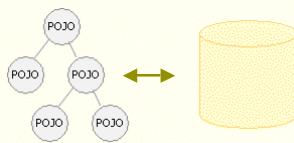




Java Persistence API v1.0

a standard for *ORM frameworks*
and *POJO based Persistence*



Magnus Larsson
ml@callistaenterprise.se



Agenda

- Bored of writing Data Access Objects?
- Terminology
 - ORM - Object Relational Mapping frameworks
 - POJO based persistence
- Introducing JPA...
 - What is it?
 - Examples!
- Details & Guidelines
 - Until time runs out...
- Summary



Bored of writing Data Access Objects?

- A sample DAO method – “find by Primary Key”

```
public SampleDTO sampleLookup(String id) {
    Connection      c = null;
    PreparedStatement ps = null;
    ResultSet        rs = null;
    SampleDTO        dto = null;

    try {
        c = getDataSource().getConnection();
        ps = c.prepareStatement("SELECT ...");
        ps.setString(1, id);
        rs = ps.executeQuery();
        if (rs.first()) {
            dto = new SampleDTO(id, rs.getString(1), rs.getString(2), rs.getString(2));
        }
    } catch (SQLException se) {
        throw new SampleDAORuntimeException(se);
    } finally {
        if (rs != null) try {rs.close();} catch (SQLException se) {}
        if (ps != null) try {ps.close();} catch (SQLException se) {}
        if (c != null) try {c.close();} catch (SQLException se) {}
    }
    return dto;
}
```

Cadec 2007 - Java Persistence API v1.0, Slide 3
Copyright 2007, Callista Enterprise AB



Bored of writing DAO's?

- What about instead looking up an entity by

```
Sample entity = entityManager.find(Sample.class, id);
```

- Or inserting an entity in the database by

```
Sample entity = new Sample();
entity.setAttr1(attr1);
entity.setAttr2(attr2);

entityManager.persist(entity);
```

- Or updating an entity in the database by

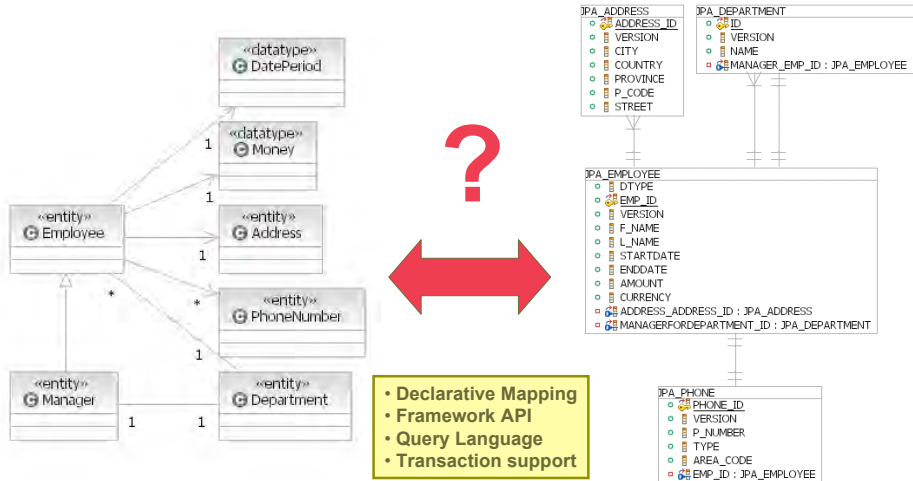
```
entity.setAttr(newValue);
```

- ...and doing it based on a standard...

Cadec 2007 - Java Persistence API v1.0, Slide 4
Copyright 2007, Callista Enterprise AB



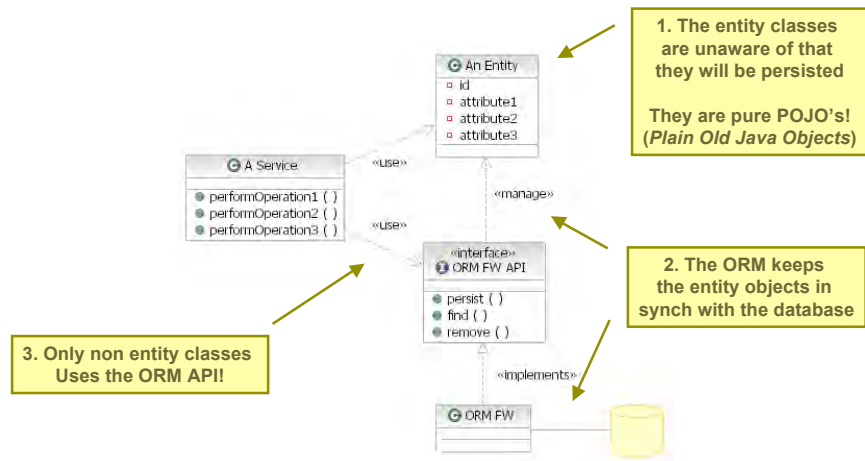
ORM - Object Relational Mapping framework



