Java™ Persistence API in 60 Minutes

Marina Vatkina, Sun Microsystems
Mitesh Meswani, Sun Microsystems
Pramod Gopinath, Sun Microsystems

http://glassfish.dev.java.net

Session TS-9056

Overall Presentation Goal

A head start on using Java™ Persistence API

Agenda

O-R Mapping

EntityManager and PersistenceContext

Query

EntityManager Types

Packaging

Agenda

O-R Mapping

EntityManager and PersistenceContext

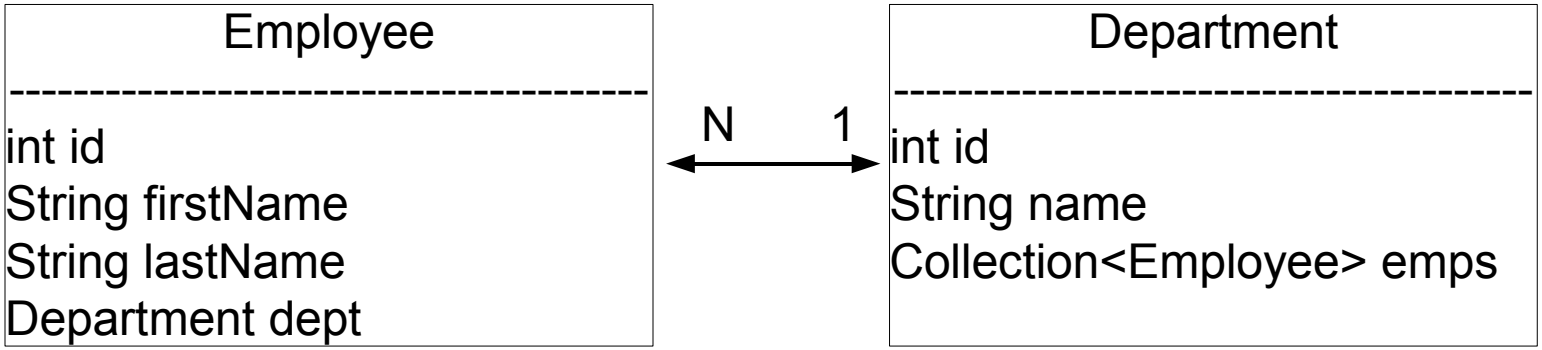
Query

EntityManager Types

Packaging

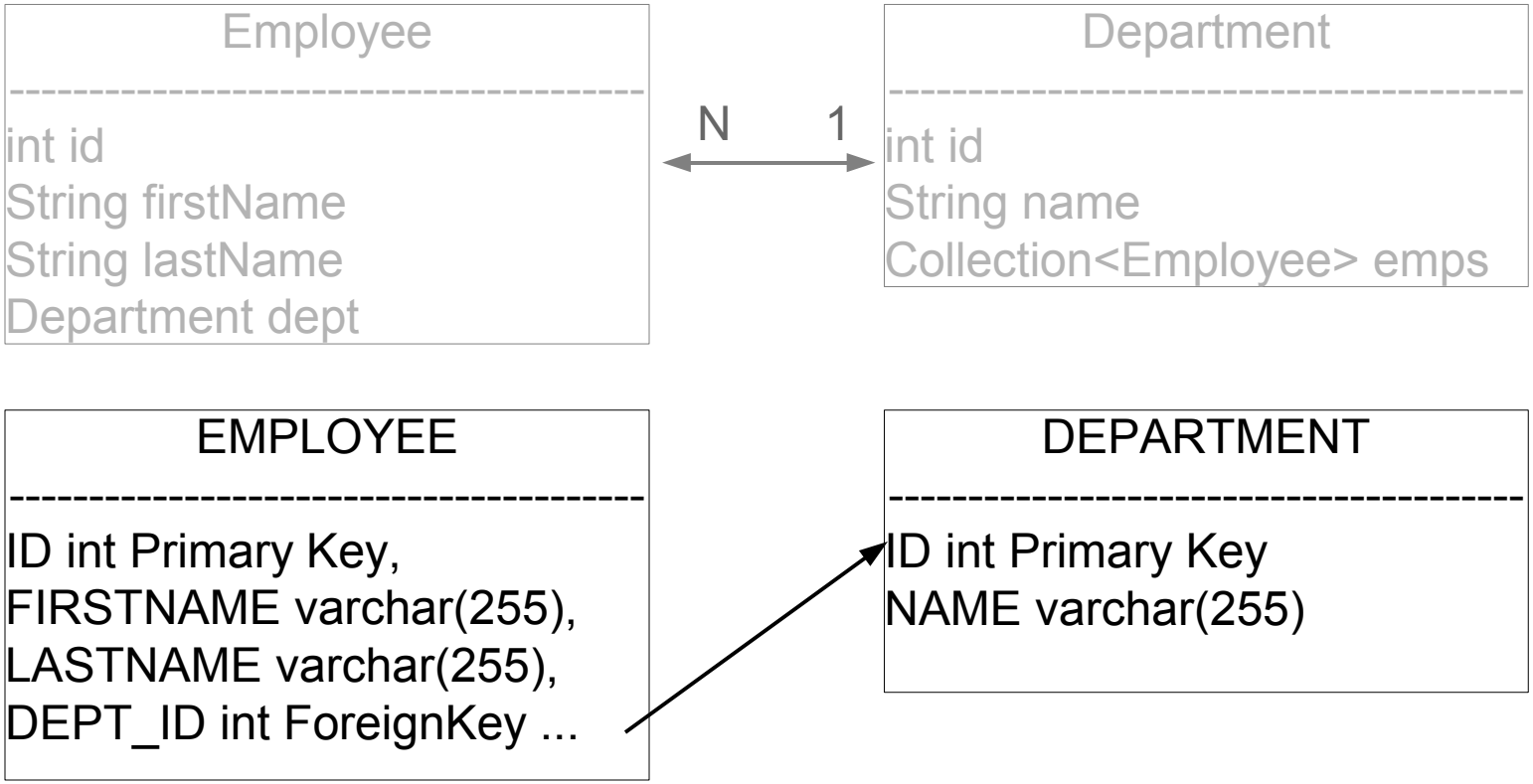
O-R Mapping

Example Data Model



O-R Mapping

Example Data Model



Entity Class for Employee

```
@Entity class Employee {
    @Id private int id;
    private String firstName;
    private String lastName;

    pubic Employee (int id, String firstName, String
           lastName) {
       ...
    }
    public String getFullName(){
           return firstName + lastName;
    }
   ...
}
```

Entity Class for Employee

```
@Table(name="EMPLOYEE")
@Entity class Employee {
    @Id private int id;
    private String firstName;
    @Column(name="LASTNAME")
    private String lastName;

