Partner ORACLE UNIVERSITY

Certifica TIC Beyond Certification

INICIO

Desarrollo de Aplicaciones Empresariales con Spring Framework Core 4

ISC. Ivan Venor García Baños

ATC Advanced Training Consultants

SCRUMstudy Authorized Training Partner

MEXICO FIRST

SE SECRETARÍA DE ECONOMÍA

CANIETI www.canieti.org

Agenda

1. Presentación
2. Objetivos
3. Contenido
4. Despedida

3. Contenido

i. Introducción a Spring Framework
ii. Spring Core
iii. Spring AOP
iv. Spring JDBC – Transaction
v. Spring ORM – Hibernate 4
vi. Fundamentos Spring MVC y Spring REST
vii. Fundamentos Spring Security

v. Spring ORM – Hibernate 4

v. Spring ORM – Hibernate 4 (a)

v.i Introducción
- a. ¿Qué es ORM?
- b. ¿Qué es Spring ORM?

v.ii Hibernate 4
- a. Sesiones
- b. Ciclo de vida objetos persistentes
- c. Transacciones
- d. Mapeo de Entidades con Anotaciones

v. Spring ORM – Hibernate 4 (b)

v.iii Integración Spring ORM – Hibernate 4

a. Configuración DataSource

b. Configuración TransactionManager

c. Configuración SessionFactory

d. Implementación Hibernate DAO

Práctica 27. Configuración capa DAO Spring ORM - Hibernate 4

v.i Introducción

Objetivos de la lección

v.i Introducción

- Revisar qué es un ORM.
- Conocer a grandes rasgos cuales son los frameworks de persistencia Java más populares.
- Comprender el beneficio de utilizar Spring ORM.

v.i Introducción

a. ¿Qué es ORM?

b. ¿Qué es Spring ORM?

v.i Introducción (a)

- ¿Qué es ORM?

- ORM (Object Relational Mapping) es un técnica de programación que convierte objetos entre sistemas incompatibles, tal como lo son las bases de datos y los lenguajes de programación.

- Consiste en la transformación de:
  - Tablas de una base de datos a clases Java (POJOs) y,
  - Columnas de tablas a propiedades de clases Java

v.i Introducción (b)

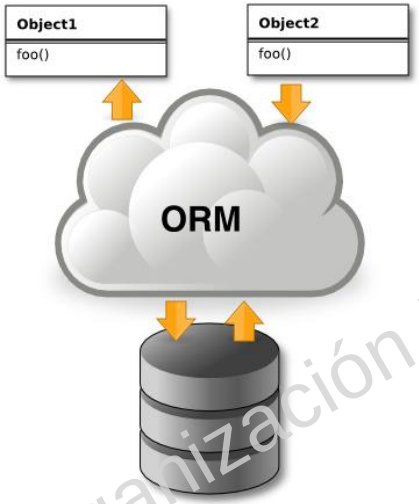
- ¿Qué es ORM?

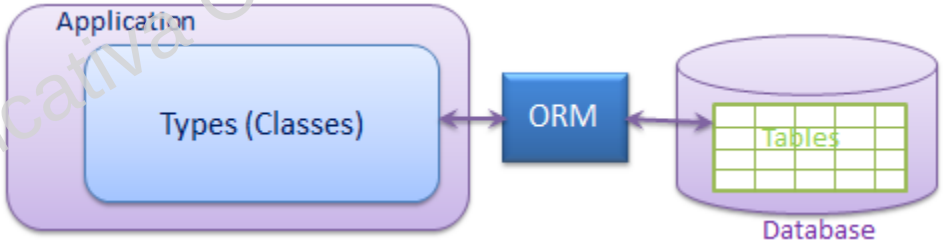
- ORM es una técnica utilizada para convertir datos y tipos de un lenguaje de programación orientado a objetos, como Java, a un modelo de datos ER distribuido en una base de datos relacional.

- Un framework ORM es la herramienta que implementa la técnica del Mapeo Objecto-Relacional (ORM).

v.i Introducción (c)

- ¿Qué es ORM?





v.i Introducción (d)

- ¿Qué es ORM?

- Frameworks ORM:
  - Hibernate
  - JPA
  - MyBatis
  - JDO
  - EclipseLink
  - TopLink
  - entre otros

v.i Introducción

a. ¿Qué es ORM?

b. ¿Qué es Spring ORM?

v.i Introducción (a)

- ¿Qué es Spring ORM?

- Spring ORM, al igual que el *DAOSupport* que provee Spring JDBC, es una capa de abstracción que permite la integración de Spring con los frameworks de persistencia más populares tales como Hibernate, JPA, JDO o MyBatis.

- Spring ofrece clases de integración para cualquiera de los frameworks de persistencia soportados y, ésta integración, es de manera natural siguiendo los principios de configuración de beans de Spring.

v.i Introducción (b)

- ¿Qué es Spring ORM?

- En otras palabras Spring ORM es una capa adaptadora (*adapter)* entre el contenedor de IoC de Spring y cualquier framework de persistencia soportado.

v.i Introducción (c)

- ¿Qué es Spring ORM?

- Ventajas de utilizar Spring ORM
  - Menor cantidad código de configuración para el framework ORM
  - Facilidad para probar implementaciones DAO con Frameworks ORM
  - Manejo de excepciones *(DAOSupport)*
  - Manejo de Recursos *(DataSource, SessionFactory, EntityManager)*
  - Integración natural con manejo transaccional *(TransactionManager)*

Resumen de la lección

v.i Introducción

- Comprendimos lo que es un ORM.
- Conocimos a grandes rasgos cuales son los principales frameworks Java de persistencia.
- Comprendimos que es el módulo Spring ORM.