Cadec 2007 - Java Persistence API v1.0, Slide 5
 Copyright 2007, Callista Enterprise AB



POJO based persistence



NOTE: Also known as "transparent persistence"

Cadec 2007 - Java Persistence API v1.0, Slide 6
 Copyright 2007, Callista Enterprise AB



POJO based persistence

```
public class Employee {
    private Long id;
    private Long version;
    private String firstName;
    private String lastName;
    private Money salary;
    private Address address;

    // Setters and getters
    // left out ☺
}
```

3. Only non entity classes
Uses the ORM API!



1. The entity classes
are unaware of that
they will be persisted

They are pure POJO's!
(Plain Old Java Objects)

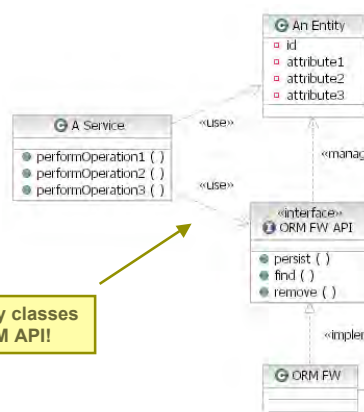
2. The ORM keeps
the entity objects in
synch with the database

NOTE: Also known as
"transparent persistence"

Cadec 2007 - Java Persistence API v1.0, Slide 7
Copyright 2007, Callista Enterprise AB



POJO based persistence



3. Only non entity classes
Uses the ORM API!

1. The entity classes
are unaware of that
they will be persisted

They are pure POJO's!
(Plain Old Java Objects)

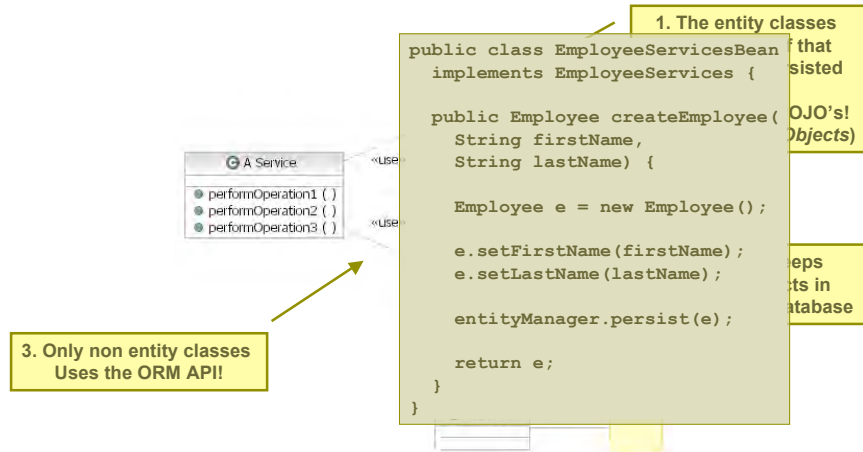
2. The ORM keeps
the entity objects in
synch with the database

NOTE: Also known as
"transparent persistence"

Cadec 2007 - Java Persistence API v1.0, Slide 8
Copyright 2007, Callista Enterprise AB