    pubic Employee (int id, String firstName, String
          lastName) {
       ...
    }
    public String getFullName(){
          return firstName + lastName;
    }
   ...
}
```

Entity Class for Employee

```
@Entity class Employee {
    @Id private int id;
    private String firstName;
    private String lastName;
    @ManyToOne
    private Department dept;
    pubic Employee (int id, String firstName, String
          lastName) {
       ...
    }
    ...
    public void setDepartment(Department department){
      dept = department;
    }
    public Department getDepartment(){
          return dept;
    }
}
```

Entity Class for Department

```
@Entity class Department {
    @Id private int id;
    private String name;
    @OneToMany(mappedBy = "dept")
    private Collection<Employee> emps = new ...;

    pubic Department (int id, String Name){
       ...
    }

    public void addEmployee(Employee e){
      emps.add(e);
    }

    public void removeEmployee(Employee e){
      emps.remove(e);
    }
}
```

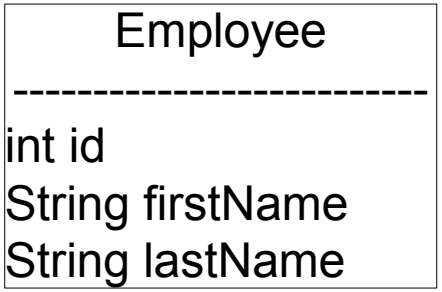
Entity Class for Department—Property-Based Persistence

