Building EJB™ 3.0 Applications: A Simple Matter of Point and Squish

Mike Keith
Merrick Schincariol

Oracle Corp.
http://otn.oracle.com/ejb3

TS-3616

Goal

Learn how to build and package Enterprise JavaBeans™ (EJB™) 3.0 based applications

Agenda

Introduction to EJB 3.0 Specification

Building Components

Building Entities

Packaging Up the Components

Adding the Entities

Other Packaging Options

Summary

Introduction to EJB 3.0 Specification

Components and Entities

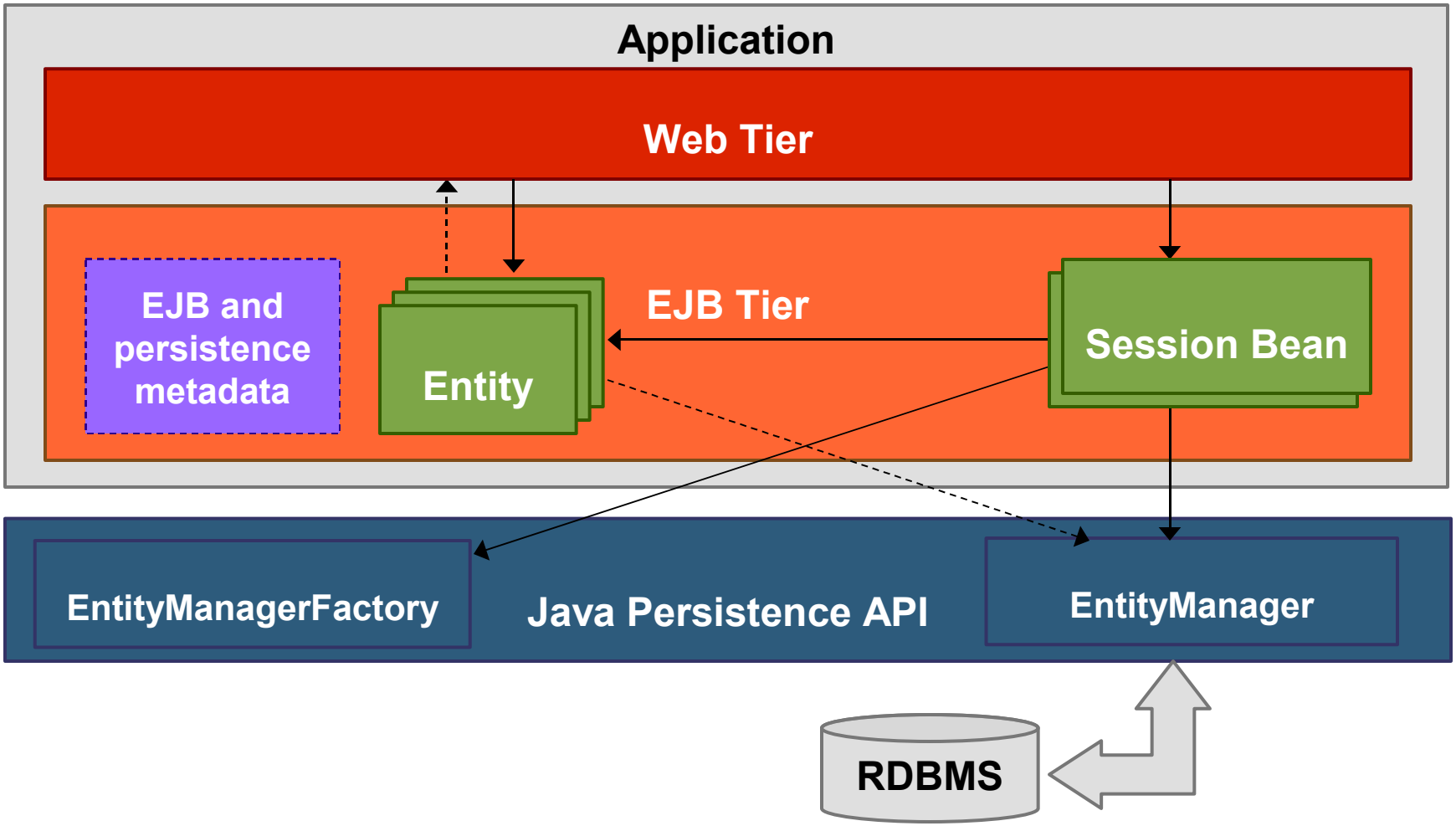
- Components
  - Traditional EJB based components
    - Session beans, message-driven beans, CMP/BMP entity beans
  - Declarative container services
    - Transactions, security, concurrency
    - Managed by Container
      - Deployment, distribution, execution, access
      - Bound to the managing container
  - Primary application vehicle
    - Houses business domain logic