POJO based persistence

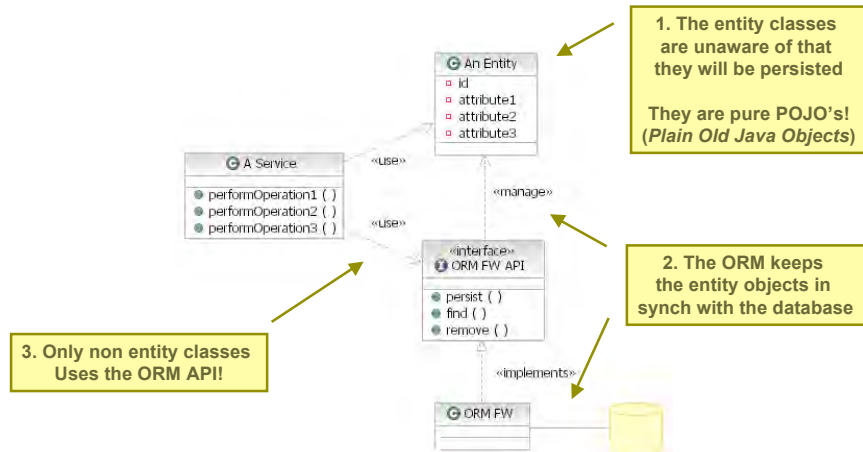


Cadec 2007 - Java Persistence API v1.0, Slide 9
 Copyright 2007, Callista Enterprise AB

NOTE: Also known as "transparent persistence"



POJO based persistence



Cadec 2007 - Java Persistence API v1.0, Slide 10
 Copyright 2007, Callista Enterprise AB

NOTE: Also known as "transparent persistence"



Where are we?

- Bored of writing Data Access Objects?
- Terminology
 - ORM - Object Relational Mapping frameworks
 - POJO based persistence
- **Introducing JPA...**
 - **What is it?**
 - **Examples!**
- Details & Guidelines
 - Until time runs out...
- Summary

Cadec 2007 - Java Persistence API v1.0, Slide 11
Copyright 2007, Callista Enterprise AB



JPA 1.0 – Java Persistence API

- What is it?
 - A standard for Object Relational Mapping frameworks
 - Based on "*POJO based persistence*"
 - Modeled to a great deal after JBoss Hibernate
 - Released in May 2006
 - As a part of EJB 3.0 and Java EE 5.0
 - Replaces EJB 2.x Entity Beans in EJB 3.0
 - Runs on Java SE 5.0
 - JPA does not require neither EJB 3.0 nor Java EE 5.0!
 - Pluggable in Java EE 5.0
 - The JPA implementation can be replaced in Java EE 5.0

Cadec 2007 - Java Persistence API v1.0, Slide 12
Copyright 2007, Callista Enterprise AB



JPA 1.0 – Java Persistence API

- JPA 1.0 implementations

- Sun Glassfish Toplink Essentials
 - JBoss Hibernate
 - Apache Open JPA
 - Oracle Toplink
 - BEA Kodo
 - SAP JPA
 - CocoBase
 - JPOX
-
- ```
graph TD; A[Sun Glassfish Toplink Essentials] -- based on --> B[Apache Open JPA]; B -- based on --> C[BEA Kodo]; D[JBoss Hibernate] -- based on --> E[Oracle Toplink];
```

Cadec 2007 - Java Persistence API v1.0, Slide 13  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 – Key Features

- Key Features

- Declarative mapping
  - Between entity object model and database schema
- A Manager API, *Entity Manager*
  - For persisting, finding and removing objects
  - Handles Session and Cache management
- A query language, *Java Persistence QL*
  - Similar to Hibernate QL
  - Operates on the entity object model
- Support for transactions
  - Both JPA resource local and JTA/EJB CMT global transactions

Cadec 2007 - Java Persistence API v1.0, Slide 14  
Copyright 2007, Callista Enterprise AB



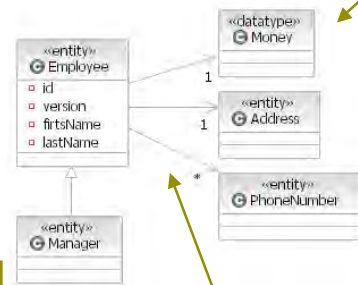
## JPA 1.0 - Declarative Object Relational Mapping

1. Classes are mapped to tables (including inheritance)

5. Use either annotations or deployment descriptors

4. Support for embedded objects

2. Fields are mapped to columns (including pk, auto pk, version and enum)



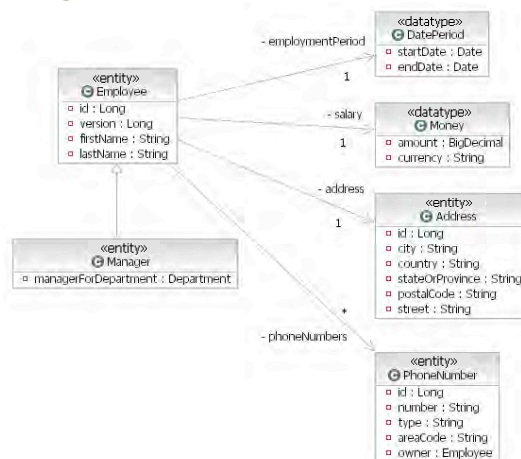
3. All types of relationships are supported (1-1, 1-M, M-M, uni- and bi-directional)