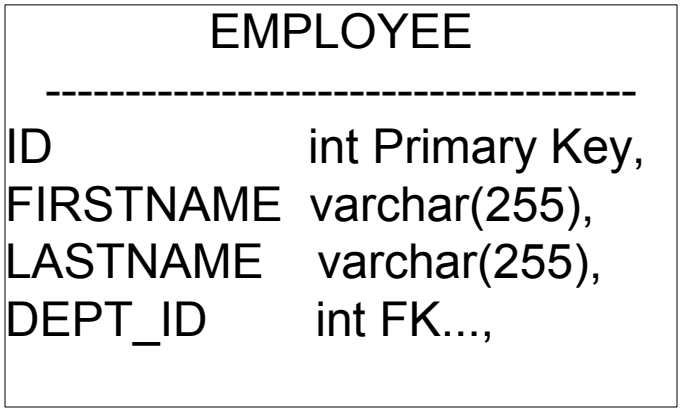
```
@Entity class Department {
  private int id;
  private String name;

  // property 'id'
  @Id public void getId() { return id; }
  public void setId(int id) { this.id = id; }

  ...
}
```

Entity Class for Department—Property-Based Persistence

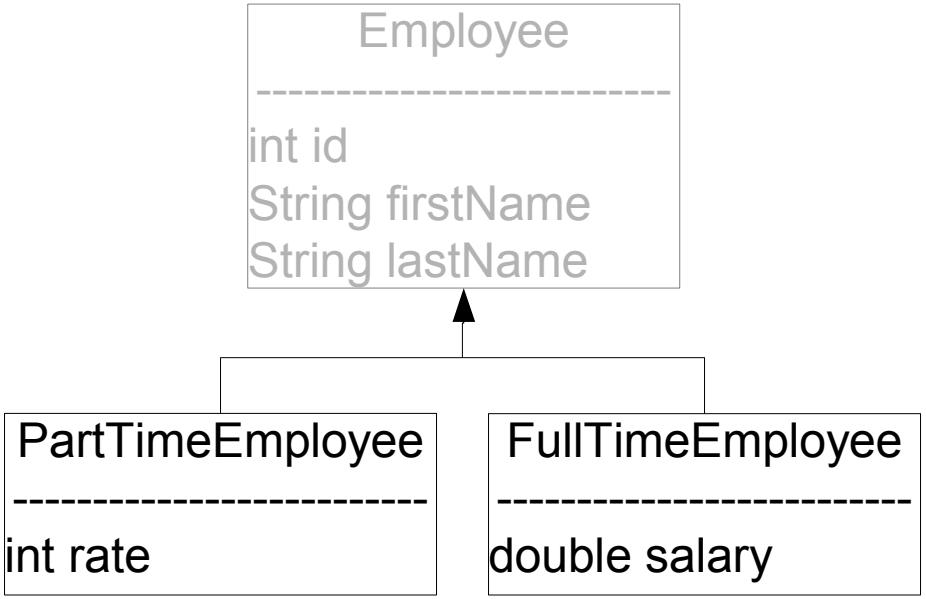
```
@Entity class Department {
  private int id;
  private String name;
  Collection<Employee> emps;

  // property 'id'
  @Id public void getId() { return id; }
  public void setId(int id) { this.id = id; }

  // property 'name'
  public String getName() { return name; }
  public void setName(String name) { this.name = name}

  // property 'emps'
  @OneToMany(mappedBy = "dept")
  public Collection<Employee> getEmps() { return emps; }
  public void setEmps(Collection<Employee> emps)
  { this.emps = emps; }

  // Business methods stay the same

}
```

Inheritance





Inheritance



EMPLOYEE

ID	int Primary Key,
FIRSTNAME	varchar(255),
LASTNAME	varchar(255),
DEPT_ID	int FK...,
RATE	int NULL,
SALARY	decimal NULL,
DISCRIM	varchar(30)

Entity Classes with Inheritance

```
@Entity
public abstract class Employee {
   @Id int id;

   ...
}
```

Entity Classes with Inheritance

```
@Entity
@DiscriminatorColumn(name = "discrim")
public abstract class Employee{
  @Id int id;
  ...
}

@Entity
public class FullTimeEmployee extends Employee {

  private double salary;
}
```

