions (for instance, remote exceptions)
 - No bean (or business) interface required

The first JPA 1.0 specification, which was shipped with Java EE 5, used the annotation support that was part of Java SE 5. Your metadata can be specified either using annotations, which are very intuitive and natural for programmers, or it can be done in XML so you have more control over how you want the mapping without having to change the code.

One thing to highlight on this slide is that an application server is not required in order to reap the benefits of JPA. JPA is fully functional in a JSE mode. That aids development because developers can test their mapping to the database without needing an application server to do it. Once your database interaction is tested and modified to your liking you can bring it into your enterprise application using the model-view-controller model.

Web application example – MVC framework



On this slide, you see the Model-View-Controller model. One of the key things is that you want to separate your database interactions (Model) from the rest of your application. That is where JPA comes into play because it can be done completely separate from the rest of your application. Presentation code goes in the View and your business logic in the Controller.

Annotate entity object class

```
import javax.persistence.*;

@Entity
public class Client {

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private int ID;

    private String name;
    private String industry;
    private String headquarterLocation;
    private String cepClient;
    private String betaClient;
    private String productsInUse;
    private String wasVersions;

    public Client() {
        ...
    }

    //all necessary getter and setter methods
    ...
}
```

Import the javax.persistence packages

Declare the class as an entity

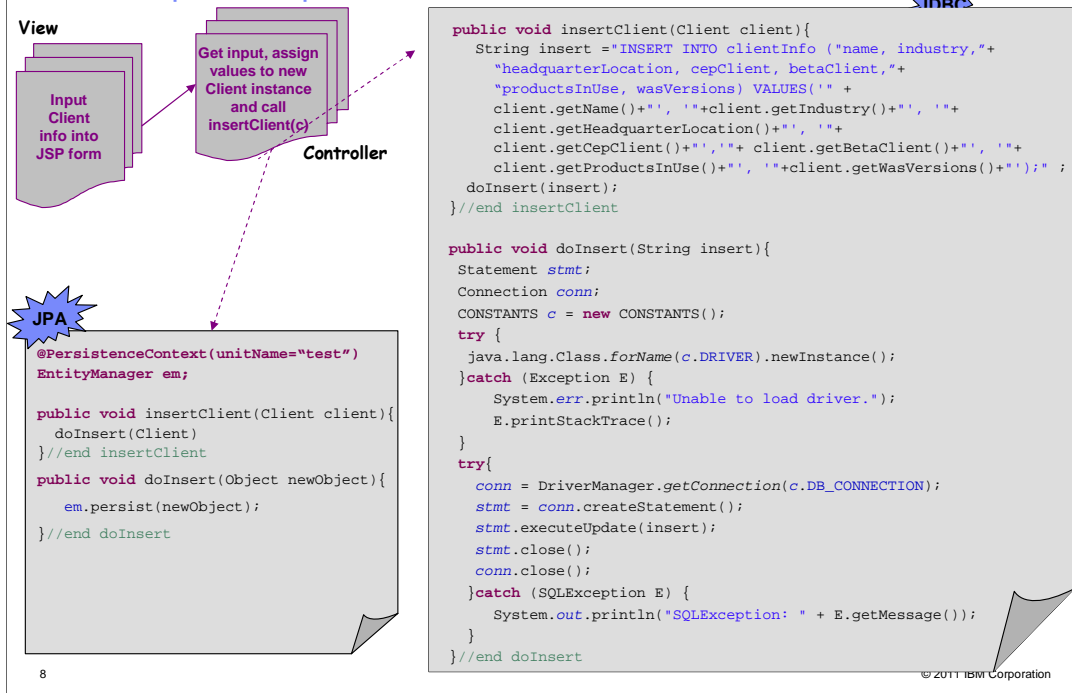
Identify a property as a primary key

7

© 2011 IBM Corporation