Cadec 2007 - Java Persistence API v1.0, Slide 15  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 - Examples

- Class Diagram



Cadec 2007 - Java Persistence API v1.0, Slide 16  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 - Examples

- Entity Employee with friends...

```
@Entity
@Inheritance(strategy=SINGLE_TABLE)
public class Employee {

 @Id
 @GeneratedValue(strategy=AUTO)
 private Long id;

 @Version private Long version;
 @Column(length=20)
 private String firstName;
 private String lastName;
 @Embedded private Money salary;
 @Embedded private DatePeriod employmentPeriod;

 @OneToOne
 private Address address;

 @OneToMany(mappedBy = "owner")
 private List<PhoneNumber> phoneNumbers;

 // Setters and getters left out ☺
```

```
@Entity
public class Manager extends Employee {
 @OneToOne
 private Department mgrForDept;
```

```
@Embeddable
public class Money {
 public enum CurrencyEnum
 {SEK, EUR, USD};

 private BigDecimal amount;
 private CurrencyEnum currency;
```

Cadec 2007 - Java Persistence API v1.0, Slide 17  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 - Examples

- Table mapping

```
@Entity
@Inheritance(strategy=SINGLE_TABLE)
public class Employee {

 @Id
 @GeneratedValue(strategy=AUTO)
 private Long id;

 @Version private Long version;
 @Column(length=20)
 private String firstName;
 private String lastName;

 @Embedded private Money salary;
 @Embedded private DatePeriod employmentPeriod;

 @OneToOne
 private Address address;

 @OneToMany(mappedBy = "owner")
 private List<PhoneNumber> phoneNumbers;

 // Setters and getters left out to save some space ☺
```

```
CREATE TABLE "EMPLOYEE" (
 "ID" BIGINT NOT NULL,
 "VERSION" BIGINT,
 "FIRSTNAME" VARCHAR(20),
 "LASTNAME" VARCHAR(255),
 "AMOUNT" NUMERIC(19, 2),
 "CURRENCY" INTEGER,
 "STARTDATE" DATE,
 "ENDDATE" DATE,
 "ADDRESS_ADDRESS_ID" BIGINT,
 "DTYPE" VARCHAR(255) NOT NULL,
 "MGRFORDEPT_ID" BIGINT
)
```

Cadec 2007 - Java Persistence API v1.0, Slide 18  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 - Examples

- Table mapping

```
@Entity
@Inheritance(strategy=SINGLE_TABLE)
public class Employee {

 @Id
 @GeneratedValue(strategy=AUTO)
 private Long id;

 @Version
 private Long version;
 @Column(length=20)
 private String firstName;
 private String lastName;

 @Embedded
 private Money salary;
 @Embedded
 private DatePeriod employmentPeriod;

 @OneToOne
 private Address address;
}
```

```
CREATE TABLE "EMPLOYEE" (
 "ID" BIGINT NOT NULL,
 "VERSION" BIGINT,
 "FIRSTNAME" VARCHAR(20),
 "LASTNAME" VARCHAR(255),
 "AMOUNT" NUMERIC(19, 2),
 "CURRENCY" INTEGER,
 "STARTDATE" DATE,
 "ENDDATE" DATE,
 "ADDRESS_ADDRESS_ID" BIGINT,
 "DTYPE" VARCHAR(255) NOT NULL,
 "MGRFORDEPT_ID" BIGINT
)
```

JPA-providers can create database tables "at startup" in runtime, examples:

- Toplink: `<property name="toplink.ddl-generation" value="create-tables"/>`
- Hibernate: `<property name="hibernate.hbm2ddl.auto" value="update" />`
- Open JPA: `<property name="openjpa.jdbc.SynchronizeMappings" value="buildSchema"/>`

Copyright 2007, Callista Enterprise AB

CALLISTA

## JPA 1.0 – API

- An API for persisting, finding and removing objects

- An `EntityManager` is used to manage persistent objects
  - Handles the session (and caching) where the persistent objects lives when "managed"
- An `EntityManagerFactory` is used for creating `EntityManagers`

```
«Java Interface»
EntityManagerFactory

● createEntityManager ()
● createEntityManager ()
● close ()
● isOpen ()
```