Entity Classes with Inheritance

```
@Entity
@DiscriminatorColumn(name = "discrim")
public abstract class Employee{

  ...
}

@Entity
//@DiscrimiatorValue defaults to "FullTimeEmployee"
public class FullTimeEmployee extends Employee {

  private double salary;
}

@Entity
@DiscriminatorValue("PT")
public class PartTimeEmployee extends Employee {

  private int rate;
}
```

O-R Mapping

Advanced Concepts

- Embedded fields: @Embeddable, @Embedded
- @Id vs. @EmbeddedId vs. @IdClass
- @GeneratedValue
  - GenerationType
    TABLE, SEQUENCE, IDENTITY,AUTO
- Mapping overrides (using external mapping files)
- Multiple tables: @SecondaryTable
- FetchType: EAGER, LAZY
- DDL generation: standard vs provider specific

Agenda

O-R Mapping

EntityManager and PersistenceContext

Query

EntityManager Types

Packaging

EntityManager

- The interface used to communicate with Persistence runtime

```
// Create a new Department entity
Department d = new Department(1, "sales");

// Make the entity persistent
em.persist(d);

// Find a Department entity
Department d1 = em.find(Department.class, 1);

// Delete the Department entity
em.remove(d1);
```

Persistence Context

- It represents a collection of entities managed by an EntityManager
- For each persistent entity identity there is a unique entity instance in a given Persistence Context

Persistence Context Types

- Transaction Scoped Persistence Context
  - Entities managed by a persistence context become detached at end of the transaction
  - Queries can run outside a transaction; fetched objects are detached
- Extended Persistence Context
  - Entities remain managed for the life of EntityManager
  - Non transactional changes allowed

EntityManager—Advanced Operations

```
// Flush the current persistence context into database
em.flush();

// Refresh the entity from database overwriting any
// changes
d.setName("Marketing");
em.refresh(d);

// Clear the current persistence context. All managed
// entities become detached
em.clear();
```

Updates

- What is an update?

```
Department d = em.find(Department.class, 1);

// Modifying a managed entity
d.setName("newName");
```

- When are updates flushed?
  - Before a query is executed
  - Transaction commit
  - Explicit flush using em.flush()

Detach and Merge

- Detach
  - Persistence Context ends
  - em.clear() is called
  - deserialized
- Merge

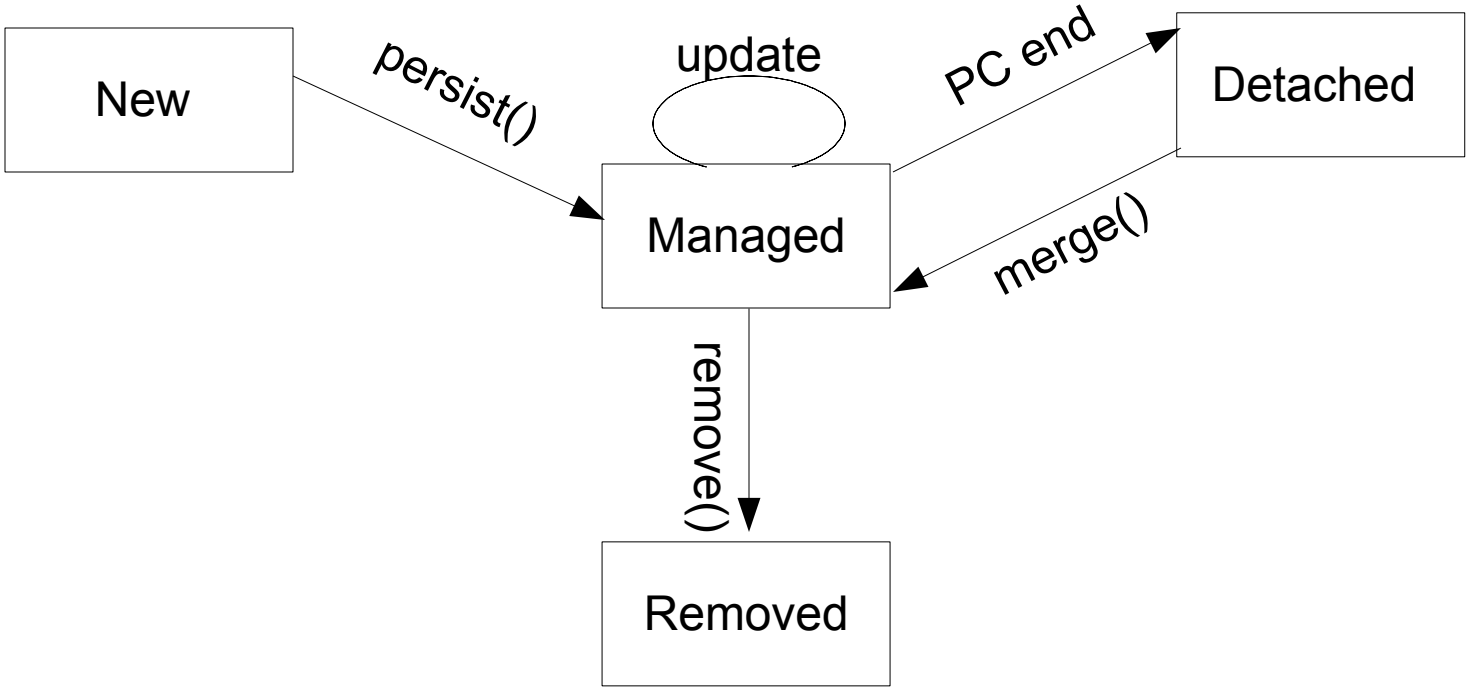
```
Department updateDepartment(Department d) {
 assert (em.contains(d) == false);
 Department d1 = em.merge(d);
 ...
 return d1;
}
```