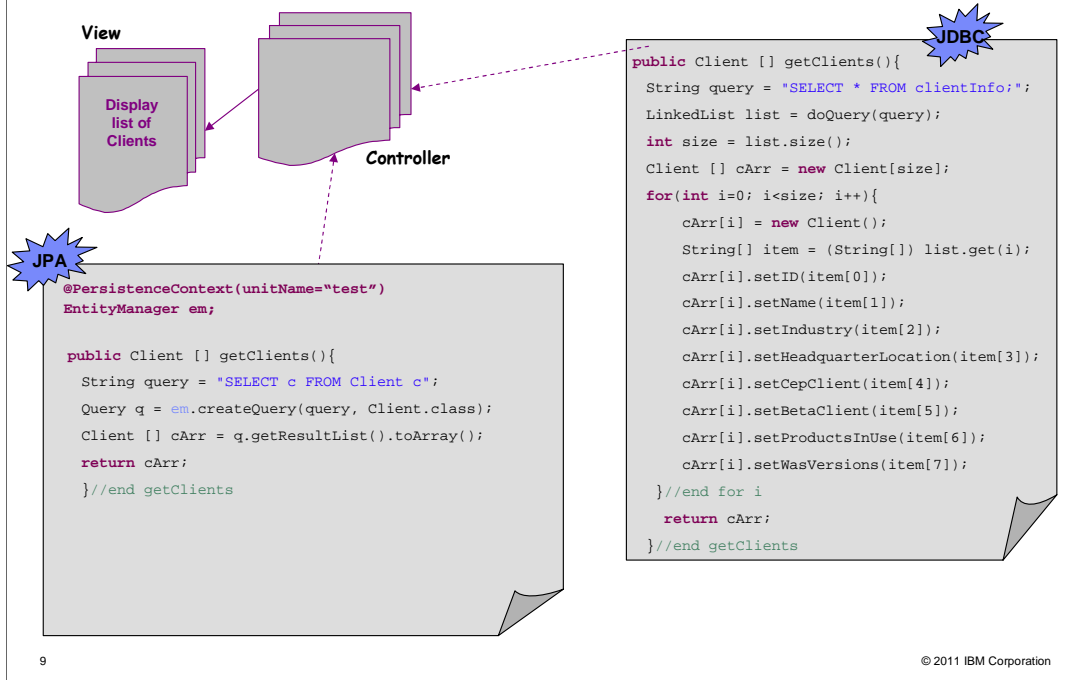
This chart shows a very basic view of a JPA entity. Note here that there are two things required to have a JPA entity: the `@Entity` annotation and the `@Id` annotation. Both are necessary to show that this is an entity within the JPA specification terminology. Entities map to rows in your table. The columns of your table are identified here by the properties or attributes of this entity. The `@Id` annotation defines your table's primary key.

JPA compared to plain JDBC – Insert



This chart (and the next) shows some differences between the JDBC and JPA programming models. As the WebSphere team has been learning over the past few releases, many users are still using JDBC. Now with JPA you do not need to deal with all of the complexities you have been dealing with using JDBC. You do not need to know SQL or how to parse results from SQL; the JPA runtime does all of that for you. In the lower left corner note that the Client object here is a Plain Old Java Object (POJO). In order to write it to the database one updates the attributes of the object then persists the object. All of the data held in the attributes of the POJO are transferred to the data base without you having to write SQL. The attributes in your POJO correlate to columns in your database table.

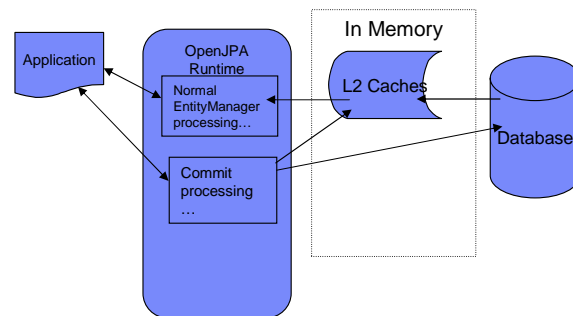
JPA compared to plain JDBC – Retrieve



This page shows a retrieval example. It is no longer necessary to issue an SQL statement to fetch a bunch of rows and then parse the returned data and load it into an object. In the code snippet in the lower left corner you can see the JPQL statement necessary to populate the Client object from the database. The JPQL query language is at the object level and JPA takes care of the database interaction. Again, the attributes of the POJO correlate to columns in your database table. Compare this to the code to the right required to perform the same action using JDBC.