Esta página fue intencionalmente dejada en blanco.

v.ii Hibernate 4

Objetivos de la lección

v.ii Hibernate 4

- Revisar los principales objetos que intervienen en el funcionamiento de Hibernate como framework de persistencia.
- Comprender la utilidad del objeto Session.
- Conocer los diferentes estados de los objetos persistentes que maneja Hibernate.
- Conocer como se utiliza, a grandes rasgos, el API Transaccional de Hibernate.
- Verificar como se realiza el mapeo objeto-relacional entre clases (Entidades) a tablas en la base de datos.

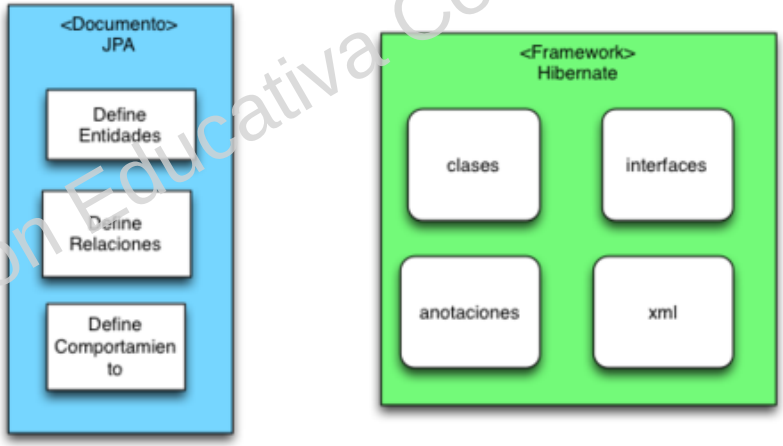
v.ii Hibernate 4

a. Sesiones
b. Ciclo de vida objetos persistentes
c. Transacciones
d. Mapeo de Entidades con Anotaciones

v.ii Hibernate 4 (a)

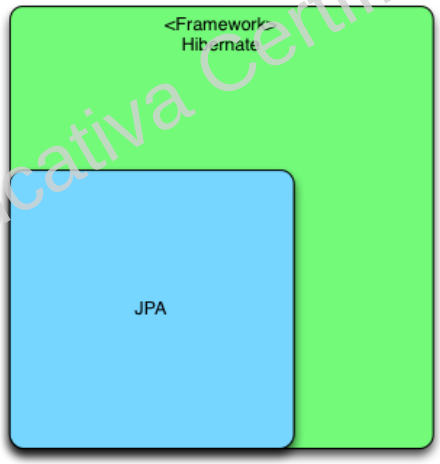
- Hibernate es un framework de persistencia ORM de alto rendimiento.

- Hibernate es una implementación JPA, aunque Hibernate existió antes que JPA.



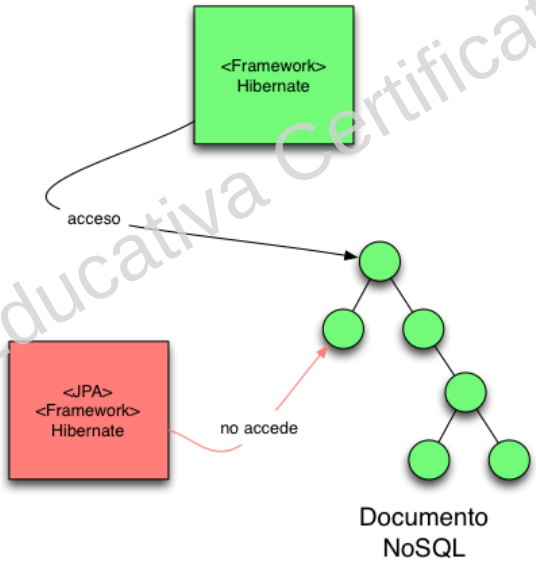
v.ii Hibernate 4 (b)

- Hibernate es a implementación más popular de JPA debido a que ofrece mucha funcionalidad que no se especifica en el estándar JPA.



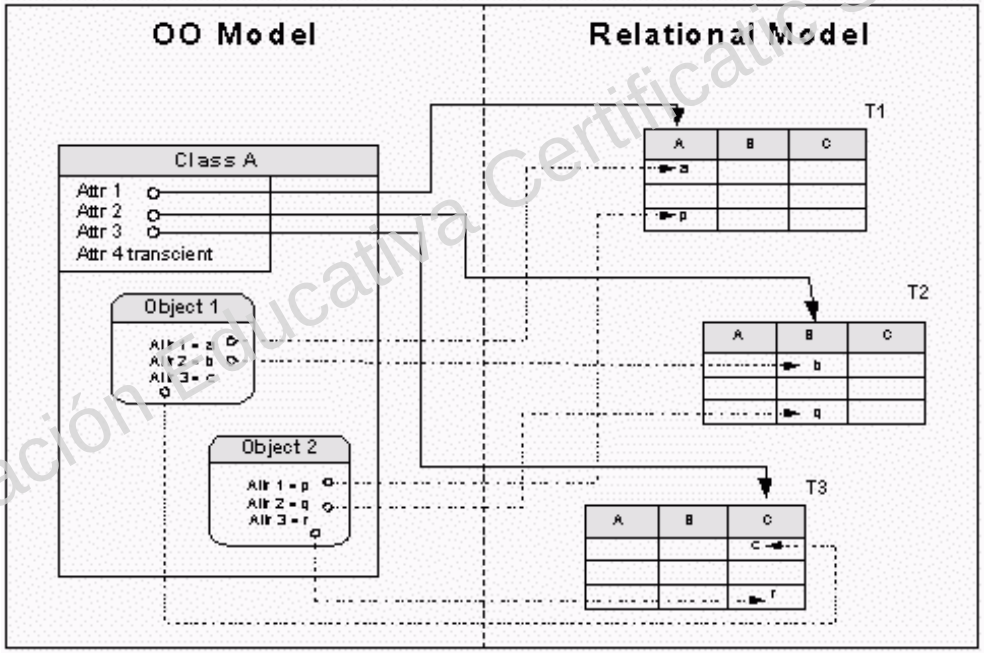
v.ii Hibernate 4 (c)

- Hibernate soporta trabajar con bases de datos NoSQL y JPA no.