Things to Remember About Updates

```
// Detached entities need to be merged into current
// persistence context to be managed
void deleteDepartment(Department d) {

  // em.remove(d); - does not work

  Department d1 = em.merge(d);
  em.remove(d1);
}
```

```
// Relationships need to be updated on both sides
Department d = em.find(Department.class, 1);
Employee   e = em.find(Employee.class, 10);

e.setDepartment(d); // Owning side
d.addEmployee(e);   // Dependent side
```

Lifecycle of an Entity



Agenda

O-R mapping

EntityManager and PersistenceContext

Query

EntityManager Types

Packaging

Queries

Java Persistence Query Language

- SQL-like Query language
- Operates over abstract persistence schema of the entities
- Provides functionality to select, update, and delete
- Portable across databases

Using Query API to Retrieve Data

```
// Retrieve a collection
Query q = em.createQuery(
"select d from Department d where d.id > :deptid");

q.setParameter("deptid", 1);
Collection departments = q.getResultList();
```

Using Query API to Retrieve Data

```
// Retrieve a collection
Query q = em.createQuery(
"select d from Department d where d.id > :deptid");

q.setParameter("deptid", 1);
Collection departments = q.getResultList();


// Retrieve a single instance
Query q = em.createQuery(
"select d from Department d where d.id = :deptid");

q.setParameter("deptid", 1);

Department d = q.getSingleResult();
```

Bulk Update and Delete

```
// Bulk update: give full-time employees a raise
Query q = em.createQuery(
      "update FullTimeEmployee e
       set e.salary = e.salary * 1.10");

int noOfEmployeesUpdated = q.executeUpdate();


// Bulk delete: delete 2 old departments
Query q = em.createQuery(
"delete from department d where d.id < 3");

int noOfDepartmentDeleted = q.executeUpdate();
```

Queries: Things to Remember

- Queries are polymorphic

```
// Retrieves instances of PartTimeEmployee and
// FullTimeEmployee
"select e from Employee e where e.firstName like 'Jo%'"
```

- Use Named queries

```
@NamedQuery(
  name  = "findAllEmployeesByName",
  query = "select e from Employee e where
           e.firstName = :name"
)
```

- Name is PersistenceUnit scoped

```
// It is a good practice to fully qualify the name
@NamedQuery(name="com.acme.hr.Employee.findByName",
                    ....)
```

SQL Generated for Queries

- Using a String function
  - **Persistence Query:**

```
select e from Employee e where
locate(e.firstName, 'J') = 1
```

  - **SQL on Oracle:**

```
select e.id, e.firstName, e.lastName, e.dept_id from
Employee e where
INSTR(e.firstName, 'J') = 1
```