Advantages to using OpenJPA

- Object level programming model
 - More natural for programmers
- Container “free”
 - No requirement on an application server
 - JEE, JSE, and now OSGi
- Common programming model across databases
 - No database-specific SQL or JDBC
- Ability to Cache database records in memory
 - Fewer trips to the database



10

© 2011 IBM Corporation

This discussion has been talking about programming at the object level and taking advantage of JPA's ability to do object mapping. Perhaps there are programmers in your shop who specialize in database interaction and are more comfortable with SQL. But, the more natural way for most programmers is to write code is at the object level. JPA allows database interaction at the object level.

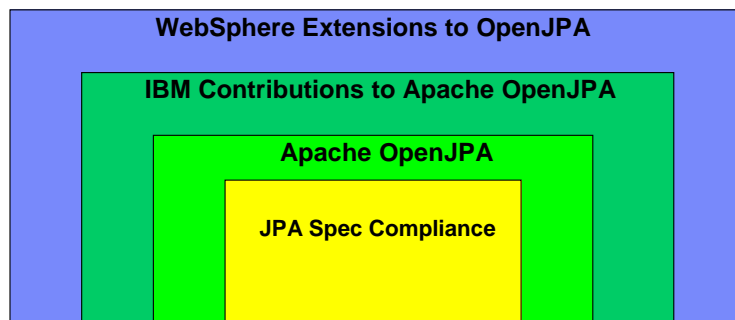
The concept of being container free means that JPA can be used even without an application server of any type, though JEE, JSE, and OSGi are supported.

There is no need to use database specific SQL or JDBC, making code more portable.

JPA also provides some memory caching of records so you do not have to go out to the database for every retrieval. This holding of data in L2 cache improves performance.

WebSphere's JPA architecture

- WebSphere's JPA solution
 - Spec compliant
 - Feature rich
 - Extensible
 - Competitive with hibernate and EclipseLink
 - Production ready with full IBM support



11

© 2011 IBM Corporation

This chart shows the relationship between Apache OpenJPA and WebSphere's JPA solution. Starting with Apache OpenJPA at the bottom of the chart and, of course, it is Spec Compliant. IBM has added enhancements and performance improvements, debugging ability and logging that have been donated to the Apache project. IBM has a vibrant community in Apache. Most of the WebSphere JPA development team members are committers on the Apache OpenJPA project, several are on the project management committee, so IBM is very involved in the Apache OpenJPA project. The team continually contributes bug fixes, features, performance improvements, test cases, and documentation updates to the project.

On top of what IBM contributes to Apache, the WebSphere offering has the value add of integration with key IBM and WebSphere stack products like DB2 and WebSphere eXtreme Scale (WXS). Also added are extended logging and translated messages in all of the languages supported by WebSphere Application Server. The WebSphere Application Server JPA 2.0 feature is production ready with full IBM support.

JPA 2.0 highlights

This section discusses a brief overview of the JPA 2.0 feature set.

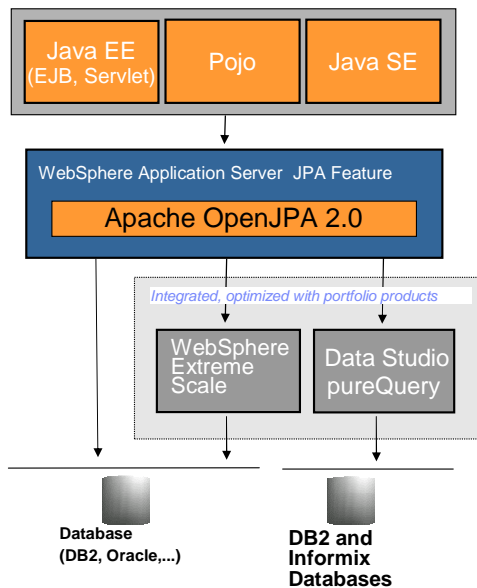
WebSphere Application Server V8 and the Java Persistence API 2.0

- Based on Apache OpenJPA, a leading open source Java persistence framework.
- Provides the Apache OpenJPA 2.1 implementation with IBM enhancements to benefit integration with WebSphere Application Server.
- The Apache OpenJPA 2.1 implementation includes improvements and benefits over previous releases and even beyond the JPA 2.0 specification.