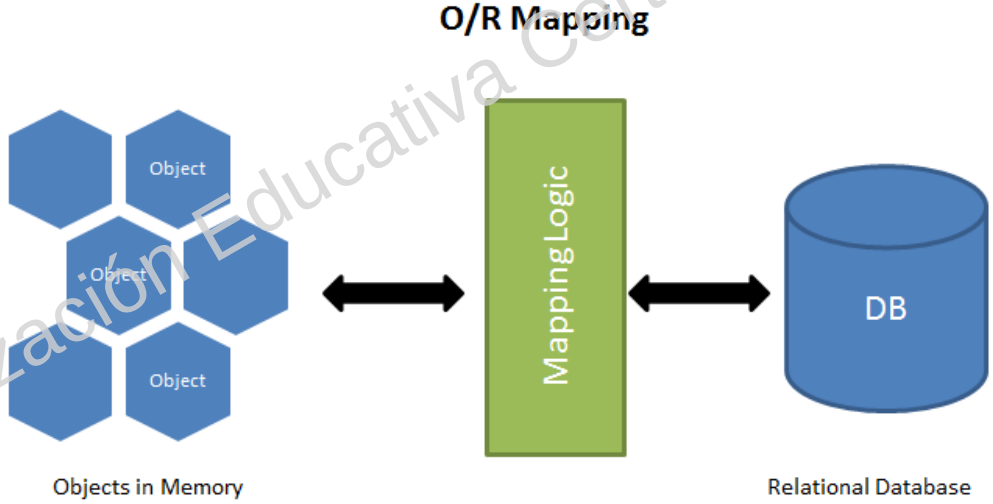
v.ii Hibernate 4 (d)

- Hibernate posibilita trabajar con una base de datos orientada a objetos virtual.



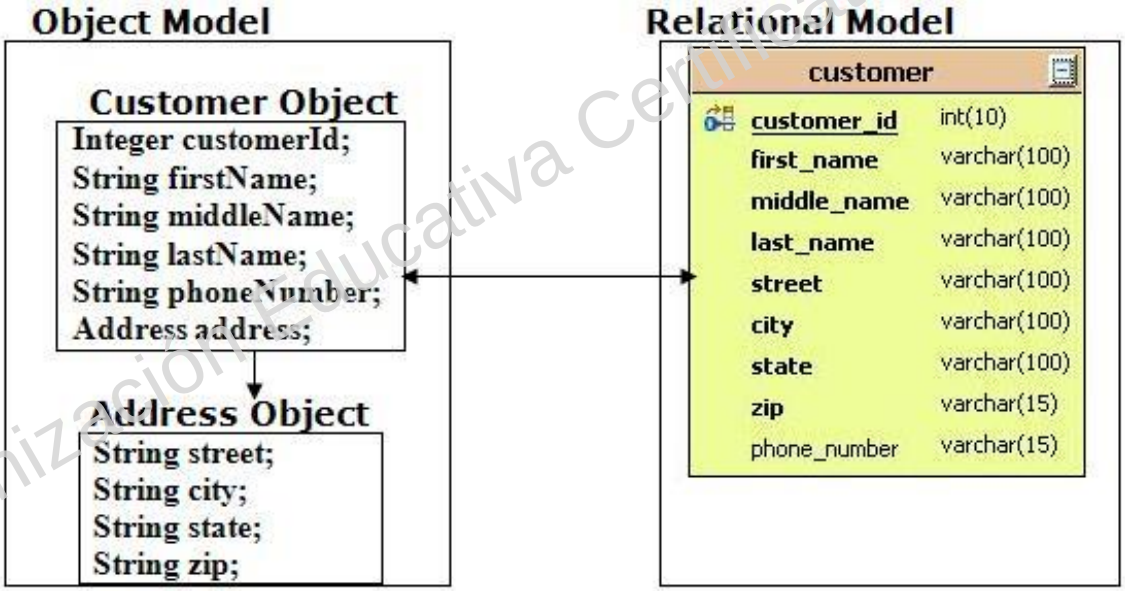
v.ii Hibernate 4 (e)

- Hibernate mapea clases Java (POJO) a Tablas de base de datos y Tipos Java a Tipos SQL y viceversa, mediante configuración por medio de XML o @Anotaciones (propietarias de Hibenate o estándar JPA).



v.ii Hibernate 4 (f)

- Hibernate permite mapear componentes que permite mapear dos o más clases a una misma tabla.



v.ii Hibernate 4 (g)

- Ventajas (a)
  - Mapeo ORM mediante configuración por XML (no invasivo).
  - Soporte de @Anotaciones estándar JPA.
  - API simple para ejecutar operaciones CRUD sobre base de datos mediante la utilización de POJOs (Entidades).
  - Facilidad de mantenimiento a Entidades (POJO).
  - No requiere utilizar consultas SQL.

v.ii Hibernate 4 (h)

- Ventajas (b)
  - Mapeo automático entre tipos Java y SQL y, viceversa.
  - Ampliamente utilizado por la comunidad Java.
  - No requiere *application server.*
  - Manipula asociaciones entre complejas entre Entidades (tablas) y minimiza el acceso físico a la base de datos.

