

JPA with Hibernate 3.0

Lesson 02: The Persistence Life
Cycle

Lesson Objectives

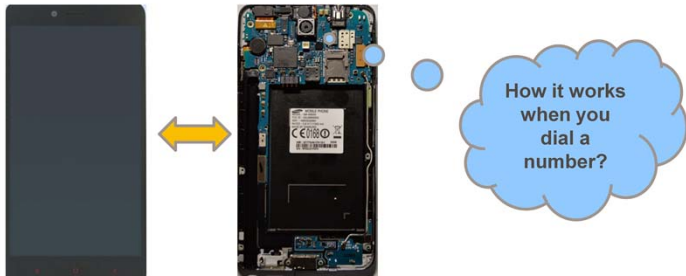
- After completing this lesson, participants will be able to understand:
 - What is persistence?
 - Persistence Life Cycle



2.1: Introduction

How ORM works?

- ORM provides high-level of abstraction



The diagram shows a black mobile phone on the left and its internal components (circuitry, battery, etc.) on the right. A yellow double-headed arrow connects them, representing the abstraction layer. A blue thought bubble next to the internal components contains the text: "How it works when you dial a number?".

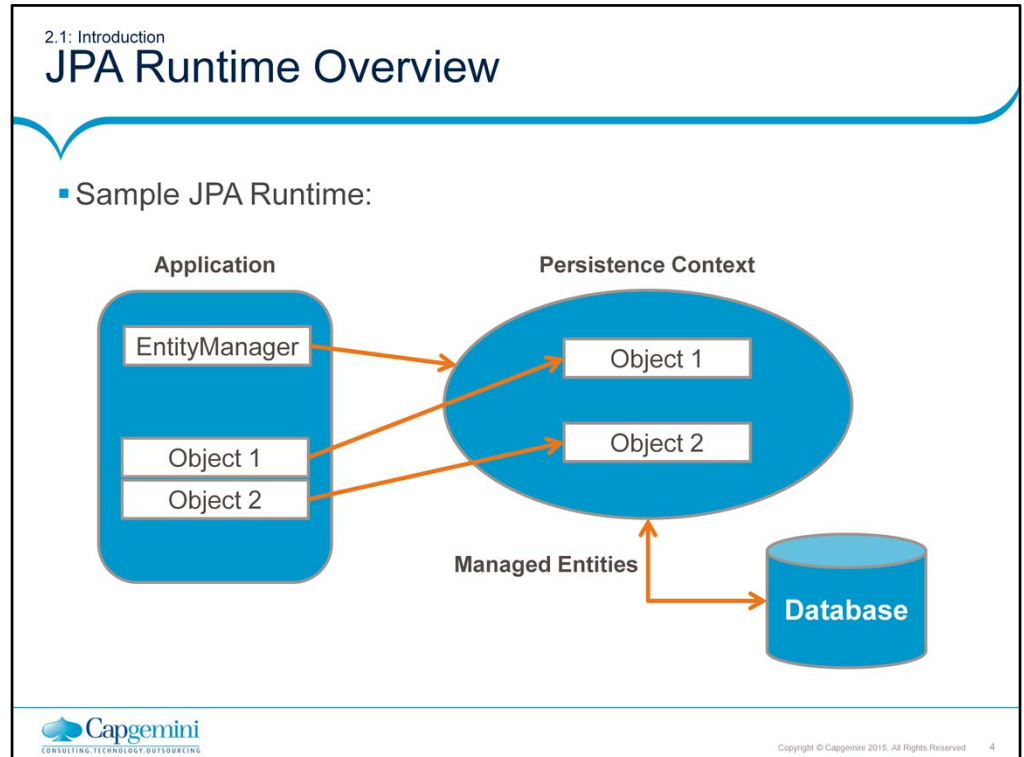
Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 3

Before we start working with ORM, it is very important to understand how ORM works. The slide shows an example of abstraction, when we dial or receive a call on mobile, lot of functionality goes in background. We as a user, least bothered about internal component working due to abstraction.

Similarly, objects created in your application, when passed to ORM, get stored in database table. How it happens? What work goes in background? How your object persisted in database?

This lesson tries to answer abstract working of ORM and we will discuss in details about the object's persistence life cycle.

