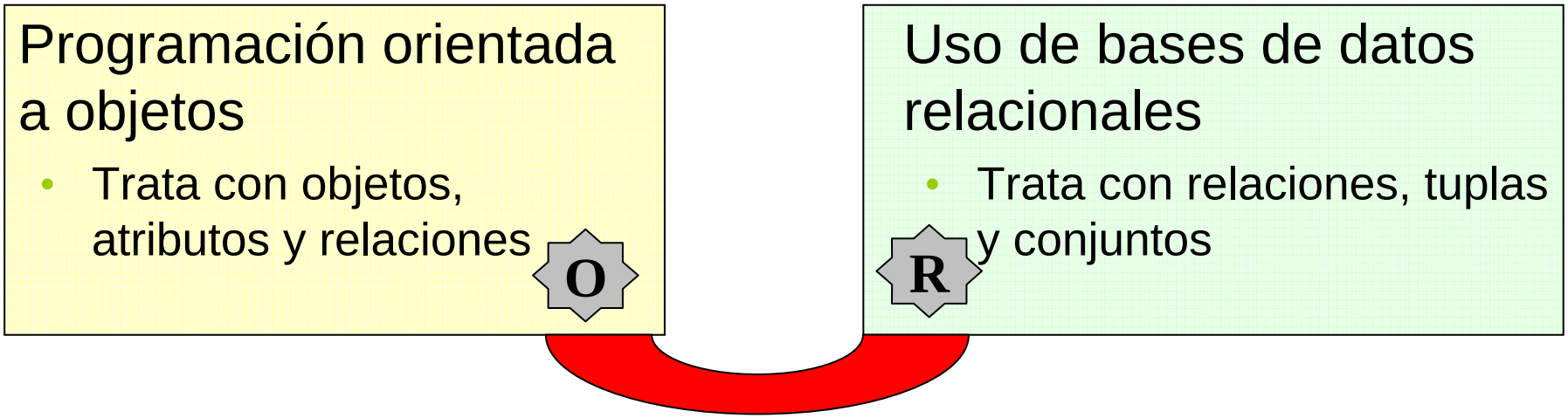
HIBERNATE

Sesión 1: Introducción a Hibernate. Configuración e inicio

Puntos a tratar

- ¿Por qué necesitamos Hibernate?
- Arquitectura Hibernate
- Configuración de Hibernate
- Resumen de pasos de configuración
- Hibernate y Eclipse

¿Por qué necesitamos Hibernate?