- Desventajas
  - Lentitud manejando altos volúmenes de datos.
  - Curva de aprendizaje compleja.

v.ii Hibernate 4 (i)

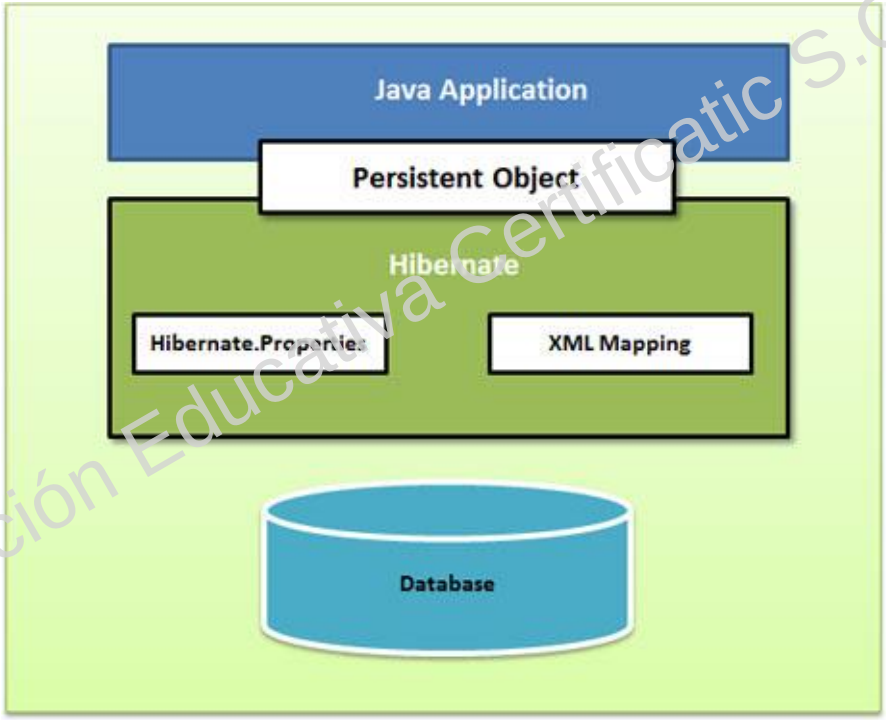
- Soporte a proveedores de bases de datos.

- Hibernate soporta la mayoría de Manejadores de Bases de Datos.

  - HSQL Database Engine
  - DB2/NT
  - MySQL
  - PostgreSQL
  - FrontBase
  - Oracle
  - Microsoft SQL Server Database
  - Sybase SQL Server
  - Informix Dynamic Server

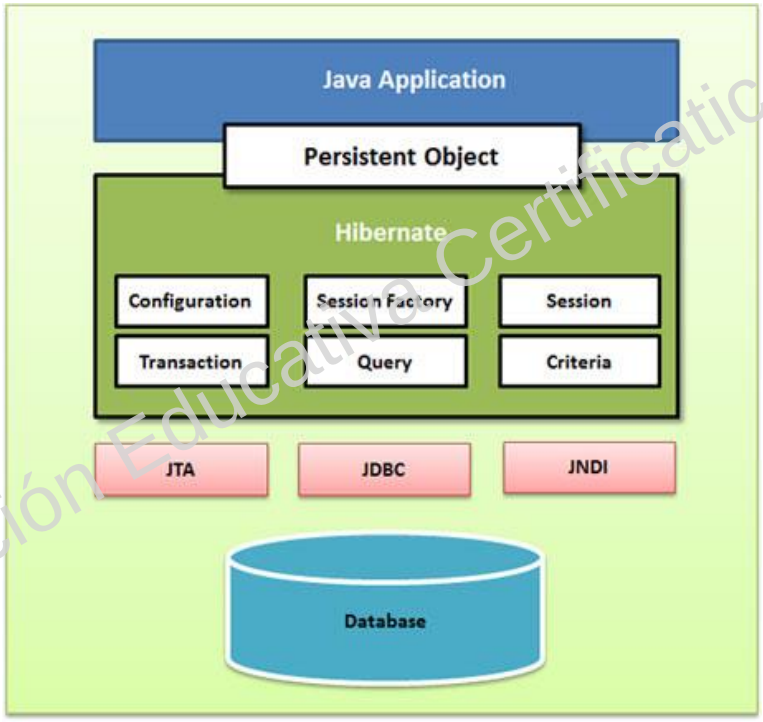
v.ii Hibernate 4 (j)

- Arquitectura



v.ii Hibernate 4 (k)

- Arquitectura



v.ii Hibernate 4

a. Sesiones
b. Ciclo de vida Objetos Persistentes
c. Transacciones
d. Mapeo de Entidades con Anotaciones

v.ii Hibernate 4 (a)

- Sesiones

- Una sesión en Hibernate *(Session)* es un conexión virtual con la base de datos y se utiliza para interactuar con la base de datos.

- La *Session* está diseñada para ser instanciada cada vez que sea necesario interacción con base de datos.

- Se recomienda liberar el recurso *Session* lo antes posible.

v.ii Hibernate 4 (b)