**Entity Manager:**

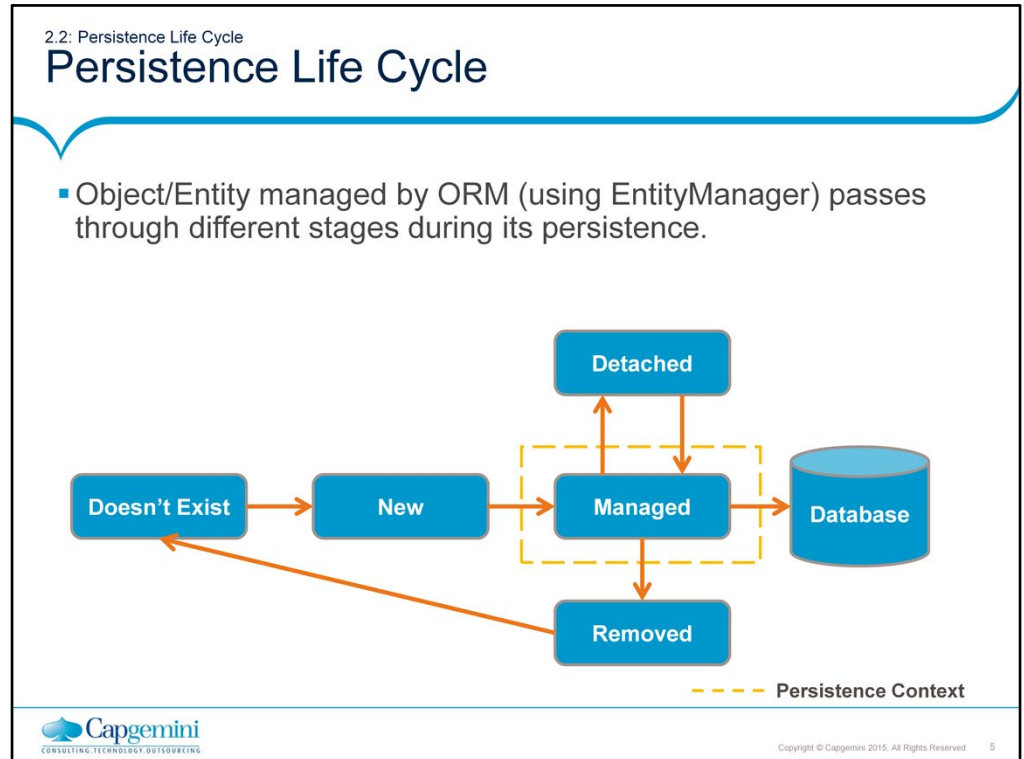
The EntityManager is the primary interface used by application developers to interact with the JPA runtime.

Persistence Context:

Persistence context defines a scope under which particular entity instances are created, persisted, and removed.

Every EntityManager manages **its own** persistence context. In short, persistence context is a memory area for EntityManager to work on entity instance.

The slide diagram shows a sample JPA runtime. JPA uses EntityManager instance to manage objects which required to be persisted. Such objects are called **Entities**. Entities managed by EntityManager travels through different life cycle phases. This lesson primarily focuses on entities persistence life cycle.



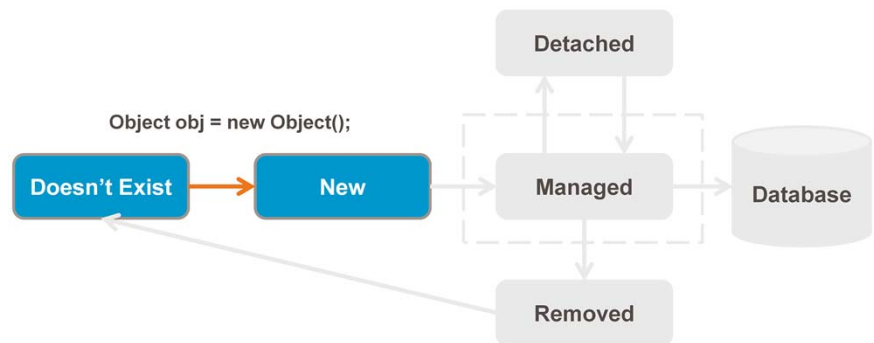
The slide diagram shows different states of entity. When entity is in managed state, the data of entity is persisted in database.

The remaining slides discuss about entity states in details.