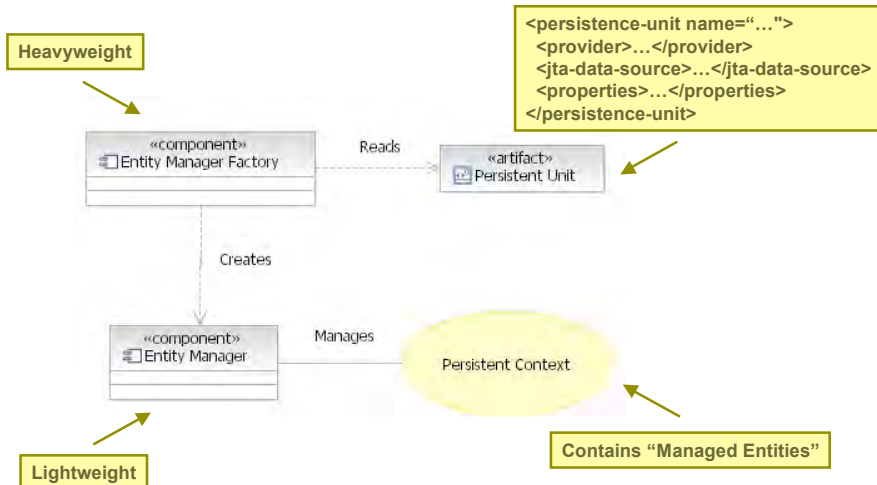
```
«Java Interface»
EntityManager

● persist ()
● merge ()
● remove ()
● find ()
● getReference ()
● flush ()
● setFlushMode ()
● getFlushMode ()
● lock ()
● refresh ()
● clear ()
● contains ()
● createQuery ()
● createNamedQuery ()
● createNativeQuery ()
● createNativeQuery ()
● createNativeQuery ()
● joinTransaction ()
● getDelegate ()
● close ()
● isOpen ()
● getTransaction ()
```

Cadec 2007 - Java Persistence API v1.0, Slide 20  
Copyright 2007, Callista Enterprise AB

CALLISTA

## JPA 1.0 – Components



Cadec 2007 - Java Persistence API v1.0, Slide 21  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 – Types of Entity Managers/ Persistent Contexts

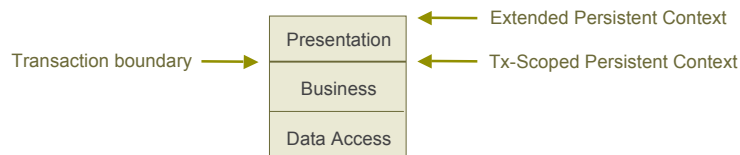
- Application-Managed Persistence Context
  - The application manages the Entity Manager, i.e. is responsible for
    - Creation of Entity Managers
    - Closing the Entity Managers (and its Persistent Context)
    - Transaction demarcation
    - Enlisting Entity Managers with global JTA Transactions, if any
    - Coordinating Entity Managers and Active Transactions
  - Typically used in a Java SE 5.0 environment

Cadec 2007 - Java Persistence API v1.0, Slide 22  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 – Types of Entity Managers/ Persistent Contexts

- Container-Managed Persistence Context
  - All above is handled by the EJB 3.0 Container ☺
  - Comes in two flavors
    - Transaction-Scoped
      - The Persistent Context is closed at transaction demarcation
    - Extended
      - The Persistent Context is open between transactions
      - Requires a statefull session bean
      - Updates queued until next transaction demarcation



Cadec 2007 - Java Persistence API v1.0, Slide 23  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 - Examples

- Usage of JPA with Container Managed Persistence Context

### Ejb 3.0 Session bean

```
@Stateless
public class EmployeeServicesBean implements EmployeeServices {

 private EntityManager m_em = null;

 @PersistenceContext(unitName="Hibernate_Derby_Jta")
 public void setEntityManager(EntityManager em) {
 m_em = em;
 }
}
```

### persistence.xml (JPA Deployment Descriptor)

```
<persistence-unit name="Hibernate_Derby_Jta" transaction-type="JTA">
 <provider>org.hibernate.ejb.HibernatePersistence</provider>
 <jta-data-source>jdbc/Jee5TestDb</jta-data-source>
 <properties>
 <property name="hibernate.dialect" value="org.hibernate.dialect.DerbyDialect" />
 <property name="hibernate.hbm2ddl.auto" value="update" />
 <property name="hibernate.show_sql" value="true" />
 </properties>
</persistence-unit>
```

