Introduction to EJB

- What is an EJB?
- EJB History
- Hello, World

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

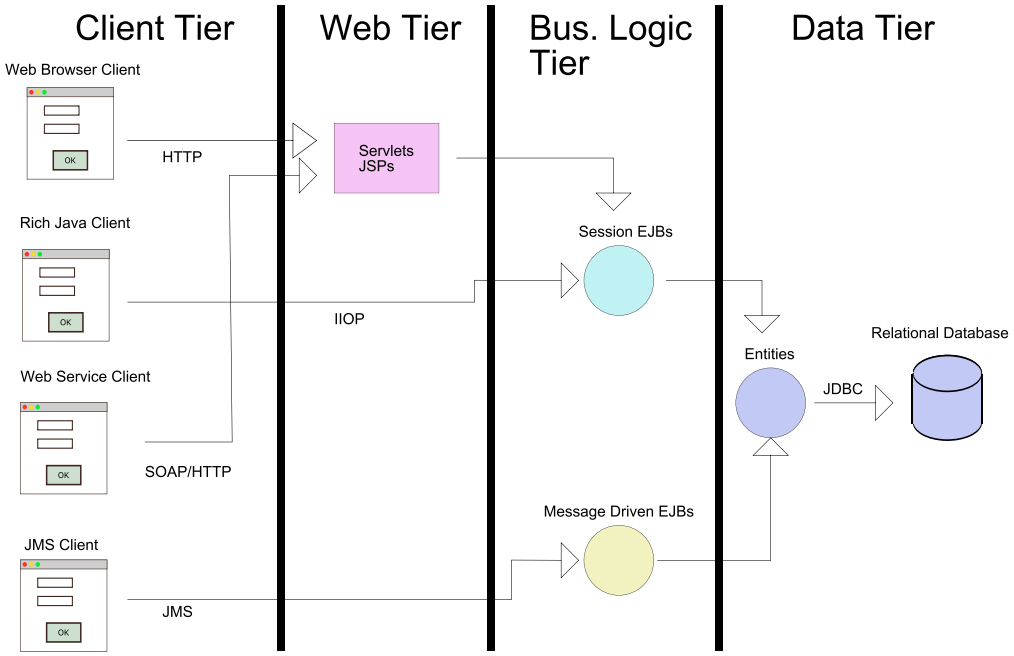
Introduction to EJB

- From the Sun Web site:
  - **Enterprise JavaBeans (EJB)** technology is the server-side component architecture for Java Platform, Enterprise Edition (Java EE). EJB technology enables rapid and simplified development of distributed, transactional, secure and portable applications based on Java technology.
  - **The Java Persistence API** is the standard API for the management of persistence and object/relational mapping. It provides an object/relational mapping facility for application developers using a Java domain model to manage a relational database. The Java Persistence API is part of the Java EE platform. It can also be used in Java SE environments.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

JEE Architectures

- EJBs provide the **business logic** tier in a the standard JEE architecture
- **Entities** provide the object-relational mapping layer



Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

EJBs vs JavaBeans

- Simple JavaBeans provide **properties** (via get/set methods), **methods** and **events**
- Simple JavaBeans do not need any runtime support

```
1    public class Student
2    {
3      private int id;
4      private double gpa;
5
6      public int getId() {return id;}
7      public void setId(int i) {id=i;};
8      public double getGpa() {return gpa;}
9      public void setGpa(double d) {gpa=d;}
10   }
```


Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

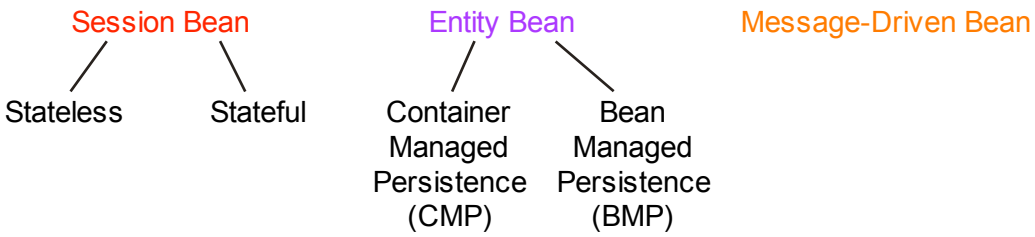
EJB History

- EJB 1.0 was released in mid 1998 and defined session EJBs (stateless and stateful) and entity EJBs (bean-managed and container-managed persistence)
- EJB 1.1 was released in late 1999 and deprecated support for JavaBean-based deployment descriptors and required container-managed persistence
- EJB 2.0 was released in mid 2001 and added support for container-managed relationships and the EJB Query Language
- EJB 3.0 was released in 2006 and is geared toward a simplified programming model using JDK5 annotations
- EJB 3.1 was released in 2009 and added a no-interface view for session EJBs and other enhancements

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

A Look Back at EJB2

- The EJB2 specification defined a distributed, persistent component model for Java, but was widely criticized as being too complex to code for, maintain and administer
- EJB2 component types:



Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Introduction to EJB3

- EJB3 is a significant rewrite of the earlier Enterprise JavaBeans specifications
- The primary goal is easier programming and configuration:
    - Use of POJOs
    - Use of JDK5 **annotations**
    - Less reliance on XML deployment descriptors

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

What is a POJO?