As stated on this chart and as mentioned previously, the JPA2.0 feature in WebSphere Application Server V8 is based on Apache OpenJPA.

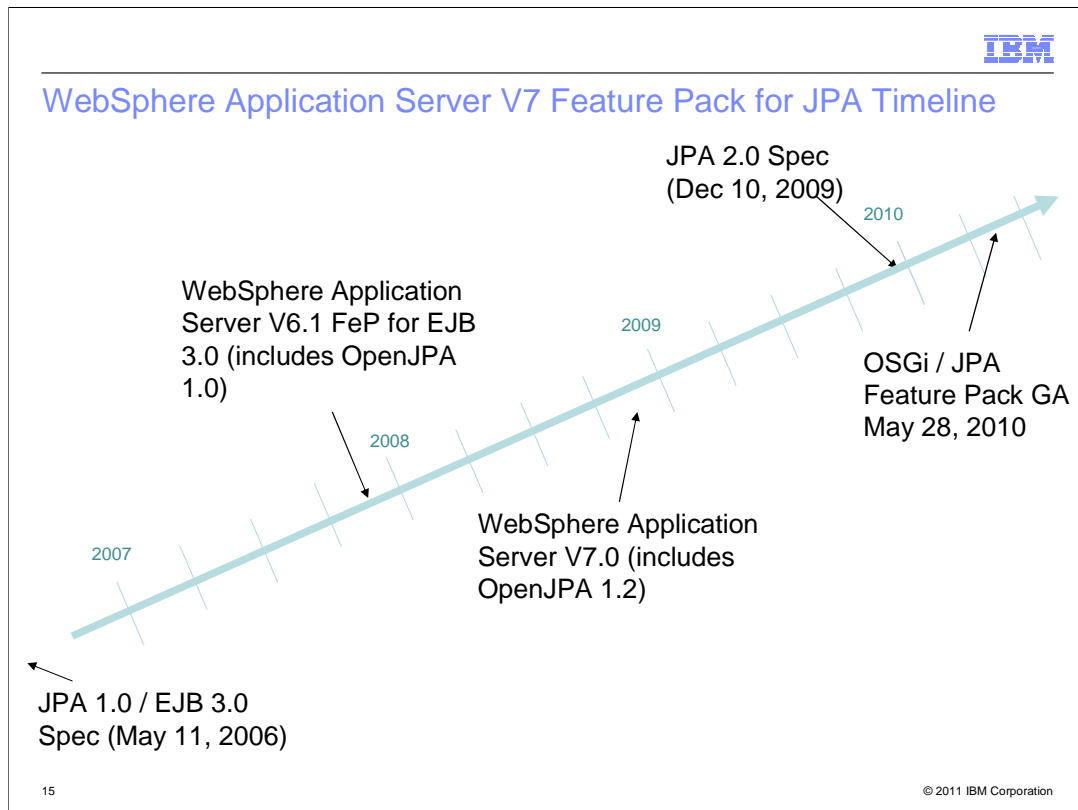
The majority of the IBM JPA team's development work goes into code that is contributed to the OpenJPA project. Extensions are then added to make the IBM feature complete. This means all OpenJPA functionality, extensions, and configurations are unaffected by the WebSphere Application Server extensions. Users running OpenJPA applications do not need to make any changes to use their applications in WebSphere Application Server but users who want to use other IBM products and WebSphere Application Server Stack products will benefit from the enhancements made for the JPA solution as shown on the next slide.