Pluggable JPA

Cadec 2007 - Java Persistence API v1.0, Slide 24  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 - Examples

- Create a Employee and persist it

### Ejb 3.0 Session bean

```
public Employee createEmployee(String firstName, String lastName, int salary...) {
 Employee e = new Employee();

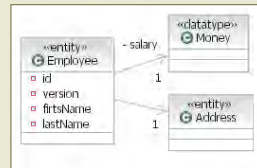
 // Init the emp-object itself
 e.setFirstName(firstName);
 e.setLastName(lastName);

 // Create an salary-object and update the emp
 e.setSalary(new Money(new BigDecimal(salary), SEK));

 // Create an address-object and update the emp
 Address a = new Address();
 a.setCity(city);
 ...
 e.setAddress(a);

 // We are done with the object graph, let's persist it...
 m_em.persist(e);

 return e;
}
```



Cadec 2007 - Java Persistence API v1.0, Slide 25  
Copyright 2007, Callista Enterprise AB

CALLISTA

## JPA 1.0 - Examples

- Find Employees using Queries

### Ejb 3.0 Session bean

```
public List<Employee> findEmployees(int start, int max, String lastName) {
 return m_em.createQuery(
 "SELECT e FROM Employee e WHERE e.lastName LIKE :lastName
 ORDER BY e.lastName, e.firstName")
 .setParameter("lastName", lastName + "%")
 .setFirstResult(start)
 .setMaxResults(max)
 .getResultList();
}
```

### Caller

```
List<Employee> empList = employeeServices.findEmployees(0, 10, "A");

for (Employee employee : empList) {
 System.out.println(
 employee.getId() + " : " +
 employee.getSalary().getAmount() + " " +
 employee.getSalary().getCurrency());
}
```

Cadec 2007 - Java Persistence API v1.0, Slide 26  
Copyright 2007, Callista Enterprise AB

CALLISTA

## JPA 1.0 - Examples

- Find Employees using Named Queries

### JPA 1.0 ORM.XML

```
<named-query name = "findEmp">
 <query>
 SELECT e FROM Employee e WHERE e.lastName LIKE :lname
 ORDER BY e.lastName, e.firstName
 </query>
</named-query>
```

### Ejb 3.0 Session bean

```
public List<Employee> findEmployees(int start, int max, String lastName) {
 return m_em.createNamedQuery("findEmp")
 .setParameter("lastName", lastName + "%")
 .setFirstResult(start)
 .setMaxResults(max)
 .getResultList();
}
```

Cadec 2007 - Java Persistence API v1.0, Slide 27  
Copyright 2007, Callista Enterprise AB

CALLISTA

## Where are we?

- Bored of writing Data Access Objects?
- Terminology
  - ORM - Object Relational Mapping frameworks
  - POJO based persistence
- Introducing JPA...
  - What is it?
  - Examples!
- Details & Guidelines
  - Until time runs out...
- Summary

Cadec 2007 - Java Persistence API v1.0, Slide 28  
Copyright 2007, Callista Enterprise AB

CALLISTA

## JPA 1.0 - Details & Guidelines

- Managed and Detached entities
- Guidelines for Persistent Contexts
- Lazy and Eager loading
- Constructor expressions in queries
- Cascading operations
- Bulk Update and Delete
- Validations

Cadec 2007 - Java Persistence API v1.0, Slide 29  
Copyright 2007, Callista Enterprise AB



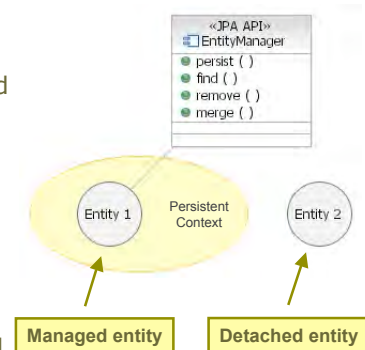
## JPA 1.0 – Managed and Detached entities

- *Managed Entities*
  - Updates automatically saved
  - Related objects automatically loaded (if required)
    - E.g.:

```
emp.getPhoneNumbers()
```

results in

```
SELECT ... FROM PHONE_NUMBER...
```
- *Detached Entities*
  - Managed entities becomes detached when the Entity Manager is closed
  - Becomes managed again after using the *merge*-operation