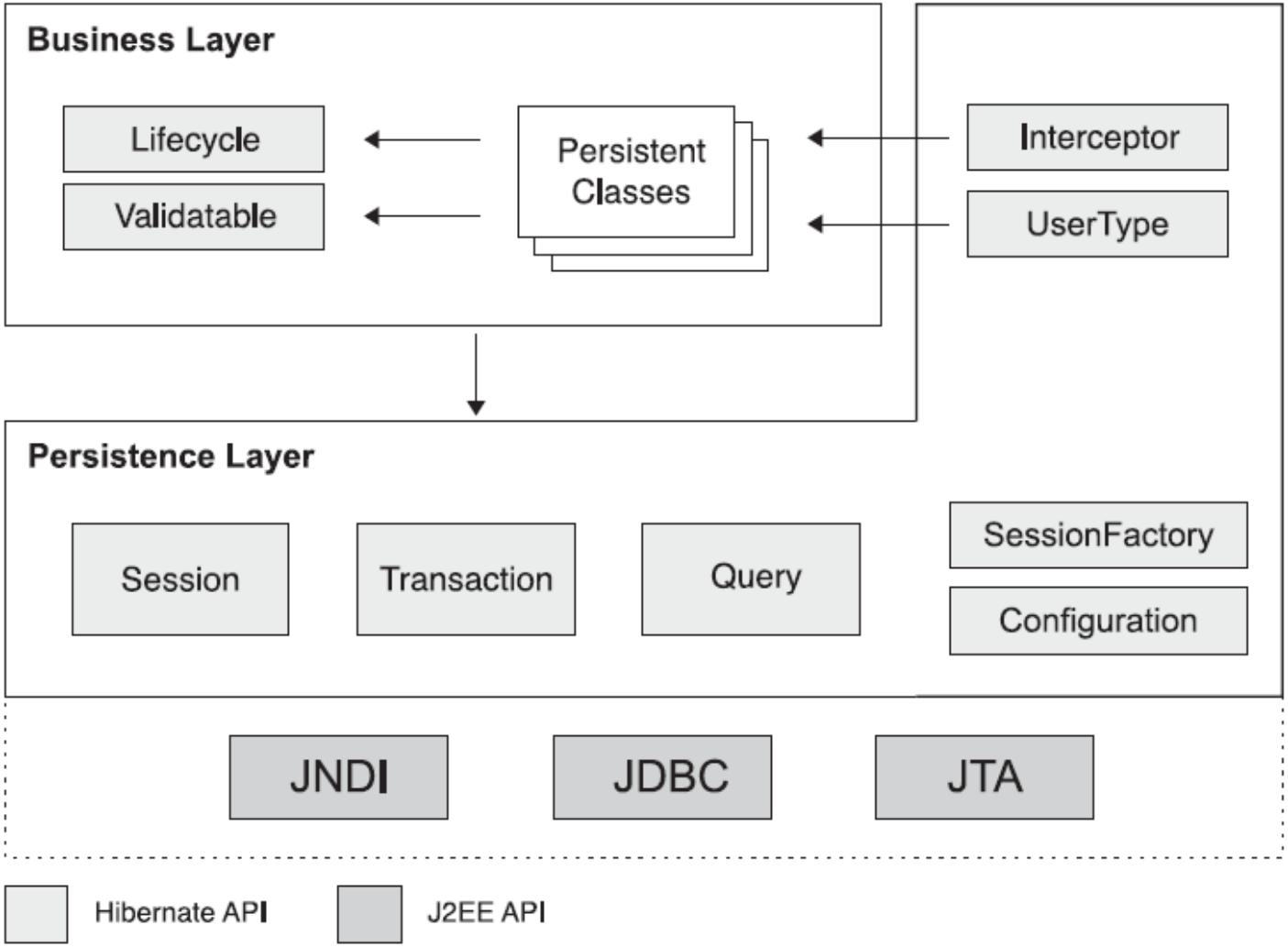
ORM: Object-Relational Mapping

- Problema: un 35% del código de una aplicación para realizar la correspondencia O ↔ R
- Solución: utilizar una ORM, por ejemplo Hibernate

Ventajas de utilizar Hibernate

- **Productividad**
  - Nos permite concentrarnos en el problema del negocio
- **Mantenibilidad**
  - Menos líneas de código
- **Rendimiento**
  - Optimización de las tareas de persistencia
- **Independencia del vendedor**