Introduction to EJB 3.0 Specification

Components and Entities

- Entities
  - EJB 3.0 Java Persistence API
    - Persistent POJOs
  - No additional container services
    - “Inherits” container services of components
  - Managed by a local EntityManager
    - Accessed through EntityManager API
    - May be detached and merged back in
  - Fine-grained persistent state
    - Used by components containing the domain logic

EJB 3.0 Application Architecture



Agenda

Introduction to EJB 3.0 Specification

Building Components

Building Entities

Packaging Up the Components

Adding the Entities

Other Packaging Options

Summary

Building Components

What's in a Session Bean?

Session Bean = Interface +
Implementation class +
Metadata

- One or more business interfaces
- Concrete POJO implementation class
- Implementation class implements interface(s)
- Minimal metadata
  - Either annotations or XML

Building Components

Airline Reservation Session Bean

Business Interface

```
public interface AirlineReservation {
  public void addFlight(int id, String destination,
                        Time departure, Time arrival);
  public Collection<Flight> getFlights(String destination
                                       Date departure);
  public Collection<Passenger> getPassengers(String name);
  public int addPassenger(String name, Date dob);
  public boolean book(int passId, int flightId);
  public boolean cancel(int passId, int flightId);
}
```

Building Components

Airline Reservation Session Bean

Implementation

```
public interface AirlineReservationBean
      implements AirlineReservation {

  EntityManager em;

  public void addFlight(int id, String destination,
                        Time departure, Time arrival) {
    em.persist(
        new Flight(id, destination, departure, arrival));
  }
```

Building Components

Airline Reservation Session Bean

Implementation (Cont.)

```
public Collection<Flight> getFlights(String destination,
                                     Date departure) {
    return (Collection<Flight>)
      em.createNamedQuery("Flight.findByDest")
        .setParameter("dest", destination)
        .setParameter("depDate", departure)
        .getResultList();
  }
  public int addPassenger(String name, java.util.Date dob) {
    // ...
  }
  public Collection<Passenger> getPassengers(String name) {
    // ...
  }
}
```

Building Components

Airline Reservation Session Bean

Implementation (Cont.)

```
  public boolean book(int passId, int flightId) {
    Flight flight = em.find(Flight.class, flightId);
    Passenger pgr = em.find(Passenger.class, passId);
    return flight.addPassenger(pgr);
  }

  public boolean cancel(int passId, int flightId) {
    Flight flight = em.find(Flight.class, flightId);
    Passenger pgr = em.find(Passenger.class, passId);
    return flight.removePassenger(pgr);
  }
}
```

Building Components

Airline Reservation Session Bean

Annotation Metadata

```
@Stateless
public interface AirlineReservationBean
      implements AirlineReservation {

  @PersistenceContext
  EntityManager em;

  // . . .


}
```

Agenda

Introduction to EJB 3.0 Specification

Building Components

Building Entities

Packaging Up the Components

Adding the Entities

Other Packaging Options

Summary

Building Entities

What's in an Entity?