WebSphere Application Server JPA 2.0 Feature Architecture



- Apache OpenJPA 2 provides:
 - JPA 2.0 standards currency:
 - JPA 2.0 (JSR 317) includes important improvements on the data persistence and relational database management capabilities. Early delivery of part of the JEE 6 standard.
 - WebSphere Application Server Integration:
 - Enhanced management, debugging, monitoring, and security for WebSphere Application Server.
 - WebSphere eXtreme Scale integration enhancements:
 - Features that enable JPA to use WebSphere eXtreme Scale as a level2 cache in order to improve data access performance
- *WebSphere JPA adds:*
 - Additional WebSphere Application Server Integration:
 - Installation, extended debugging
 - Data Studio pureQuery runtime optimization:
 - Performance and monitoring improvements for database access

This chart shows pictorially how the JPA feature integrates with WebSphere eXtreme Scale and the IBM database products. In addition to working in SE mode and in a Java EE container, JPA can also be used in the OSGi container. The OSGi runtime is another feature in WebSphere Application Server V8.



This chart, although specific to the Feature Pack, demonstrates IBM's commitment to supporting and promoting the JPA framework.

Notice the increased velocity of implementation of the JPA specifications by WebSphere Application Server. As one can see the difference in time between the first JPA 1.0 specification and the availability of the function in WebSphere Application Server was much longer than for the JPA 2.0 deliverable. Considerable effort is being spent to keep customers current with specifications.

The JPA 2.0 functionality from the Feature Pack has been rolled into WebSphere Application Server V8.

JPA 2.0 highlights

- O/R mapping and domain modeling
 - Embeddables
 - Access types
 - Enhanced map collections
 - Derived identities
 - JPQL update
- Enhanced locking
- Runtime API updates
 - EntityManagerFactory API, EntityManager API, Query API, Typed Result API
- Metamodel and criteria API
- Bean validation
 - Provides entity validation based on JSR-303 semantics

There are many changes to the JPA spec in 2.0. There were considerable improvements in the object relational mapping and domain modeling area. There are also improvements to the runtime APIs providing access to caching capabilities and having additional lock managers available.

JPA 1.0 was only capable of optimistic lock management but JPA 2.0 has the capability to do pessimistic lock management.

One of the big items with JPA 2.0 is the meta-model and criteria APIs that provide a programmatic interface for generating queries so you do not have to rely upon strings of JPQL to do queries. You can do it programmatically and change the various predicates and where clauses that you want to do with your queries.

Bean validation is a separate JSR (JSR-303) but it has tight integration with the JPA 2.0 specification. That capability is provided as well so you can validate your entities before or after they get persisted to the database.

Object-relational persistence is a key developer requirement for many application developer scenarios. JPA is the Java EE standard for object-relational persistence and was first introduced as part of Java EE 5. As part of the Java EE 6 standards, JPA 2.0 (JSR-317) updates object-relational capabilities with important developer APIs and enhancements.

Additional detail on these new features can be found in the References section of this presentation.

OpenBooks sample program description

Now for a brief description of the OpenBooks Sample.