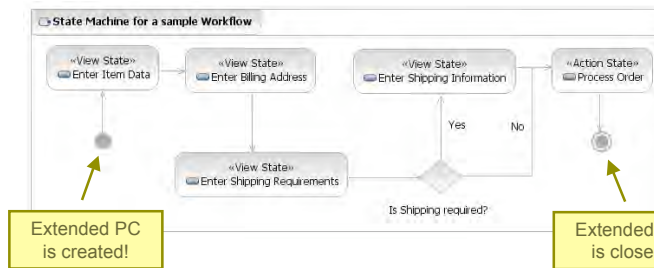
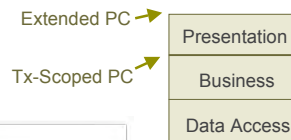
Cadec 2007 - Java Persistence API v1.0, Slide 30  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 – Guidelines for Persistent Contexts (PC)

- Use Container-Managed PC whenever possible

- Use Transaction-Scoped PC by default
- Consider using Extended PC for
  - Well defined Workflows



- **Note:** See later on how JBoss Seam simplifies use of Extended PC!

Cadec 2007 - Java Persistence API v1.0, Slide 31  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 – Lazy and Eager loading

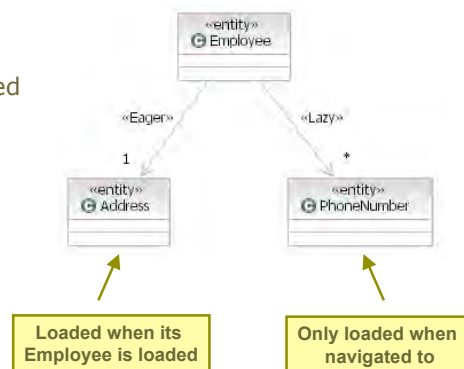
- Controlling when related entities are loaded

- *Eager*

Load related entities when the "parent" entity is loaded

- *Lazy*

Load related entities if they are navigated to



Cadec 2007 - Java Persistence API v1.0, Slide 32  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 - Lazy and Eager loading

- Lazy

```
@OneToMany(cascade=ALL, mappedBy = "owner", fetch=LAZY)
private List<PhoneNumber> phoneNumbers;
```

- Potential performance problem: Many small SQL statements

- Eager

```
@OneToMany(cascade=ALL, mappedBy = "owner", fetch=EAGER)
private List<PhoneNumber> phoneNumbers;
```

- Potential performance problem: Too large SQL statements

- Guideline

- Use lazy loading on relations and "fetch join" on queries

```
SELECT e FROM Employee e LEFT JOIN FETCH e.phoneNumbers WHERE ...
```

Cadec 2007 - Java Persistence API v1.0, Slide 33  
Copyright 2007, Callista Enterprise AB



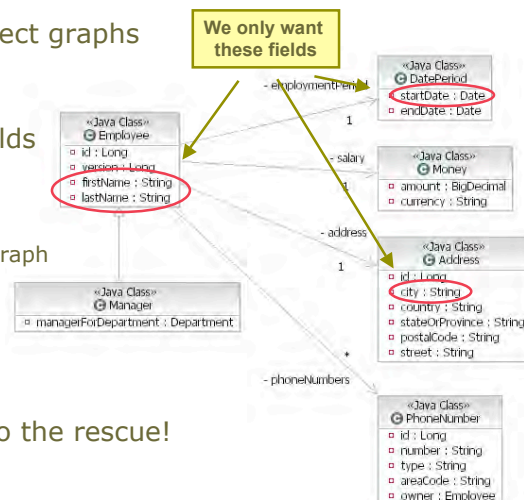
## JPA 1.0 - Constructor Expressions in queries

- Normal queries return object graphs

- Not suitable for queries that only require some fields in a complex object graph

- Don't want to
  - traverse the full object graph to get the fields
  - populate object graph with unused fields

- *Constructor Expressions* to the rescue!