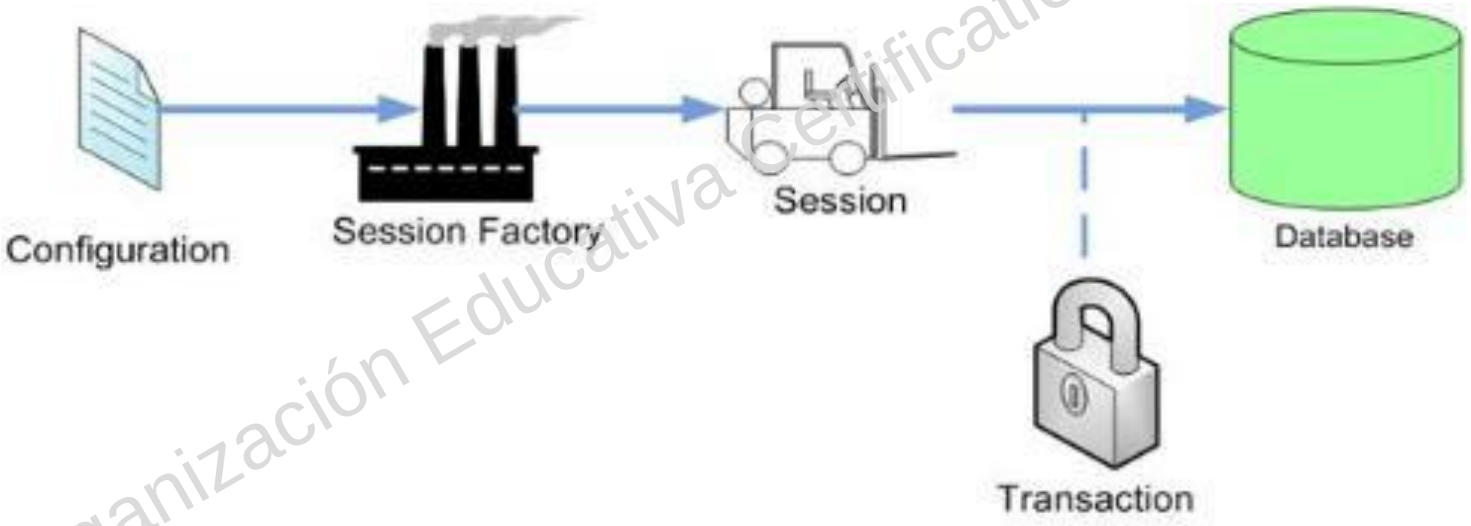
- Sesiones

- Para obtener una *Session* y poder trabajar con Hibernate primeramente es necesario configurar un objeto ***SessionFactory***.

- El objeto **SessionFactory** es el encargado de instanciar objetos Session y para hacerlo requiere como dependencia un objeto ***DataSource***.

v.ii Hibernate 4 (c)

- Sesiones



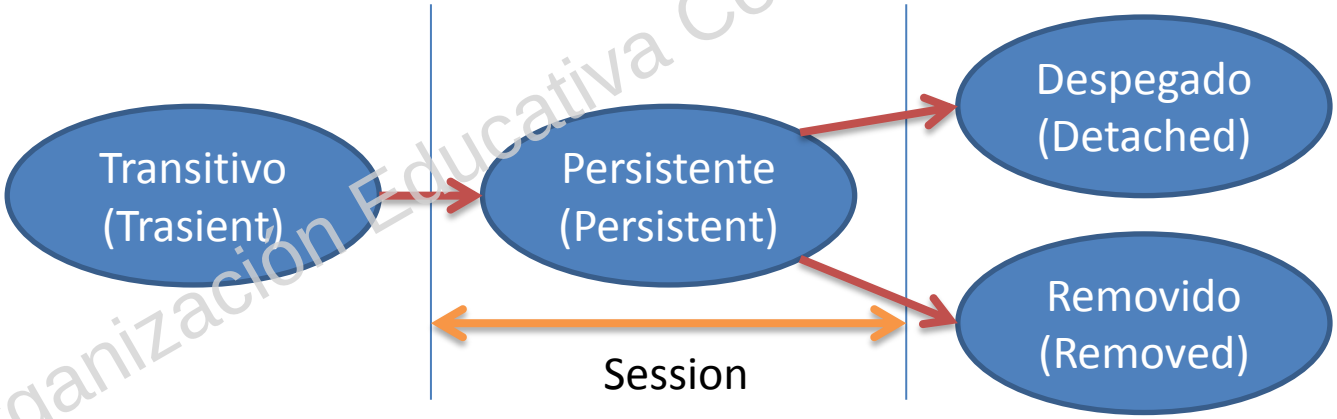
v.ii Hibernate 4

a. Sesiones
b. Ciclo de vida objetos persistentes
c. Transacciones
d. Mapeo de Entidades con Anotaciones

v.ii Hibernate 4 (a)

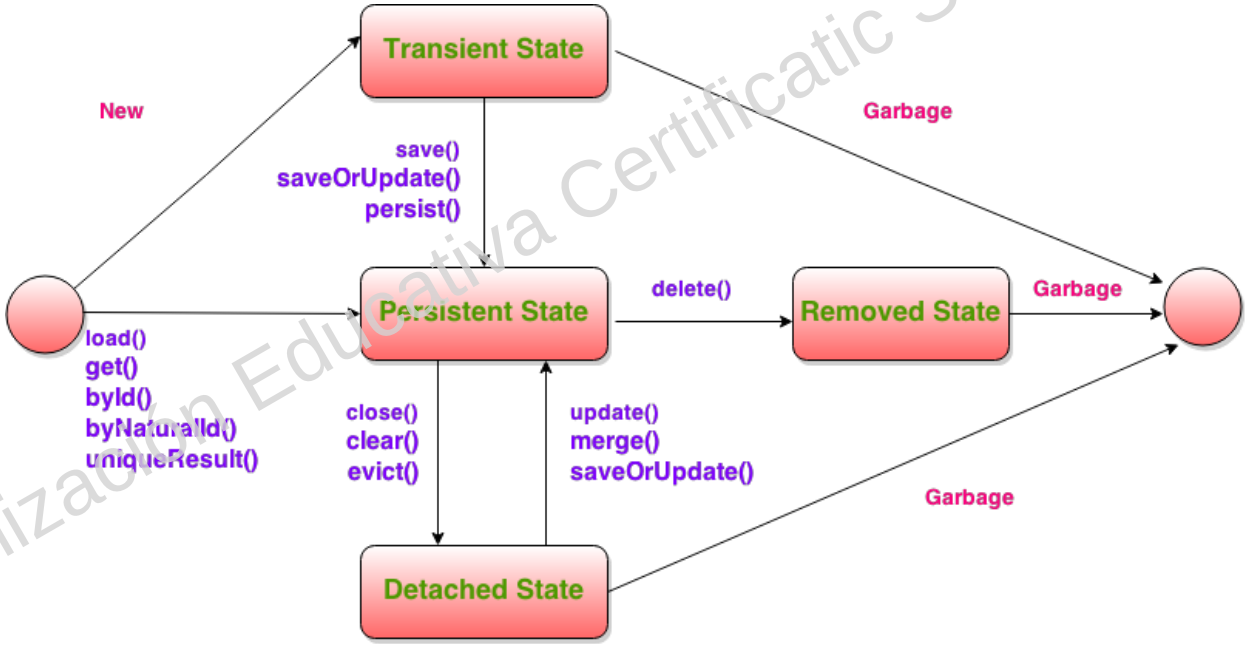
- Ciclo de vida de objetos persistentes

- Los objetos manejados por hibernate tienen 4 estados principalmente.