Entity = Implementation class +
Metadata

- No business interface required
- Concrete POJO implementation class
- Minimal metadata
  - Either annotations or XML
  - Default values for partial or absent metadata

Building Entities

Passenger Entity

Implementation

```
public class Passenger {

   int id;
   String name;
   Flight flight;

   // Getter & setter methods, etc.

}
```

Building Entities

Passenger Entity

Annotation Metadata

```
@Entity
public class Passenger {

  @Id @GeneratedValue
  int id;
  String name;
  @ManyToOne
  Flight flight;

  // Getter & setter methods, etc.

}
```

Building Entities

Flight Entity

Implementation

```
public class Flight {

  int id;
  String destination;
  java.sql.Time departure;
  java.sql.Time arrival;
  Collection<Passenger> passengers;

  // Constructor, getter, setter methods, etc.
  // ...

  public boolean addPassenger(Passenger pass) { ... }
  public boolean removePassenger(Passenger pass) { ... }
}
```

Building Entities

Flight Entity

Annotation Metadata

```
@Entity
public class Flight {

  @Id
  int id;
  @Column(name="DEST")
  String destination;
  Time departure;
  Time arrival;
  @OneToMany(mappedBy="flight")
  Collection<Passenger> passengers;

  // Methods, etc.
}
```

Agenda

Introduction to EJB 3.0 Specification

Building Components

Building Entities

Packaging Up the Components

Adding the Entities

Other Packaging Options

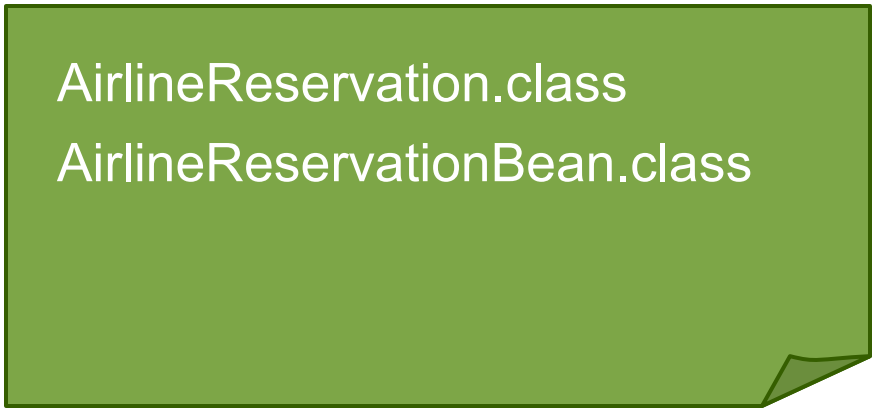
Summary

Packaging Up the Components

Creating an EJB Based JAR

- The EJB based components go in an EJB based JAR file, which contains
  - Business interface(s)
  - Implementation classes

airlineEjb.jar



Packaging Up the Components

Using XML

- May optionally use XML deployment descriptor
  - Create ejb-jar.xml file and add XML metadata
  - Add to META-INF directory in EJB based JAR file
  - Do not need annotations in bean classes

airlineEjb.jar

AirlineReservation.class
AirlineReservationBean.class
META-INF/ejb-jar.xml

ejb-jar.xml

```
<ejb-jar>
  <session>
    <ejb-class>
     AirlineReservationBean
    </ejb-class>
  </session>
</ejb-jar>
```

Agenda

Introduction to EJB 3.0 Specification

Building Components

Building Entities

Packaging Up the Components

Adding the Entities

Other Packaging Options

Summary

Adding the Entities

Entities in the EJB Based JAR

- Need to create persistence unit metadata in XML file called persistence.xml
  - Names the persistence unit and defines its configuration
  - Defines the data source where entities are stored
  - Vendor properties for additional configuration
  - Other optional metadata if needed
    - Persistence provider class if different provider is used
    - Additional ORM mapping files
    - JAR files of additional entities

Adding the Entities

persistence.xml File

```
<persistence>
  <persistence-unit name="AirlineReservation">
    <data-source>jdbc/OracleDB</data-source>
    <properties>
      <property name="toplink.ddl-generation"
                value="create-tables"/>
      <property name="toplink.logging.level"
                value="FINEST"/>
    </properties>
  </persistence-unit>
</persistence>
```

Adding the Entities

Entities in the EJB Based JAR

- Easiest way is to package entities in EJB based JAR with the components
  - Add entity classes
  - Add persistence.xml file to META-INF directory

airlineEjb.jar

AirlineReservation.class
AirlineReservationBean.class
Flight.class
Passenger.class
META-INF/persistence.xml

Adding the Entities

Using XML for Mapping

- Can put some or all the mapping metadata in XML mapping files
  - May use one or more mapping files
  - Can define mapping defaults
    - For all of the entities in the persistence unit
    - For only the entities listed in the mapping file
  - Annotations on entities not required
  - At runtime XML can override annotations
    - May override annotated mappings with XML mappings
    - May disable all annotations on entities in persistence unit