Arquitectura Hibernate



Configuración de Hibernate

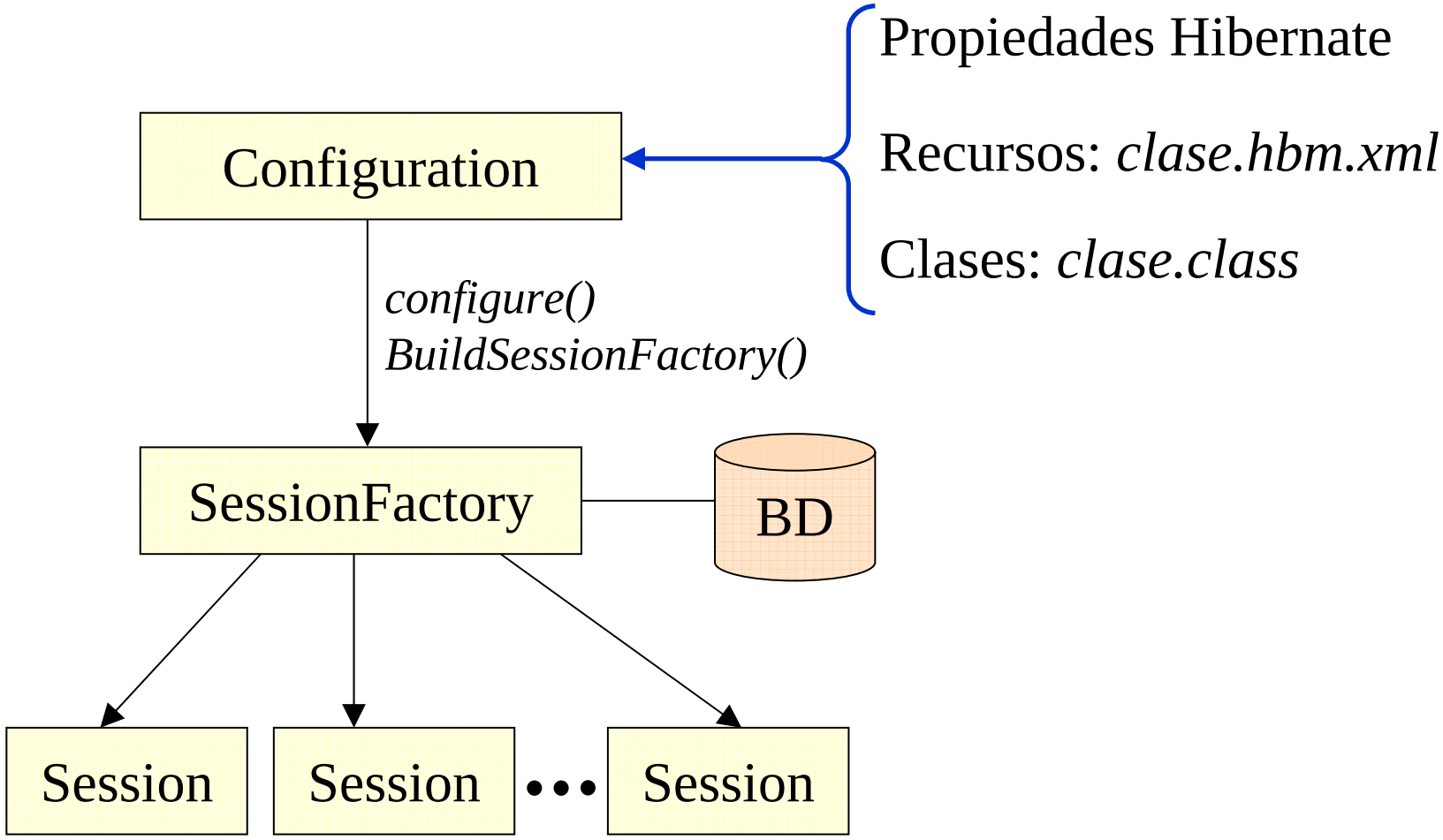
- Vamos a utilizar Hibernate en un entorno NO gestionado:
  - La propia aplicación gestiona las conexiones y transacciones

- Para utilizar hibernate necesitamos configurarlo :
  - org.hibernate.cfg.Configuration
  - Representa los mapeados entre los tipos Java y una base de datos SQL
  - Contiene un conjunto de propiedades de configuración

Formas de configurar Hibernate

- Mediante programación (código Java)
  - *new Configuration().setProperty("propiedad","valor");*
- Mediante variables incluidas como -D
  - *java -Dpropiedad=valor*
- Mediante *hibernate.properties* en *classpath*
  - Un ejemplo en directorio etc/
  - Contenido fichero: *nombre_variable valor_variable*
- Mediante *hibernate.cfg.xml*
  - Incluyendo elementos <*property*>

