USO DE MYBATIS

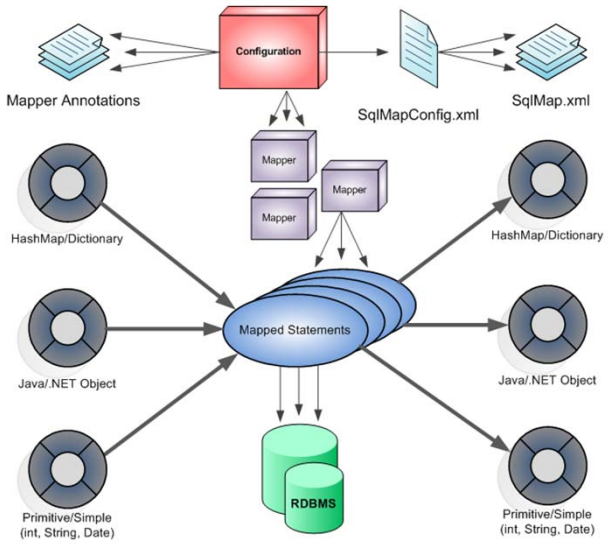
INICIO

- Descargar MyIbatis de la Web
  - http://code.google.com/p/mybatis/wiki/Downloads
  - Ibatis es una librería, no requiere proceso de instalación. Basta con descargar la distribución binaria y descomprimirlo en un directorio
  - A continuación añadir los JAR al classpath de la aplicación
    - **ibatis-core-3.jar—Shared iBATIS classes**
    - **ibatis-sqlmap-2.3.0.jar—The iBATIS SQL mapping classes**
- Documentación
  - http://mybatis.github.io/mybatis-3/index.html

COMPONENTES

- **SQLMaps**
  - Permite leer y guardar objetos Java en una BD relacional sin usar JDBC y sin mezclar Java y SQL
    - Ej. Coche.xml
- **DAO**
  - Capa de abstracción que oculta los detalles de la solución de persistencia y provee una interfaz común al resto de la aplicación
    - Ej. unican.taller.dao
- **Fichero de configuración**
  - Ej. configuracionIbatis.xml

El fichero de configuración como los de mapeo deben estar en el classpath, es decir, o bien dentro del jar, o bien en los directorios de búsqueda de clases.



FICHERO DE CONFIGURACIÓN- EJEMPLO

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE sqlMapConfig
  PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
  "http://ibatis.apache.org/dtd/sql-map-config-2.dtd">

<sqlMapConfig>
<settings enhancementEnabled="true" lazyLoadingEnabled="true"
cacheModelsEnabled="true" maxTransactions="5" maxRequests="32"
maxSessions="10" useStatementNamespaces="true" />

 <transactionManager type="JDBC" >
  <dataSource type="SIMPLE">
   <property name="JDBC.Driver" value="com.microsoft.sqlserver.jdbc.SQLServerDriver"/>
   <property name="JDBC.ConnectionURL"
              value="jdbc:sqlserver://localhost:1433;databaseName=taller"/>
   <property name="JDBC.Username" value="nombre"/>
   <property name="JDBC.Password" value="passwd"/>
  </dataSource>
 </transactionManager>

 <sqlMap resource="coche.xml" />

</sqlMapConfig>
```

Conviene ponerla para que Ibatis verifique que es correcto el resto del fichero

Se ponen tantos elementos <sqlMap resource> como se necesiten

SQLMAPS

- Se basan en ficheros de configuración XML
- Se tiene que dar un nombre (o un id) a cada consulta que vaya a ser usada en la aplicación
- Los parámetros de las consultas (entrada o salida) pueden ser primitivas (Integer, String…), objetos o HashMap
- A través de las propiedades se pueden sustituir valores que deban ser configurados dinámicamente

FICHERO DE MAPEO- EJEMPLO

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE sqlMap
  PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
  "http://ibatis.apache.org/dtd/sql-map-2.dtd" >

<sqlMap >
<select id="getCoche" resultClass=" unican.taller.model.Coche ">
     SELECT
        ID_COCHE as id,
        MARCA as marca,
        MATRICULA as matricula,
        FECHAMATRICULA as fechaMatricula,
        IDPROPIETARIO as idPropietario
      FROM COCHE
  WHERE ID_COCHE = #valor#
</select>
....
</sqlMap>
```

Alias para mapear con las propiedades de las clases DAO

```
CREATE TABLE [dbo].[Coche](
        [id_coche] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,
        [marca] [varchar](20) NULL,
        [matricula] [varchar](20) NULL unique,
        [fechaMatricula] [date] NULL,
        [idPropietario] [int] NULL)
```