  - **SQL on Sybase:**

```
select e.id, e.firstName, e.lastName, e.dept_id from
Employee e where
CHARINDEX(e.firstName, 'J') = 1
```

SQL Generated for Queries

- Relationship navigation
  - **Persistence Query:**
    ```
    select e from Employee e where
    e.department.name = “sales”
    ```
  - **SQL:**
    ```
    select e.id, e.firstName, e.lastName from Employee e,
    Department d where
    e.dept_id = d.id and d.name = "sales"
    ```

SQL Generated for Queries

- Fetch Joins
  - **Persistence Query:**

```
select d from Department d left join fetch d.emps
where d.name = "sales"
```

  - **SQL:**

```
select d.id, d.name,
e.id, e.firstName, e.lastName, e.dept_id
from
Department d left outer join Employee e on
d.id = e.dept_id
where d.name = 'sales'
```

Using Native SQL

- Native SQL can be used when Java Persistence Query Language is not sufficient

```
// Create a query using native sql
Query q = em.createNativeQuery(
          "select d.ID, d.NAME from DEPARTMENT d where
           <A db specific clause>", Department.class);

// Create a query using native sql
Collection departments = q.getResultList();
```

- @SQLResultSetMapping needs to be specified for queries where result is not a single entity class
- Non-portable across databases!

Advanced Query API

- Scrolling through result set

```
Query q = em.createQuery("select e from Employees e
                                order by e.id ")
q.setFirstResult(5);
q.setMaxResult(10);
Collection<Employee> emps = q.getResultList();
```

- Query hints
  - Provider-specific

```
q.setHint("toplink.pessimistic-lock", 1 );
```

Agenda

O-R mapping

EntityManager and PersistenceContext

Query

EntityManager Types

Packaging

EntityManager Types

- Can be classified based on the transaction type
  - JTA
  - RESOURCE_LOCAL
- Can also be classified based on lifecycle
  - Container managed
    - Obtained by Injection or JNDI lookup
    - Always JTA
  - Application managed
    - Created using EntityManagerFactory
    - May be either JTA or RESOURCE_LOCAL

EntityManager Types

	Container Managed	Application Managed

JTA	1. Injection
	`@Stateless`
	`public class MyBean {`
	`@PersistenceContext(`
	`unitName = "HumanResource")`
	`private EntityManager em;`
	`}`
	--- 	
	2. Lookup
	`@PersistenceContext(name="foo",`
	`unitName="Human..")`
	`public class MyServlet ... {`
	`public void doGet(...) {`
	`InitialContext ic = ...;`
	`EntityManager em =`
	`(EntityManager)ic.lookup(`
	`"java:comp/env/foo");`
	`....`
	`}`
	`}`	

Resource Local		

EntityManager Types

	Container Managed	Application Managed

JTA		`@Stateless`
		`public class MyBean {`
		`  @PersistenceUnit(`
		`  unitName = "HumanResource")`
		`  private EntityManagerFactory emf;`
		
		`  void doSomething() {`
		`  EntityManager em =`
		`  emf.createEntityManager(); // 1`
		`  utx.begin(); // 2`
		`  // Not needed if 1 is after 2`
		`  em.joinTransaction();`
		`  ...`
		
		`  utx.commit();`
		`  ...`
		`  em.close()`
		`  }`
		`}`

Resource Local		

EntityManager Types

	Container Managed	Application Managed
JTA		

Resource Local	N/A	Stateless
		public class MyBean {
		@PersistenceUnit(
		unitName = "HumanResource")
		private EntityManagerFactory emf;
		
		void doSomething() {
		EntityManager em =
		emf.createEntityManager();
		...
		EntityTransaction tx =
		em.getTransaction();
		tx.begin()
		...
		tx.commit();
		...
		em.close()
		}
		}

Obtaining an EntityManagerFactory

- Java EE environment
  - Injected or looked up from JNDI

```
@PersistenceUnit EntiyManagerFactory emf;
```

  - Container manages the lifecycle of emf
- Java SE environment, non-Java EE Container
  - Application manages the lifecycle of emf

```
properties.load(...);
EntityManagerFactory emf =
   Persistence.createEntityManagerFactory("Human...",
          properties);
  .....
 emf.close();
```

