



Persistence - Part 4

Philippe Collet, contains 78,3% of slides from
Sébastien Mosser



Advanced concepts

& tricks...

Stop!

[https://github.com/collet/4A_ISA_TheCookieFactory/
blob/develop/chapters/Persistence.md](https://github.com/collet/4A_ISA_TheCookieFactory/blob/develop/chapters/Persistence.md)

First

Set up: Persistence Unit (general)

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
    http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">

  <persistence-unit name="tcf_persistence_unit" transaction-type="JTA">
    <jta-data-source>TCFDataSource</jta-data-source>

    <class>fr.unice.polytech.isa.tcf.entities.Customer</class>
    <class>fr.unice.polytech.isa.tcf.entities.Item</class>
    <class>fr.unice.polytech.isa.tcf.entities.Order</class>

    <exclude-unlisted-classes>true</exclude-unlisted-classes>
    <properties>
      <property name="openjpa.jdbc.SynchronizeMappings" value="buildSchema(ForeignKeys=true)"/>
    </properties>
  </persistence-unit>

</persistence>
```



DataSource

Setup: Prod \neq Test

```
<resources>
  <Resource id="production" type="DataSource">
    JdbcDriver    org.hsqldb.jdbcDriver
    JdbcUrl       jdbc:hsqldb:file:proddb
    UserName      sa
    Password
    LogSql        true
    JtaManaged   true
  </Resource>
</resources>
```

WEB-INF/resources.xml

```
<property name="properties">
  my-datasource = new://Resource?type=DataSource
  my-datasource.JdbcUrl = jdbc:hsqldb:mem:TCFDB;shutdown=true
  my-datasource.UserName = sa
  my-datasource.Password =
  my-datasource.JtaManaged = true
  my-datasource.LogSql = true
</property>
```

arquillian.xml

Set up: Bytecode enhancement (OpenJPA specific)

```
<plugin>
  <groupId>org.apache.openjpa</groupId>
  <artifactId>openjpa-maven-plugin</artifactId>
  <version>2.4.1</version>
  <configuration>
    <includes>**/entities/*.class</includes>
    <addDefaultConstructor>true</addDefaultConstructor>
    <enforcePropertyRestrictions>true</enforcePropertyRestrictions>
  </configuration>
  <executions>
    <execution>
      <id>enhancer</id>
      <phase>process-classes</phase>
      <goals>
        <goal>enhance</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Dedicated Java agent

Annotations: Structural constraints / Validation

```
@Entity
public class Customer implements Serializable {

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

    @NotNull
    private String name;

    @NotNull
    @Pattern(regexp = "\\d{10}+", message = "Invalid creditCardNumber")
    private String creditCard;

    @OneToMany(mappedBy = "customer")
    private Set<Order> orders = new HashSet<>();

    // ...
}
```

Classical querying

```
int id = 42;  
Customer c = (Customer) entityManager.find(Customer.class, id);
```

```
entityManager.createQuery("DELETE FROM Customer").executeUpdate();
```

Issues?

EQL: EJB Query Language

```
@Override
public Optional<Customer> findByName(String name) {
    CriteriaBuilder builder = entityManager.getCriteriaBuilder();

    CriteriaQuery<Customer> criteria = builder.createQuery(Customer.class);
    Root<Customer> root = criteria.from(Customer.class);

    criteria.select(root).where(builder.equal(root.get("name"), name));
    TypedQuery<Customer> query = entityManager.createQuery(criteria);

    try {
        return Optional.of(query.getSingleResult());
    } catch (NoResultException nre){
        return Optional.empty();
    }
}
```

Query Typing

Cascading

There is a containment relationship between Customers and Orders.
Deleting a Customer should delete the associated Orders transitively

```
public class Customer {  
  
    // ...  
  
    @OneToMany(cascade = {CascadeType.REMOVE}, mappedBy = "customer")  
    private Set<Order> orders = new HashSet<>();  
  
    // ...  
}
```