Adding the Entities

Mapping File

```
<entity-mappings>
  <entity class="Passenger">
    <id name="id">
      <generated-value/>
    </id>
    <many-to-one name="flight"/>
  </entity>
  <entity class="Flight">
    <id name="id"/>
    <basic name="destination">
      <column name="DEST"/>
    </basic>
    <one-to-many name="passengers" mapped-by="flight"/>
  </entity>
</entity-mappings>
```

Adding the Entities

Packaging Mapping Files

- Mapping files can be anywhere on the classpath of the module
  - Default file name orm.xml in META-INF in JAR
  - Other mapping files must be listed in persistence.xml

airlineEjb.jar

AirlineReservation.class
AirlineReservationBean.class
Flight.class
Passenger.class
META-INF/persistence.xml
META-INF/orm.xml

Agenda

Introduction to EJB 3.0 Specification

Building Components

Building Entities

Packaging Up the Components

Adding the Entities

Other Packaging Options

Summary

Other Packaging Options

Persistence Archive

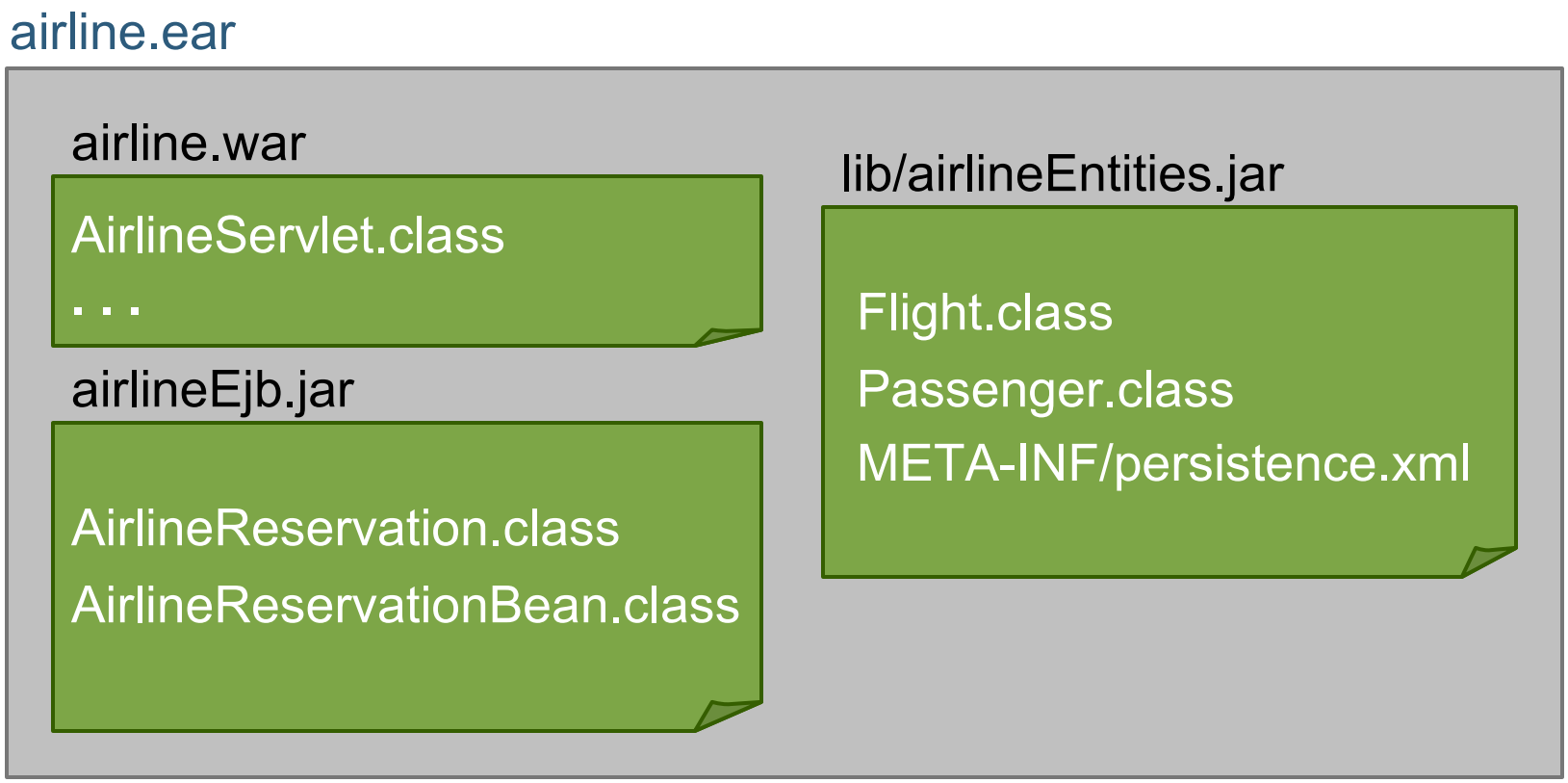
- Can create a separate persistence archive
  - Put all of the persistence artifacts in the JAR
  - Put archive in library directory in the application EAR
  - Archive is shared amongst all application modules