Agenda

O-R mapping

EntityManager and PersistenceContext

Query

EntityManager Types

Packaging

Persistence Unit

- A set of managed classes that are mapped to a single database and their mapping metadata
- An entity manager factory and all entity managers obtained from it
- Is defined in the persistence.xml file

Persistence Unit Root in Java EE

- Is the jar file or directory whose META-INF directory contains the persistence.xml file
- Possible locations are
  - An EJB-JAR file
  - The WEB-INF/classes directory of a WAR file
  - A jar file in the WEB-INF/lib directory of a WAR file
  - A jar file in the root of the EAR
  - A jar file in the EAR library directory
  - An application client jar file

persistence.xml for Java EE

```
<persistence xmlns="..." version="1.0">
    <!-- Unit name identifies a persistence unit -->
    <persistence-unit name="HumanResource">
    ...
    </persistence-unit>
</persistence>
```

persistence.xml for Java EE

```
<persistence xmlns="..." version="1.0">
    <!-- Unit name identifies a persistence unit -->
    <persistence-unit name="HumanResource">
      <!-- The database connection -->
      <jta-data-source>jdbc/MyDB</jta-data-source>
      ....
  </persistence-unit>
</persistence>
```

persistence.xml for Java EE

```
<persistence xmlns="..." version="1.0">
    <!-- Unit name identifies a persistence unit -->
    <persistence-unit name="HumanResource">
      <!-- The database connection -->
      <jta-data-source>jdbc/MyDB</jta-data-source>
      <!-- optional O-R mapping file -->
      <mapping-file>entities-mapping.xml</mapping-file>
     ...
    </persistence-unit>
</persistence>
```

persistence.xml for Java EE

```
<persistence xmlns="..." version="1.0">
    <!-- Unit name identifies a persistence unit -->
    <persistence-unit name="HumanResource">
      <!-- The database connection -->
      <jta-data-source>jdbc/MyDB</jta-data-source>
      <!-- optional O-R mapping file -->
      <mapping-file>entities-mapping.xml</mapping-file>
      <!-- Pointers to other persistence classes -->
      <jar-file>myLibraryEntities.jar</jar-file>
    </persistence-unit>
</persistence>
```

persistence.xml for Java EE

```
<persistence xmlns="..." version="1.0">
    <!-- Unit name identifies a persistence unit -->
    <persistence-unit name="HumanResource">
      <!-- The database connection -->
      <jta-data-source>jdbc/MyDB</jta-data-source>
      <!-- optional O-R mapping file -->
      <mapping-file>entities-mapping.xml</mapping-file>
      <!-- Pointers to other persistence classes -->
      <jar-file>myLibraryEntities.jar</jar-file>
    </persistence-unit>

    <!-- There can be more than one persistence units -->
    <persistence-unit name="OrderManagement">
        ...
    </persistence-unit>
</persistence>
```

persistence.xml for Java SE

```
<persistence xmlns="..." version="1.0">
    <!-- Unit name identifies a persistence unit -->
    <persistence-unit name="HumanResource">
      <!-- The persistence provider -->
      <provider>com.acme.AcmePersistence.class</provider>
      <properties>
        <!--provider specific properties -->
        <!--Includes database connection properties -->
      </properties>
      <!-- List of persistence classes -->
      <class>com.acme.hr.Employee.class</class>
      <class>com.acme.hr.Department.class</class>
    </persistence-unit>
</persistence>
```

For More Information

- Java Persistence API specification
  http://jcp.org/en/jsr/detail?id=220
- Java EE reference implementation Project Glassfish
  http://glassfish.dev.java.net
- NetBeans IDE supports Java Persistence API
  http://www.netbeans.org/community/releases/55/
- Persistence page for Project Glassfish
  http://glassfish.dev.java.net/javaee5/persistence
- Simple Persistence examples
  http://glassfish.dev.java.net/javaee5/persistence/persistence-example.html

Project GlassFish

Building a Java EE 5 Open Source Application Server

- Simplifying Java application development with Java EE 5 technologies
- Includes JWSDP, EJB 3.0, JSF 1.2, JAX-WS and JAX-B 2.0
- Supports > 20 frameworks and apps
- Open source CDDL license
- Basics for the Sun Java System Application Server PE 9
- Free to download and free to deploy
- Over 1,200 members and 200,000 downloads
- Integrated with NetBeans