v.ii Hibernate 4 (b)

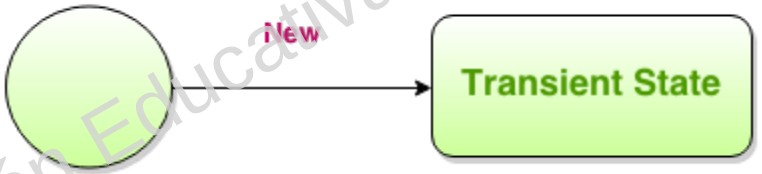
- Ciclo de vida de objetos persistentes



v.ii Hibernate 4 (c)

- Ciclo de vida de objetos persistentes

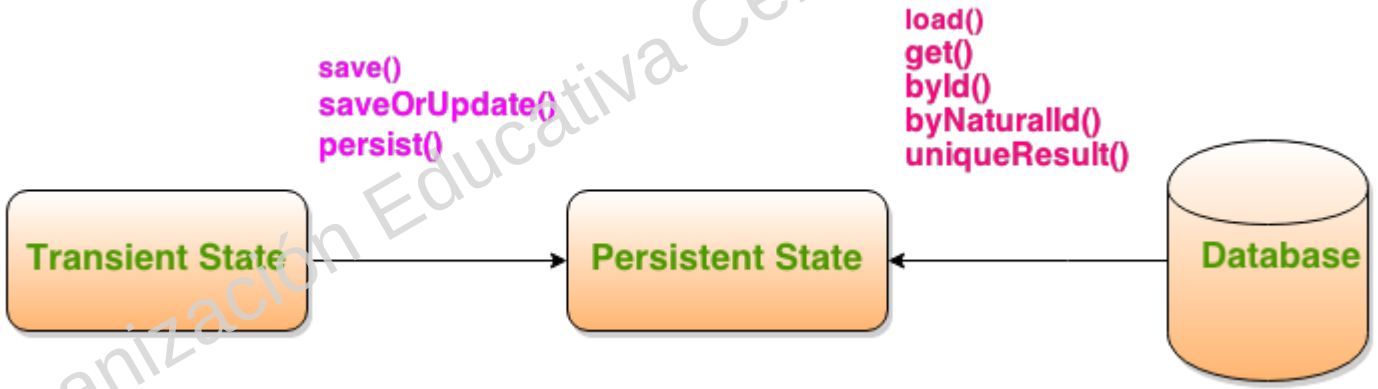
- Creación de un nuevo objeto (new)



v.ii Hibernate 4 (d)

- Ciclo de vida de objetos persistentes

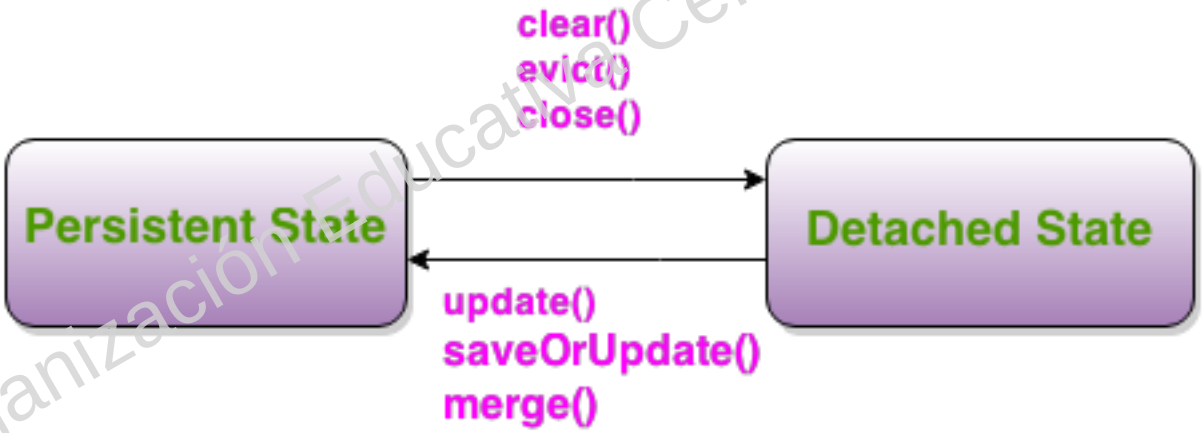
- Persistir un nuevo objeto (transitivo) u obtener un objeto persistente



v.ii Hibernate 4 (e)

- Ciclo de vida de objetos persistentes

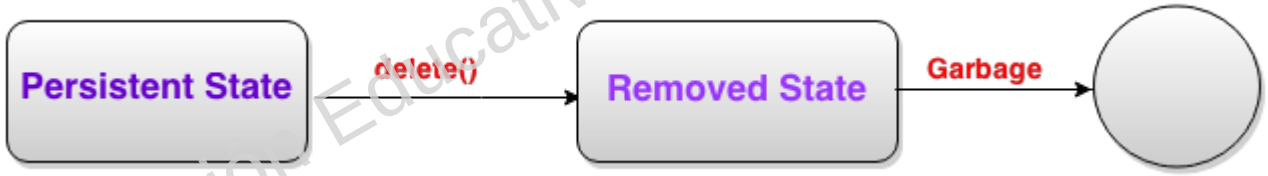
- Separar (detach) un objeto persistente o persistir un objeto separado.



v.ii Hibernate 4 (f)

- Ciclo de vida de objetos persistentes

- Eliminar un objeto persistente.



v.ii Hibernate 4

a. Sesiones
b. Ciclo de vida objetos persistentes
c. Transacciones
d. Mapeo de Entidades con Anotaciones

v.ii Hibernate 4 (a)

- Transacciones

- Como todo Framework de Persistencia, Hibernate implementa una muy consistente API Transaccional.