FICHERO DE MAPEO- EJEMPLO

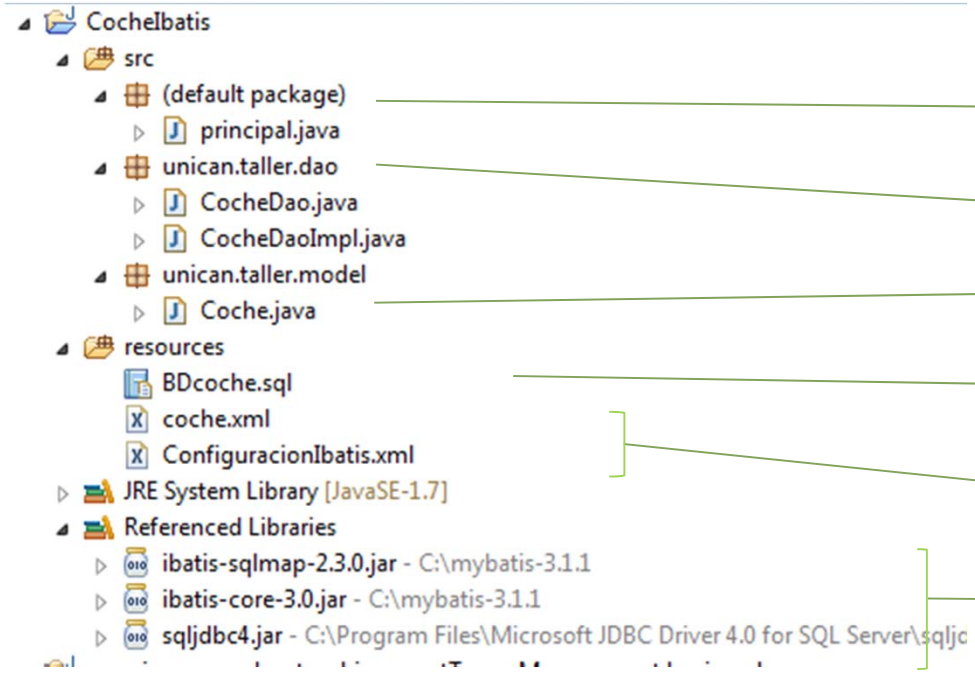
```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE sqlMap
  PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
  "http://ibatis.apache.org/dtd/sql-map-2.dtd" >

<sqlMap >
<update id="updateCoche" parameterClass="unican.taller.model.Coche">
  UPDATE COCHE SET
        MARCA = #marca#,
        MATRICULA = #matricula#,
        FECHAMATRICULA = #fechaMatricula#,
        IDPROPIETARIO = #idPropietario#
  WHERE
    ID_COCHE=#id#
</update>
….
</sqlMap>
```

FICHERO DE MAPEO

- TAGs disponibles (http://mybatis.github.io/mybatis-3/sqlmap-xml.html):
  - cache – Configuration of the cache for a given namespace.
  - cache-ref – Reference to a cache configuration from another namespace.
  - resultMap – The most complicated and powerful element that describes how to load your objects from the database result sets (consultas complejas con varios joins).
  - sql – A reusable chunk of SQL that can be referenced by other statements.
  - insert – A mapped INSERT statement.
  - update – A mapped UPDATE statement.
  - delete – A mapped DELETE statement.
  - select – A mapped SELECT statement.

PROYECTO JAVA



Prueba de capa de persistencia

Capa DAO

Modelo de Objetos de la aplicación

Script BD SQLServer

Ficheros de configuración y mapeo

Librerías Ibatis y JDBC de SQLServer

CÓDIGO JAVA

- Código para instanciar IBATIS

```
String resource = "ConfiguracionIbatis.xml";
Reader reader = Resources.getResourceAsReader(resource);
SqlMapClient sqlMap = SqlMapClientBuilder.buildSqlMapClient(reader);
```

- Código para consultar un coche

```
Integer claveCoche = 1;
Coche Muestracoche = (Coche) sqlMap.queryForObject("getCoche", claveCoche);
System.out.printf("coche 1 encontrado", Muestracoche.toString());
```

queryForObject() → devuelve un solo objeto Coche

CÓDIGO JAVA

- Código para actualizar

```
coche.setMatricula("Matr_cambiada");
sqlMap.update("updateCoche", coche);
```

- Código para consultar una lista de coches

```
List<Coche> coches = sqlMap.queryForList("getCoches", null);
```

- Código para consultar una lista de coches en HashTable

```
Map hashCoche = (Map) sqlMap.queryForObject("getHashCoche", 3);
```

RAZONES PARA USAR IBATIS

- Trabaja con cualquier BD que tenga JDBC
- Estructura de los documentos de mapeo simples, basados en xml
- Gestión de transacciones locales y globales (JTA)
- Caching configurable
- Soporte de Map, Collection, List and Primitive Wrappers (Integer, String etc.)
- Soporte de clases JavaBeans (get/set methods) y mapeo a objetos complejos (populating lists, complex object models etc.)
- Además:
  - Los modelos de objetos nunca son perfectos (ibatis no requiere que se les toque)
  - Los diseños de BD nunca son perfectos (Ibatis no requiere que se cambie)
  - Si ya se tienen ambos, es la mejor opción