Source: Sun February 2006; see website for latest stats

Q&A

Marina Vatkina
Mitesh Meswani
Pramod Gopinath

Java™ Persistence API in 60 Minutes

Marina Vatkina, Sun Microsystems
Mitesh Meswani, Sun Microsystems
Pramod Gopinath, Sun Microsystems

http://glassfish.dev.java.net

Session TS-9056

Backup Slides

ID Generation

- Options
  - Specified by application
  - Generated by persistence runtime
- Generation types
  - SEQUENCE, IDENTITY, TABLE, AUTO
- Generation type TABLE

```
@Entity public class Employee {
  ...
  @Id
  @GeneratedValue
  public int id;

}
```

ID Generation

- Generation type TABLE

```
@TableGenerator(
  name="hrIDGenerator", table="ID_GENERATOR_TBL",
  pkColumnName="KEYS",
  valueColumnName="VALUES",
  pkColumnValue="EmployeeKeys")
@Entity public class Employee {
  ...
  @Id
  @GeneratedValue(
   strategy=TABLE, generator="hrIDGenerator")
  public int id;

}
```

Packaging Examples

- In an EJB-JAR file

```
app.ear
  ejb.jar
    META-INF/persistence.xml
    com.wombat.HrManager       //business interface of slsb
    com.wombat.HrManagerImpl   // impl. of slsb
    com.acme.hr.Employee       // persistence class
    com.acme.hr.Department     // persistence class

  ejb_client.jar
    META-INF/MANIFEST.MF
    com.acme.hr.HrApp // has main() method
```

Packaging Examples

- In a WAR file

```
web.war
  WEB-INF/classes
            META-INF/persistence.xml
            com.acme.hr.HrServlet  // The servlet class
            com.acme.hr.Employee   // Persistence class
            com.acme.hr.Department // Persistence class
```

Packaging Examples

- In a lib directory of an EAR file
-
```
app.ear
  lib/entities.jar          //Has META-INF/persistence.xml
  lib/ejb-interfaces.jar
  ejbs.jar          //entities from entities.jar accessible
  web-app2.war      //entities from entities.jar accessible
  appclient.jar
```

Persistence Context

- Is associated with an EntityManger

```
// find a department with identity 1
Department d1 = em.find(Department.class, 1);

// find once again a department with the same identity
// in the same persistence context
Department d2 = em.find(Department.class, 1);


// Same instance of Department associated with the given
// identity
assert ( d1 == d2);
```

Container Managed EM

- Container manages the life cycle
- Injected

```
@Stateless
public class MyBean {
   @PersistenceContext(unitName = "HumanResource")
   private EntityManager em;
}
```

- Looked up from JNDI

```
@PersistenceContext(name="foo", unitName="Human..")
public class MyServlet extends HttpServelet {
   public void doGet(...)  {
      InitialContext ic = new InitialContext();
      EntityManager em = (EntityManager)
            ic.lookup("java:comp/env/foo");
      ....
   }
}
```

Application Managed EM

- Application manages the life cycle
- Created from EntityMangerFactory

```
public class MyServlet extends HttpServlet {

  @PersistenceUnit(unitName="HumanResource")
  private EntityManagerFactory emf;

  public void doGet(...)  {
      EntityManager em = emf.createEntityManager();
      ....
      em.close();
  }
}
```

JTA EntityManager

- Transactions are controlled through JTA
- Container Managed EntityManager
  - is always JTA
  - container registers em for transaction synchronization
- Application managed EntityManager
  - Application is responsible for transaction synchronization

```
EntityManager em = emf.createEntityManager(); // 1
utx.begin();                                  // 2
em.joinTransaction(); // Not needed if 2 is before 1
...
utx.commit();
em.close();
```

RESOURCE_LOCAL EntityManager

- Transactions are controlled by the application
- Always application managed
- Uses EntityTransaction API

```
EntityTransaction tx = em.getTransaction();
tx.begin()
   ...
tx.commit();
```

- Usually used in Java SE environment