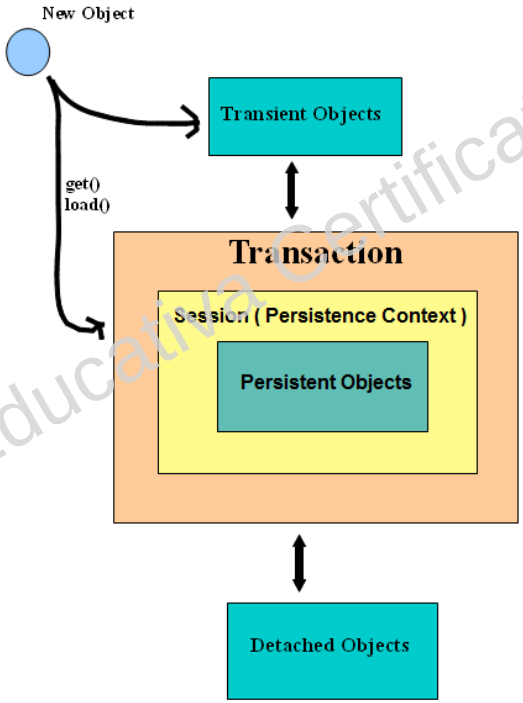
- Las Transacciones en Hibernate están asociadas a la *Session.*

- Hibernate se encarga de dejar la base de datos en un estado consistente cada ves que maneja una transacción.

- No requiere manejador de transacciones.

v.ii Hibernate 4 (b)

- Transacciones



v.ii Hibernate 4 (c)

- Transacciones, implementación de transaccionabilidad con Hibernate.

```
Session session = factory.openSession();
Transaction tx = null;
try {
    tx = session.beginTransaction();
    ...
    tx.commit();
}catch (Exception e) {
    if (tx!=null)
        tx.rollback();
    e.printStackTrace();
}finally {
    session.close();
}
```

v.ii Hibernate 4

a. Sesiones
b. Ciclo de vida objetos persistentes
c. Transacciones
d. Mapeo de Entidades con Anotaciones

v.ii Hibernate 4 (a)

- Mapeo de Entidades con Anotaciones, Entidad Account.

```
@Entity
@Table(name = "ACCOUNT_TBL")
public class Account {
    @Id
    @Column(name = "ACCOUNT_ID")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "FK_CUSTOMER_ID")
    private Customer customer;

    @Column(name = "ACCOUNT_NUMBER")
    private String accountNumber;
}
```

v.ii Hibernate 4 (b)

- Mapeo de Entidades con Anotaciones, Entidad Customer.

```
@Entity
@Table(name = "CUSTOMER_TBL")
public class Customer {
    @Id
    @Column(name = "CUSTOMER_ID")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @OneToMany(fetch = FetchType.LAZY, mappedBy = "customer")
    @Cascade(value = { org.hibernate.annotations.CascadeType.DELETE })
    private List<Account> accounts = new ArrayList<>();

    @OneToOne(fetch = FetchType.EAGER, cascade = CascadeType.ALL, mappedBy = "customer")
    private User user;

    @Column(name = "NAME")
    private String name;
}
```

v.ii Hibernate 4 (c)

- Mapeo de Entidades con Anotaciones, Entidad User.

```
@Entity
@Table(name = "USER_TBL")
public class User {
    @Id
    @Column(name = "USER_ID")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @OneToOne(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JoinColumn(name = "FK_CUSTOMER_ID")
    private Customer customer;

    @Column(name = "USERNAME")
    private String username;
}
```

Resumen de la lección

v.ii Hibernate 4

- Comprendimos los componentes principales de Hibernate.
- Comprendimos qué es y para qué se utiliza la Session de Hibernate.
- Conocimos los diferentes estados que pueden mantener los objetos manejados por Hibernate.
- Verificamos como se utiliza el API Transaccional de Hibernate.
- Revisamos a grandes rasgos como se realiza el mapeo objeto-relacional entre Entidades (modelo, POJOs) a tablas en la base de datos.

Esta página fue intencionalmente dejada en blanco.

v.iii Integración Spring ORM - Hibernate 4

Objetivos de la lección

v.iii Integración Spring ORM - Hibernate 4

- Revisar la utilidad de integrar frameworks de persistencia con Spring ORM.
- Comprender los pasos necesarios para integrar Hibernate a aplicaciones manejadas por el contenedor de IoC de Spring.
- Comprender más a fondo la utilidad del *DAOSupport.*

v.iii Integración Spring ORM - Hibernate 4

a. Configuración DataSource
b. Configuración TransactionManager
c. Configuración SessionFactory
d. Implementación Hibernate DAO

Práctica 27. Configuración capa DAO Spring ORM - Hibernate 4

v.iii Integración Spring ORM - Hibernate 4 (a)

- Spring ORM integra Hibernate, y su API Transaccional, con aplicativos manejados por el contenedor IoC de Spring.

- Spring ORM implementa las clases (beans) de integración de Hibernate en sus versiones 3, 4 y 5 (Hibernate 3 está en desuso).

- Mediante la integración de Hibernate con Spring, es posible implementar transaccionabilidad con Spring Tx mediante *TransactionManager*, implementando transaccionabilidad declarativa (Aspectos) o programática (PatformTransactionManager).

v.iii Integración Spring ORM - Hibernate 4 (b)

- Spring Tx delega la transaccionabilidad al API Transaccional de Hibernate (no reinventa) y libera al desarrollador de manipular programáticamente las transacciones con el API Transaccional de Hibernate.