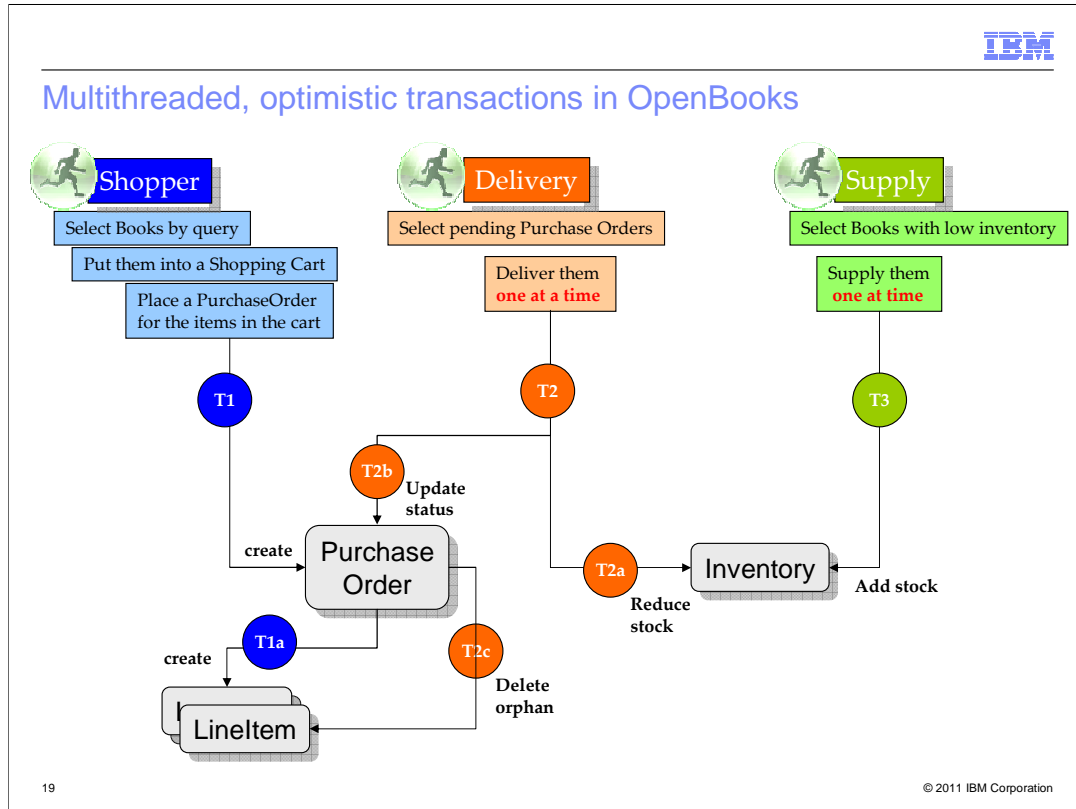


Overview

- OpenBooks is a sample application based on JPA 2.0
 - **Good practices**
 - Persistent Domain Modeling
 - JPA based Application Development
 - Testability of Transactional Applications
 - Design Patterns to operate within and without container at ease
 - **Demonstration of new features of JPA 2.0**
- <http://openjpa.apache.org/openbooks-featuring-jpa-20.html>

OpenBooks is a simple shopping cart application. It was created by one of the development teams who is a member of the JPA Expert Group, an OpenJPA Committer, and an OpenJPA Project Management Committee member. It was created to show how to best use the new functionality and to demonstrate the new features of JPA 2.0. At the time of this writing, the sample is available on the web at the link indicated on the chart. Should it move in the future, find it with the search string: +”OpenJPA” +”OpenBooks” +”Sample”.