airlineEntities.jar

Flight.class
Passenger.class
META-INF/persistence.xml
META-INF/orm.xml

Other Packaging Options

Airline Application



Other Packaging Options

Persistence Library Using Mapping File

- Can create a persistence library
  - Leave persistence.xml and mapping file in EJB based JAR
  - Put all of the entities in a separate JAR
  - Put JAR on classpath (library directory of the EAR)
  - Library is shared amongst all application modules

airlineEjb.jar

AirlineReservation.class
AirlineReservationBean.class
META-INF/persistence.xml
META-INF/orm.xml

airlineEntities.jar

Flight.class
Passenger.class

Other Packaging Options

Persistence Library Using Annotated Mappings

- When annotations are used for mapping, the entities must be referenced in persistence.xml
  - No mapping file included in EJB based JAR or library JAR
  - May be referenced by class or by library JAR
  - Classes or JAR must be on module classpath

airlineEjb.jar

AirlineReservation.class
AirlineReservationBean.class
META-INF/persistence.xml

airlineEntities.jar

Flight.class
Passenger.class

Other Packaging Options

Referencing Entities From persistence.xml

- By Class

```
<persistence-unit name="AirlineReservation">
  ...
  <class>Flight</class>
  <class>Passenger</class>
</persistence-unit>
```

- By JAR

```
<persistence-unit name="AirlineReservation">
  ...
  <jar-file>airlineEntities.jar</jar-file>
</persistence-unit>
```

Agenda

Introduction to EJB 3.0 Specification

Building Components

Building Entities

Packaging Up the Components

Adding the Entities

Other Packaging Options

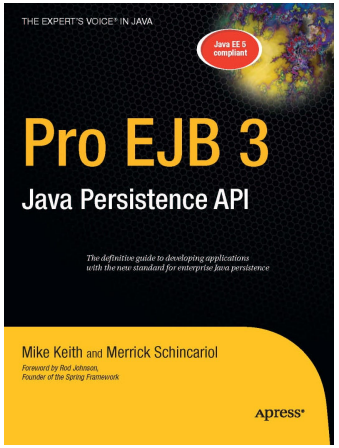
Summary

Summary

- EJB 3.0 based components and entities are easy to develop, build, package and deploy
- EJB based components and persistent entities can be packaged together or separately
- Annotations make development simpler and can facilitate practical XML-less deployment
- XML may be used instead of annotations or to override annotated mappings
- EJB technology and persistence packaging is simple for most applications and flexible enough to handle advanced requirements

Sessions and Resources

- Sessions
  - TS-9056 Java Persistence API in 60 Minutes
    Fri. @ 2:30
- Papers and Tutorials
  - http://otn.oracle.com/ejb3
- Books
  - Pro EJB 3:
    Java Persistence API (Apress)
  - Enterprise JavaBeans 3.0 (O'Reilly)



Q&A

Building EJB™ 3.0 Applications: A Simple Matter of Point and Squish

Mike Keith
Merrick Schincariol

Oracle Corp.
http://otn.oracle.com/ejb3

TS-3616