Inicio de Hibernate



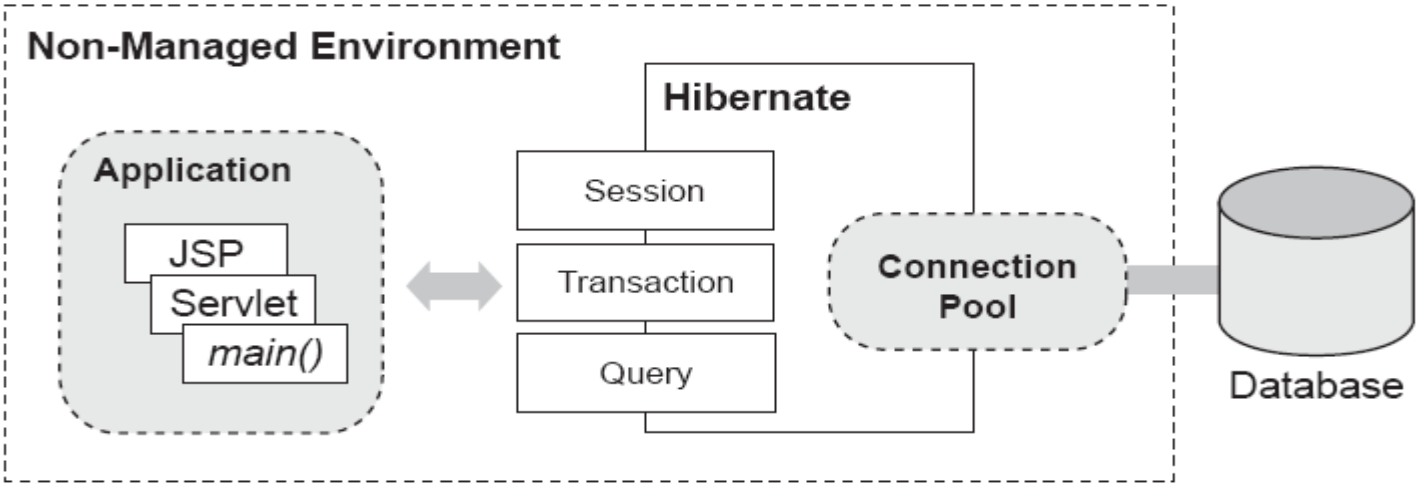
Inicialización de la SessionFactory: Ejemplos

```
Configuration cfg = new Configuration();
cfg.addResource("hello/Message.hbm.xml"); cfg.setProperties
System.getProperties()); SesionFactory sessions = cfg.buildSessionFactory();
```

```
SessionFactory sessions = new Configuration().
                  addResource ("hello/Message.hbm.xml").
                  setProperties (System.getProperties()).
        buildSessionFactory();
```

```
SessionFactory sessions = new Configuration().
                  addClass(Message.class).
                  setProperties(System.getProperties()).
                  setProperty ("hibernate.dialect",
"org.hibernate.dialect.MySQLInnoDBDialect").
                  setProperty("hibernate.connection.datasource",
 "java:comp/env/jcbc/test").buildSesionFactory();
```

Configuración de la BD




```
hibernate.connection.driver_class=com.mysql.jdbc.Driver
hibernate.connection.url = jdbc:mysql:///test
hibernate.connection.username = auctionuser
hibernate.conection.password = secret
org.hibernate.dialect.MySQLDialect
hibernate.c3p0.min_size=5
hibernate.c3p0.max_size=20
hibernate.c3p0.timeout=300
hibernate.c3p0.max_elements=50
```

Configuración de JNDI y Logging

- Enlazado de una *SessionFactory* a JNDI
  - *hibernate.session_factory_name*
  - *hibernate.jndi.url*
  - *hibernate.jndi.class*
  - Útil en entornos como Tomcat
- Uso de *Logging*
  - *log4j.jar* en el *classpath*
  - fichero *log4j.properties*

Configuración basada en XML

CABECERA

```
<hibernate-configuration>
  <session-factory name="java:hibernate/HibernateFactory">
    <!-- propiedades -->
    <property name="show_sql">true</property>
    <property name="hibernate.jndi.url"> ... </property>
    <property name="hibernate.jndi.class"> ... </property>
    <property name= "current_session_context_class">thread</property>
    <property name="hbm2ddl.auto">update</property>
    <!-- recursos -->
    <mapping resource="auction/Item.hbm.xml"/>
    <mapping resource="auction/Category.hbm.xml"/>
    <mapping resource="auction/Bid.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

gestión automática del contexto de sesiones

re-crea la BD

```
SessionFactory sf = new Configuration()
                        .configure().buildSessionFactory();
```

Pasos de configuración (Resumen)

- Situar el **.jar* del *driver* JDBC elegido y el fichero *hibernate3.jar* en nuestro *classpath*
- Añadir las dependencias de Hibernate (del directorio *lib*) en el *classpath.* (lib/README.txt contiene una lista de librerías requeridas y opcionales).
- Elegir y configurar un *pool* de conexiones JDBC.
- Determinar las propiedades de *Configuration* en un fichero *hibernate.properties* en el *classpath*.
- Crear una instancia de *Configuration* en nuestra aplicación y cargar los ficheros de mapeado XML utilizando *addResource()* o *addClass().*
- Obtener una *SessionFactory* a partir de *Configuration* llamando a *BuildSessionFactory().*

Hibernate y Eclipse (Hibernate Tools)

- Hibernate console (1 por proyecto)
  - Indica los ficheros de configuración a utilizar
  - Incluye la ruta de los POJOs y del *driver* JDBC
  - *New->Hibernate Console configuration*

- *Configuration Wizards*
  - *New->Hibernate configuration file (cfg.xml)*
  - *New->Hibernate XML mapping file (hbm.xml)*

¿Preguntas...?