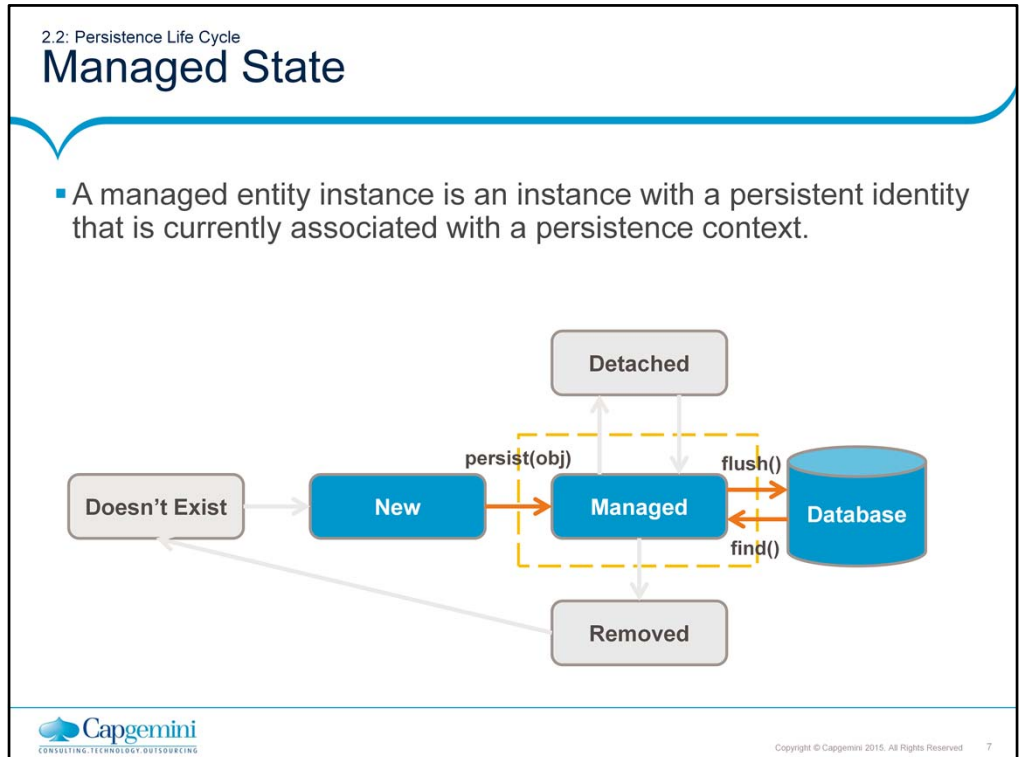
2.2: Persistence Life Cycle

New / Transient State

- An entity is new if it has just been instantiated using the new operator, and it is not associated with a persistence context. It has no persistent representation in the database and no identifier value has been assigned.



New State :When an entity object is initially created its state is **New**. In this state the object is not yet associated with an Entity Manager and has no representation in the database.

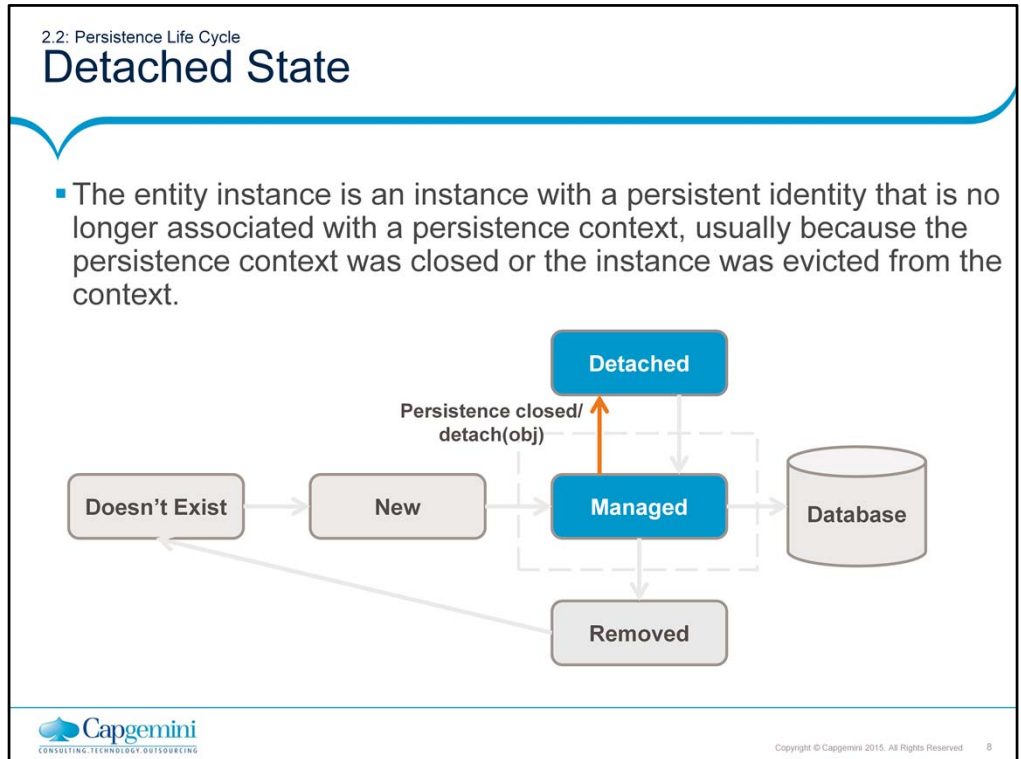


Managed State : An entity object becomes **Managed** when it is persisted to the database via an EntityManager's `persist` method which must be invoked within an active transaction. On transaction commit, the owning Entity Manager stores the new entity object to the database.

Entity objects retrieved from the database by an EntityManager are also in the **Managed** state.