Cadec 2007 - Java Persistence API v1.0, Slide 34  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 - Constructor Expressions in queries

- Example

POJO non - Entity bean

```
public class EmployeeReportData {

 private String firstName;
 private String lastName;
 private String city;
 private Date startDate;

 public EmployeeReportData
 (String firstName, String lastName, String city, Date startDate) {

 this.firstName = firstName;
 this.lastName = lastName;
 this.city = city;
 this.startDate = startDate;
 }

 // Setters and Getters...
}
```

Cadec 2007 - Java Persistence API v1.0, Slide 35  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 - Constructor Expressions in queries

- Example

Ejb 3.0 Session bean

```
public List<EmployeeReportData> findEmployeeReportData
(int startIndex, int maxResults, String lastName) {

 return m_em.createQuery(
 "SELECT new se.callista.jpa.dalitest.services.api.EmployeeReportData " +
 " (e.firstName, e.lastName, a.city, e.employmentPeriod.startDate) " +
 " FROM Employee e JOIN e.address a " +
 " WHERE e.lastName LIKE :lastName ORDER BY e.lastName, e.firstName")
 .setParameter("lastName", lastName + "%")
 .setFirstResult(startIndex)
 .setMaxResults(maxResults)
 .getResultList();
}
```

Caller

```
for (EmployeeReportData row : employeeServices.findEmployeeReportData(0, 10, "")) {
 System.out.println(row.getFirstName() + ", " + row.getLastName() + ", " +
 row.getCity() + " " + row.getStartDate());
}
```

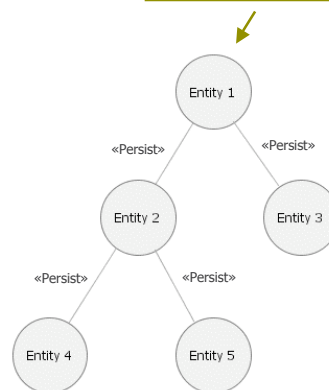
Cadec 2007 - Java Persistence API v1.0, Slide 36  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 – Cascaded operations

- Applicable for Entity Manager operations
  - Persist
  - Merge
  - Remove
  - Refresh
- Defined per relationship

All new entities will be persisted when the entity manager persist the top level entity



Cadec 2007 - Java Persistence API v1.0, Slide 37  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 - Cascaded operations

- Declaration

```
@OneToOne(cascade=ALL)
private Address address;
```
- Very powerful but obviously very dangerous
  - Persist, Merge and Remove operation can be propagated far beyond your expectations...
- Guideline
  - Do not overuse cascading operations
  - Target well defined composites

Cadec 2007 - Java Persistence API v1.0, Slide 38  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 - Bulk Update and Delete

- Execute Updates and Delete directly against the database
- We can still use the abstractions in the object model!
- **Warning:** The Persistent Context is not updated!!!
  - Usage
    - In separate transactions
    - As the first operation using Transaction-Scoped Persistence Contexts

```
@TransactionAttribute(REQUIRES_NEW)
public void assignManager(Department dept, Employee manager) {

 em.createQuery(
 "UPDATE Employee e SET e.manager = ?1 WHERE e.department = ?2 ")
 .setParameter(1, manager)
 .setParameter(2, dept)
 .executeUpdate();
}
```

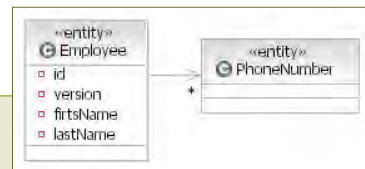
Cadec 2007 - Java Persistence API v1.0, Slide 39  
Copyright 2007, Callista Enterprise AB



## JPA 1.0 - Validations

- Validation can be performed by using Lifecycle Callback Methods
  - Lifecycle Callback Methods
    - PrePersist, PostPersist
    - PreUpdate, PostUpdate
    - PreRemove, PostRemove
  - Throwing an exception will mark the current transaction for rollback

```
@Entity public class Employee {
 @PreUpdate
 @PrePersist
 @PostPersist
 public void validate() {
 if (getPhoneNumbers() == null || getPhoneNumbers().size() == 0)
 throw new ValidationException(INVALID_EMP_PHONE_MISSING);
 }
}
```



Cadec 2007 - Java Persistence API v1.0, Slide 40  
Copyright 2007, Callista Enterprise AB



## Summary

---

- **Start using JPA today!**

- The JPA standard is based on well proven technology
- Huge vendor support
  - Both from Open Source and from Product Vendors
- Do not start new projects using ORM framework proprietary API's
  - JPA support for vendor specific features can be used if really needed
- Secure skills if you are new to ORM frameworks
  - The road from DAO/DTO J2EE Patterns to JPA have some pitfalls...

Cadec 2007 - Java Persistence API v1.0, Slide 41  
Copyright 2007, Callista Enterprise AB



## Questions?

---



Cadec 2007 - Java Persistence API v1.0, Slide 42  
Copyright 2007, Callista Enterprise AB