Multithreaded, optimistic transactions in OpenBooks

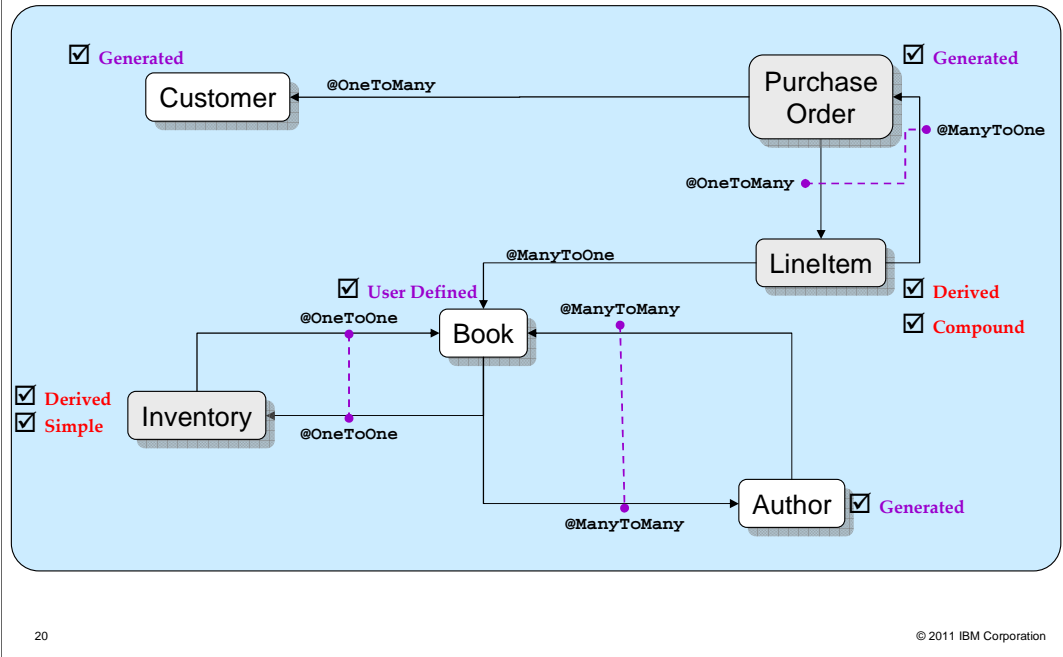


In OpenBooks, the shopper selects books, puts them in their shopping cart, and makes a purchase. Then the books are delivered and if that order depletes the supply, the supply can be replenished.

This chart shows the work flows that are performed by the application. The workflows are select and buy a book, deliver the purchased book, and supply or replenish the inventory.

A Shopper browses books, puts them into a shopping cart and eventually purchases them.

OpenBooks domain model



Here is an overview of the domain model used in the OpenBooks sample program.

There are six entities. Between the entities all of the cases of entity relationships are represented; OneToOne, OneToMany, and ManyToMany.

Also shown in the graphic is the kind of primary key each object has and how it is defined

OpenBooks user interface

The screenshot shows the OpenBooks user interface with the following components:

- Navigation Panel:**
 - Application Workflows
 - Bookstore/Book Order
 - Deliver Pending Orders
 - Search Books
 - Views
 - View Configuration
 - View Domain Model
 - View Queries
 - View Services
 - View Source Code
- Step 1: Search for Books**
 - Title:
 - Author:
 - Price: from to
 -
 - Criteria Query as SQL:


```
SELECT a
FROM book a
WHERE (a.title LIKE 'book%' AND b.price BETWEEN 100.0 AND 200.0)
```
- Step 2: Select Books to view details**
 - 3 Book selected
 - 0 Book

id	ISBN	version	price	title
1	A7670-22	1	120.611	Book-3
2	05695-41	1	116.728	Book-4
3	57882-31	1	120.584	Book-5
- Step 3: Add Book to Shopping Cart**
 - Book-5
 - By Author: 18
 - Price: 120.58400000000000
 - ISBN: 57882-31
 - Quantity:
 -
- Step 4: Purchase Books in the Shopping Cart**
 -
- SQL Log**

```
SELECT to_id, to_version, to_name FROM Customer to WHERE (to.name = ?)
SELECT to.ISBN, to.version, to.price FROM Book to WHERE (to.ISBN LIKE ? ESCAPE '\') AND to.price <= ?)
SELECT to.ISBN, to_id, to_version, to_name FROM Book to INNER JOIN Book_Author t1 ON to.ISBN = t1.BOOKS_ISBN INNER JOIN Author t2 ON t1.AUTHORS_ID = t2.ID WHERE (to.ISBN LIKE ? ESCAPE '\') AND to.price <= ?)
ORDER BY to.ISBN ASC
SELECT to.ISBN, to_version, to_price, to.Name FROM Book to WHERE (to.ISBN LIKE ? ESCAPE '\') AND to.price <= ? AND to.price <= ?)
SELECT to.ISBN, to_id, to_version, to_name FROM Book to INNER JOIN Book_Author t1 ON to.ISBN = t1.BOOKS_ISBN INNER JOIN Author t2 ON t1.AUTHORS_ID = t2.ID WHERE (to.ISBN LIKE ? ESCAPE '\') AND to.price <= ?)
ORDER BY to.ISBN ASC
```

Here is the user interface of the sample. Note that in the panel at the bottom of the window one can observe SQL statements being generated by JPA.

The program uses best programming practices and demonstrates JPA 2.0 capabilities.


You are invited to experiment with this sample. Download it, build it, run it, play with it, view the code, and modify the code. The sample provides an easy way to learn several of the new JPA 2.0 features.

References and summary

This section provides a summary of this presentation.