- A su vez el *DAOSupport,* habilita la conversión de excepciones de terceros (Hibernate) a una única jerarquía de excepciones de acceso a datos (*DataAccessException)*. Esto ocurre gracias al uso de **@Repository** sobre los beans DAO.

v.iii Integración Spring ORM - Hibernate 4 (c)

- La conversión de excepciones también se puede configurar mediante el registro de un Bean Post Processor útil cuando los bean DAO, de Hibernate o cualquier framework de persistencia, están compilados y no se tiene acceso al código fuente Java.

```
<bean class="org.springframework.dao.annotation.
                    PersistenceExceptionTranslationPostProcessor"/>

<bean id=“otherDao" class=“other.company.dao.SomeDaoImpl"/>
```

v.iii Integración Spring ORM - Hibernate 4 (d)

- Spring ORM implementa todas las clases (beans) de utilería necesarias para integrar Hibernate así como cualquier otro framework de persistencia soportado.

- Para integrar Hibernate con Spring es necesario:
  - Registrar un bean **DataSource**
  - Registrar un bean **TransactionManager**
  - Registrar un **LocalSessionFactoryBean** encargado de construir el objeto SessionFactory necesario para instanciar objetos Session.
  - Inyectar **SessionFactory** en DAOs de Spring.

v.iii Integración Spring ORM - Hibernate 4

a. Configuración DataSource

b. Configuración TransactionManager

c. Configuración SessionFactory

d. Implementación Hibernate DAO

Práctica 27. Configuración capa DAO Spring ORM - Hibernate 4

v.iii Integración Spring ORM - Hibernate 4 (a)

- Configuración DataSource

```
<bean id="datasource" class="org.apache.commons.dbcp.BasicDataSource">
    <property name="driverClassName" value="${db.driverClassName}" />
    <property name="url" value="${db.uri}" />
    <property name="username" value="${db.user}" />
    <property name="password" value="${db.pass}" />
</bean>
```

v.iii Integración Spring ORM - Hibernate 4

a. Configuración DataSource

b. Configuración TransactionManager

c. Configuración SessionFactory

d. Implementación Hibernate DAO

Práctica 27. Configuración capa DAO Spring ORM - Hibernate 4

v.iii Integración Spring ORM - Hibernate 4 (a)

- Configuración TransactionManager

```
<tx:annotation-driven transaction-manager="transactionManager" />

<bean id="transactionManager" class="org.springframework.orm.
                                    hibernate4.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory" />
</bean>
```

v.iii Integración Spring ORM - Hibernate 4

a. Configuración DataSource
b. Configuración TransactionManager
c. Configuración SessionFactory
d. Implementación Hibernate DAO

Práctica 27. Configuración capa DAO Spring ORM - Hibernate 4

v.iii Integración Spring ORM - Hibernate 4 (a)

- Configuración SessionFactory (a)

```
<bean id="sessionFactory"  class="org.springframework.orm.
                                    hibernate4.LocalSessionFactoryBean">
    <property name="dataSource" ref="datasource" />
    <property name="packagesToScan" value="com.app.domain.entities" />
    <property name="hibernateProperties">
        <props>
            <prop key="hibernate.show_sql">true</prop>
            <prop key="dialect">org.hibernate.dialect.H2Dialect</prop>
            ...
        </props>
    </property>
</bean>
```

v.iii Integración Spring ORM - Hibernate 4 (b)

- Configuración SessionFactory (b)

```
<bean id="sessionFactory“ class="org.springframework.orm.
                                 hibernate4.LocalSessionFactoryBean">
    <property name="dataSource" ref="datasource" />
    <property name="mappingResources">
        <list>
            <value>user.hbm.xml</value>
        </list>
    </property>
    <property name="hibernateProperties">
        <props>
            ...
        </props>
    </property>
</bean>
```

v.iii Integración Spring ORM - Hibernate 4

a. Configuración DataSource
b. Configuración TransactionManager
c. Configuración SessionFactory
d. Implementación Hibernate DAO
Práctica 27. Configuración capa DAO Spring ORM - Hibernate 4

v.iii Integración Spring ORM - Hibernate 4 (a)

- Implementación Hibernate DAO

- Configurados los beans **DataSource**, **TransactionManager** y **SessionFactoryBean** (fábrica de SessionFactory).

- Inyectar **SessionFactory** sobre implementaciones DAO con Hibernate.
  - XML:

```
<bean id="customerDao" class="com.app.dao.CustomerDaoImpl">
   <property name="sessionFactory" ref="sessionFactory"/>
</bean>
```

v.iii Integración Spring ORM - Hibernate 4 (b)

- Implementación Hibernate DAO

- Inyectar **SessionFactory** sobre implementaciones DAO con Hibernate.
  - @Anotaciones:

```
@Repository
public class CustomerDaoImpl implements ICustomerDao {

    @Autowired
    private SessionFactory sessionFactory;

    ...
}
```

v.iii Integración Spring ORM - Hibernate 4

a. Configuración DataSource
b. Configuración TransactionManager
c. Configuración SessionFactory
d. Implementación Hibernate DAO

Práctica 27. Configuración capa DAO Spring ORM - Hibernate 4

v.iii Integración Spring ORM - Hibernate 4. Práctica 27. (a)

- Práctica 27. Configuración capa DAO Spring ORM - Hibernate 4

- Implementar configuración de capa de acceso a datos mediante DAOs escritos con Hibernate.
- Revisar Implementación GenericHibernateDAO (implementación productiva).

- Revisar con tranquilidad, en casa, la implementación de Hibernate propuesta.

Resumen de la lección

v.iii Integración Spring ORM - Hibernate 4

- Comprendimos los beneficios de utilizar Spring ORM para integrar Hibernate (o cualquiera de los framewors de persistencia soportados) con aplicaciones manejadas por el contenedor de IoC de Spring.
- Comprendimos más a fondo la utilidad del *DAOSupport* cuando integramos frameworks de persistencia.
- Aprendimos a integrar hibernate mediante el registro de los beans necesarios.
- Analizamos una implementación DAO genérica de Hibernate.

Esta página fue intencionalmente dejada en blanco.