- **Plain Old Java Objects** are simply Java classes with no required inheritance hierarchy
- POJOs are easy to write, test and maintain

```
package hello;

import javax.persistence.*;

@Entity
public class Hello
{
   @Id
   private int helloID;
   private String msg;

   public void setHelloID(int helloID)
   {
      this.helloID = helloID;
   }
}
```

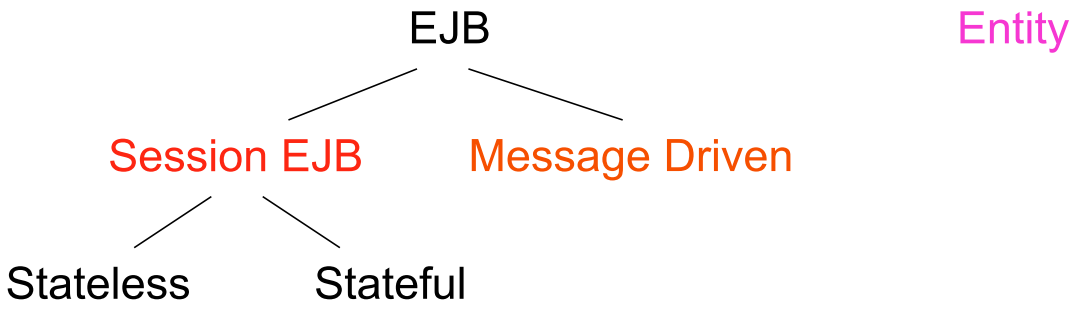


Here we show a POJO that will perform as an entity. Note the use of JDK5 annotations to provide metadata.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

EJB Types

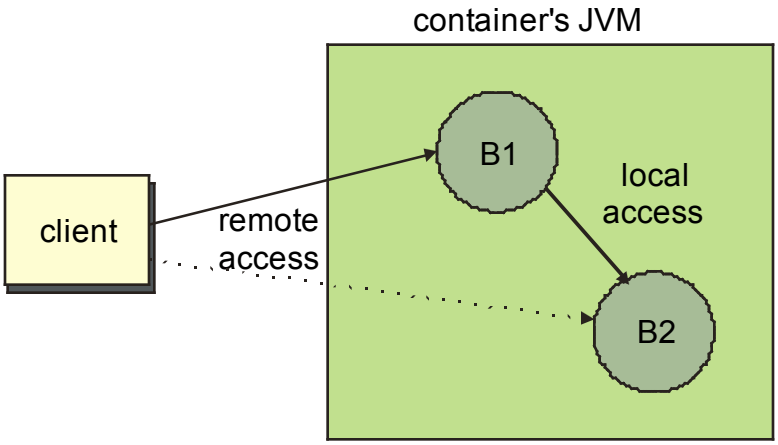
- The EJB3 specification provides for two types of **session** EJBs plus **message-driven** EJBs
- To support object/relational mapping, JEE applications uses **entities**, which are not actually EJBs



Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Local and Remote Interfaces

- Clients can invoke session beans either by a remote or a local interface
- The local interface is faster (and the default), but requires that the client be in the same JVM as the bean



Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

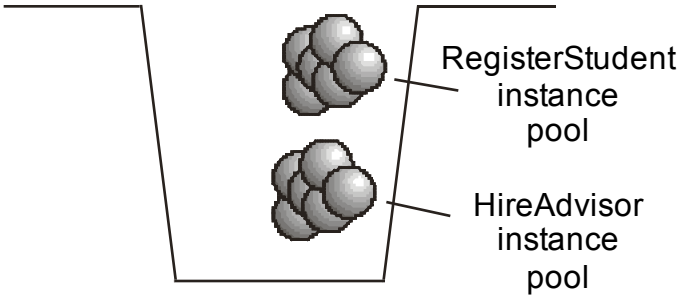
What are Session Beans?