References

- WebSphere Application Server V7 FeP for OSGi Applications and JPA 2.0 site
 - <http://www-01.ibm.com/software/webservers/appserv/was/featurepacks/osgi/>
- Apache OpenJPA code
 - <http://openjpa.apache.org/>
- WebSphere Application Server V7
 - <http://www-01.ibm.com/software/webservers/appserv/was/>
- WebSphere Application Server V6.1 Feature Pack for EJB 3.0
 - <http://www-1.ibm.com/support/docview.wss?rs=177&uid=swg21287579>
- Blogs and Wikis
 - <http://webspherepersistence.blogspot.com/>

<https://www.ibm.com/developerworks/mydeveloperworks/wikis/home?lang=en#/wiki/WebSphere%20Application%20Server%20Test%20Team>
- White papers
 - http://www.ibm.com/developerworks/websphere/techjournal/1008_col_sutter/1008_col_sutter.html
 - http://www.ibm.com/developerworks/websphere/techjournal/0909_col_sutter/0909_col_sutter.html
 - <http://www.ibm.com/developerworks/java/library/i-typesafejpa/>
 - http://www.ibm.com/developerworks/websphere/techjournal/0712_barcia/0712_barcia.html
 - http://www.ibm.com/developerworks/websphere/techjournal/0612_barcia/0612_barcia.html
- Books
 - Pro JPA 2: Mastering the Java Persistence API by Mike Keith and Merrick Schincariol
 - 
 - http://www.amazon.com/Pro-JPA-Mastering-Persistence-Technology/dp/1430219564/ref=sr_1_1?ie=UTF8&s=books&qid=1272049168&sr=8-1
 - Persistence in the Enterprise, A Guide to Persistence Technologies by Roland Barcia, et al
 - http://www.amazon.com/Persistence-Enterprise-Guide-Technologies-developerWorks/dp/0131587560/ref=sr_1_1?ie=UTF8&s=books&qid=1239033026&sr=1-1

Here are some useful references, particularly if you are just getting started with JPA.

Deliverables

- Apache OpenJPA 2.0 and 2.1 are publicly available!
 - <http://openjpa.apache.org/downloads.html>
- WebSphere Application Server V7 Feature Pack for OSGi Applications and JPA 2.0 is available now!
 - <http://www-01.ibm.com/software/webservers/appserv/was/featurepacks/osgi/>
- WebSphere Application Server V8 is available now!
<http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=AN&subtype=CA&htmlfid=897/ENUS211-139&appname=USN>
- Download and experiment with any version

Several delivery vehicles exist for your testing and experimentation. The OpenJPA 2.0 deliverable from Apache, the WebSphere Application Server Feature Pack for OSGi Applications and Java Persistence API 2.0, and WebSphere Application Server V8 are all available. You are welcome to use whichever packaging and delivery vehicle works for your environment.

Summary

- JPA 2.0 is a natural evolution of the popular JPA 1.0 specification
- OpenJPA 2.0.0-beta was the first publicly-available TCK compliant release (after the RI). With a formal OpenJPA 2.0 release coming out shortly after that.
- JPA 2.0 implementation was first provided in the OSGi/JPA 2.0 Feature Pack for WebSphere Application Server V7
- Now, JPA 2.0 is provided as part of the overall Java EE 6 solution in WebSphere Application Server V8

In summary, JPA 1.0 is already used extensively in the industry and JPA 2.0 is gaining traction.

The JPA 2.0 implementation has been refined through several release cycles. First, the Apache OpenJPA 2.0 release, then the OSGi/JPA 2.0 Feature Pack, and now with the WebSphere Application Server V8 release. The WebSphere JPA solution is solid and is ready for your production-level applications.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send email feedback:

[mailto:iea@us.ibm.com?subject=Feedback about WASV8 JPA20.ppt](mailto:iea@us.ibm.com?subject=Feedback%20about%20WASV8%20JPA20.ppt)

This module is also available in PDF format at: [../WASV8_JPA20.pdf](#)

You can help improve the quality of IBM Education Assistant content by providing feedback.



Trademarks, disclaimer, and copyright information

IBM, the IBM logo, ibm.com, and WebSphere are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the Web at "[Copyright and trademark information](http://www.ibm.com/legal/copytrade.shtml)" at <http://www.ibm.com/legal/copytrade.shtml>

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.

© Copyright International Business Machines Corporation 2010. All rights reserved.