If a managed entity object is modified within an active transaction the change is detected by the owning EntityManager and the update is propagated to the database on transaction commit.

Note: The methods shown in slide are of EntityManager interface.



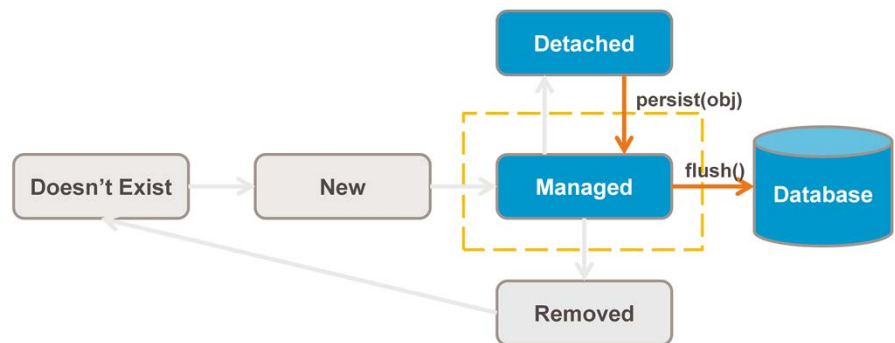
Detached State : represents entity objects that have been disconnected from the EntityManager. For instance, all the managed objects of an EntityManager become detached when the EntityManager is closed.

Note: The method shown in slide are of EntityManager interface.

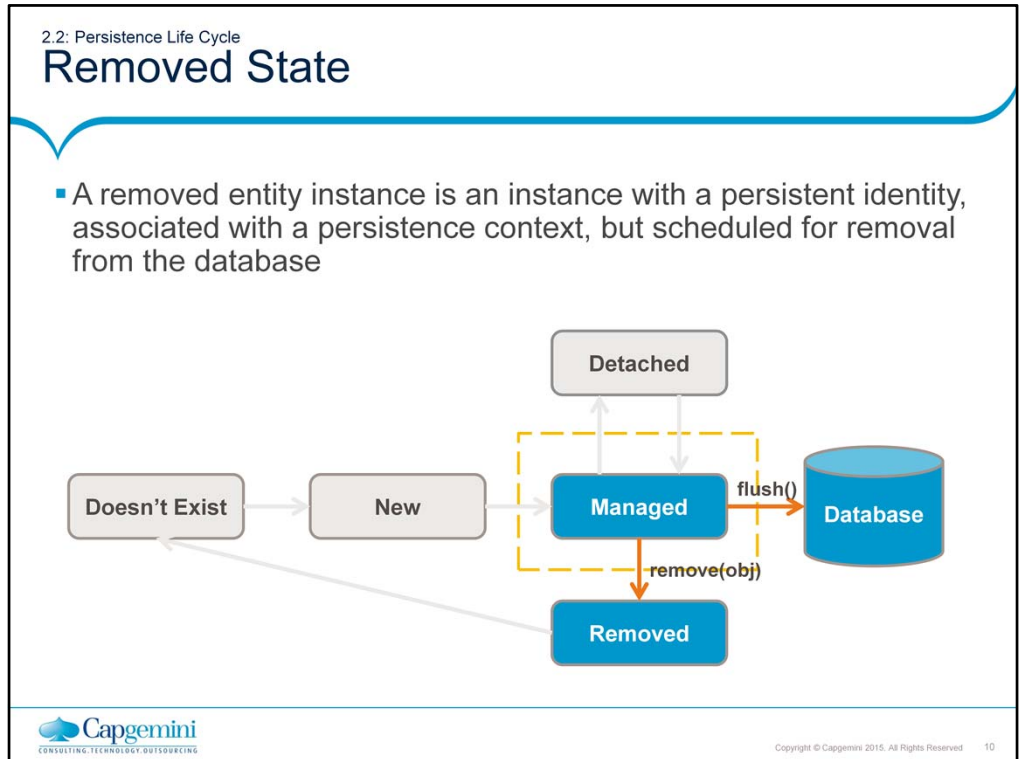
2.2: Persistence Life Cycle

Detached State

- A detached entity instance can be restored into managed state, when you persist it again. The ORM ensures state of entity instance is in sync with database.



Note: The methods shown in slide are of EntityManager interface.



Removed State : A managed entity object can also be retrieved from the database and marked for deletion, by using the EntityManager's remove method within an active transaction. The entity object changes its state from Managed to **Removed**, and is physically deleted from the database during commit.

Note: The methods shown in slide are of EntityManager interface.

Summary

- In this lesson, you have learned about:
 - Persistence Life Cycle



Review Question

- Question 1: When we persist entity it becomes managed.
 - True
 - False
- Question 2: When an entity instance is with a persistent identity and currently associated with a persistence context, then it is ____ state.
 - New
 - Managed
 - Removed