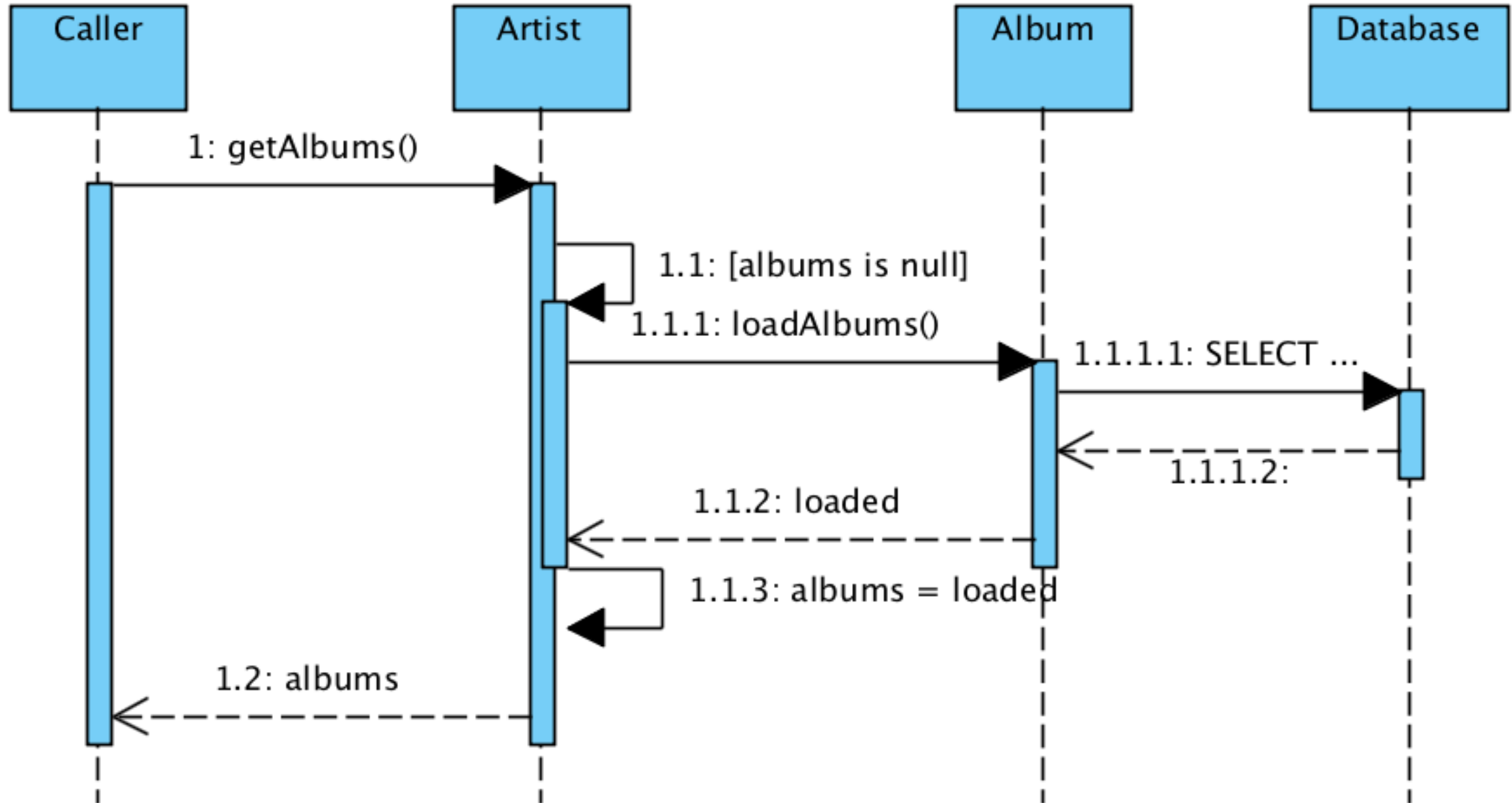
Warning: be very careful when cascading operations, especially the REMOVE one.
Deleting an Order should not delete the associated customer...

JPA Cascading Types

- **CascadeType.PERSIST** : cascade type persist means that save() or persist() operations cascade to related entities.
- **CascadeType.MERGE** : cascade type merge means that related entities are merged when the owning entity is merged.
- **CascadeType.REFRESH** : cascade type refresh does the same thing for the refresh() operation.
- **CascadeType.REMOVE** : cascade type remove removes all related entities association with this setting when the owning entity is deleted.
- **CascadeType.DETACH** : cascade type detach detaches all related entities if a “manual detach” occurs.
- **CascadeType.ALL** : cascade type all is shorthand for all of the above cascade operations.

There is no default cascade type in JPA. By default no operations are cascaded.

Lazy Loading



Default Lazy loading behavior

Relationship type	Default fetch behavior	Number of entities retrieved
One-to-one	EAGER	Single entity retrieved
One-to-many	LAZY	Collection of entities retrieved
Many-to-one	EAGER	Single entity retrieved
Many-to-many	LAZY	Collection of entities retrieved

Change it by « fetch »

```
@OneToMany(cascade = {CascadeType.REMOVE}, fetch = FetchType.LAZY, mappedBy = "customer")  
private Set<Order> orders = new HashSet<>();
```

Lazy loading of the the orders associated to a given customer

https://github.com/collet/4A_ISA_TheCookieFactory/blob/release/v2.1/j2e/src/test/java/fr/unice/polytech/isa/tcf/persistence/LazyLoadingTest.java

```
@OneToMany(cascade = {CascadeType.REMOVE}, fetch = FetchType.LAZY, mappedBy = "customer")
private Set<Order> orders = new HashSet<>();
```

```
@Test
public void lazyLoadingDemo() throws Exception {
    // Code executed inside a given transaction
    manual.begin();
    Customer john = new Customer("John Doe", "1234567890");
    entityManager.persist(john);
    Order o1 = new Order(john, Cookies.CHOCOLALALA, 3); entityManager.persist(o1); john.add(o1);
    Order o2 = new Order(john, Cookies.DARK_TEMPTATION, 1); entityManager.persist(o2); john.add(o2);
    Order o3 = new Order(john, Cookies.SOO_CHOCOLATE, 2); entityManager.persist(o3); john.add(o3);
    Customer sameTransaction = loadCustomer(john.getId());
    assertEquals(john, sameTransaction);
    assertEquals(3, john.getOrders().size()); // orders are attached in this transaction => available
    manual.commit();

    // Code executed outside the given transaction
    Customer detached = loadCustomer(john.getId());
    assertEquals(john, detached);
    assertNull(detached.getOrders()); // orders are not attached outside of the transaction => null;
}
```

OUTSIDE THE TRANSACTION (DETACHMENT...)

```
private Customer loadCustomer(int id) {
    return entityManager.find(Customer.class, id);
}
```

A single module for all entities, just like TCF