- Session beans represent **business logic**, but not business data
- Session beans run within an **EJB container** that provides services such as transactions and security
- To create a session bean, you must write both an implementation class and a separate business interface

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Stateless Session Beans

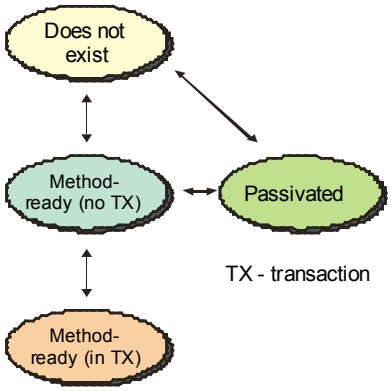
- Stateless session beans represent **single-method conversations** between the client and the bean
- An SSB should **NOT** maintain any client-provided state
- The EJB container can pre-create an **instance pool**for efficiency



Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Stateful Session Beans

- Stateful session bean conversations can span multiple methods
- A SFSB can maintain client-provided state across method invocations
- Containers cannot pool stateful session beans in the same way as stateless beans, but the container can **passivate** stateful beans to manage memory



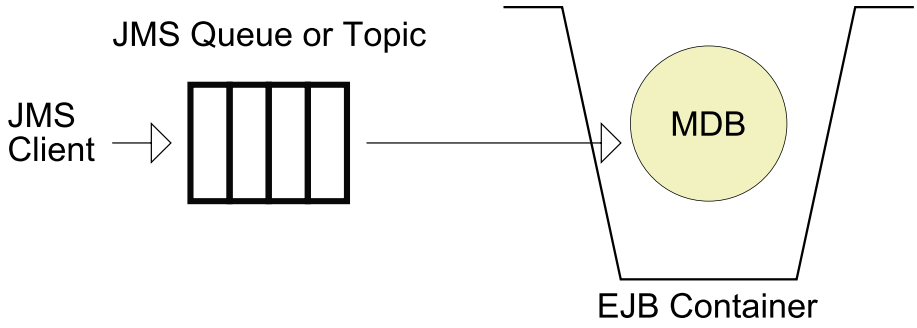
Stateful session beans are less efficient than stateless beans since each instance is dedicated to a client.

Passivation is similar to how operating systems manage memory by paging out least-recently-used storage and then paging it back in on demand.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

What are Message-Driven Beans?

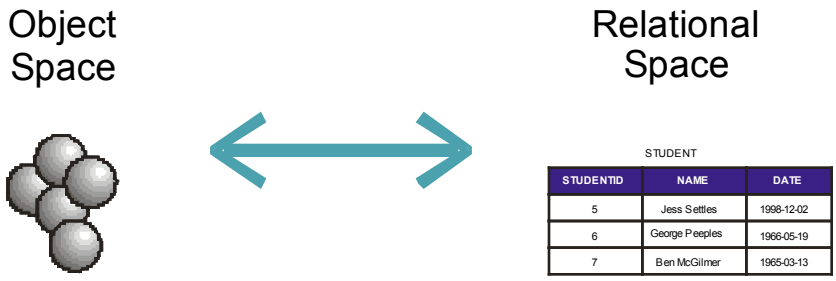
- Like stateless session beans, message-driven beans represent high-level **business logic**
- Unlike stateless beans, clients invoke MDBs asynchronously via JMS
- MDBs run within the EJB container



Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

What are Entities?

- Entities represent business data and the primitive operations that work on the data
- In EJB3, entities use the **Java Persistence API** (JPA)
- Entities do not require an EJB container



Entities are POJOs that use annotations to provide metadata as to how the entity's data should map to relational storage. In truth, entities are not really EJBs at all, since they can work in standalone Java applications without a container.

In EJB2, there were two types of entity EJBs: container-managed and bean-managed persistence. These categories go away in EJB3, where there are just entities.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Development Steps for a Simple SSB

1. Write the **business interface** for the stateless session bean
2. Write the **implementation** POJO that implements the interface using annotations to provide configuration metadata
3. Write a **client** that exercises the stateless session bean

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

The Business Interface

```
1    package shipping;
2
3    import javax.ejb.Remote;
4
5    @Remote
6    public interface Shipping
7    {
8       public double calcDomestic(String zip);
9    }
```


Note the use of the @Remote annotation to indicate that this SSB can be invoked remotely.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

The Implementation

```
1    package ship;
2
3    import javax.ejb.Stateless;
4
5    @Stateless
6    public class ShippingBean implements Shipping
7    {
8        public double calcDomestic(String zip)
9        {
10          return 3.55;
11       }
12   }
```


Note that this is a POJO: it only needs to implement the business interface shown on the last page.

Also note how it uses the @Stateless annotation to configure a stateless EJB rather than a stateful EJB.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

A Simple Client

- Here is a servlet acting as a client to the stateless session bean:

```
1    package ship;
2    . . .
3    public class MyServlet extends HttpServlet
4    {
5       @EJB
6       private Shipping shipping;
7
8       public void doGet(HttpServletRequest request,
9         HttpServletResponse response)
10      {
11         . . .
12         double amt = shipping.calcDomestic(
13           request.getParameter("zipCode");
14         . . .
15      }
16   }
```


Here we show a servlet that uses dependency injection to gain a reference to the EJB. Note that the reference type is the business interface, not the implementation class.

The servlet can then invoke the business-logic function provided by the session bean.

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Chapter Summary

In this chapter, you learned:

- The fundamentals of EJB3
- How to write a simple session bean

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2016. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum