



DIGIPLEX
business solutions

Framework de persistencia

MyPersistence v 1.0 beta 1

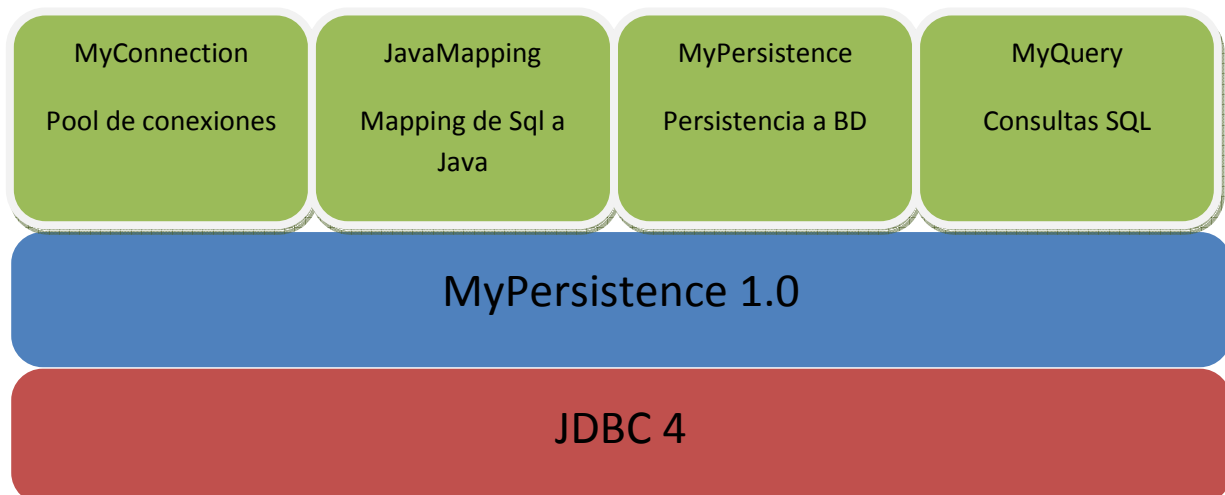
Guía básica de usuario

Introducción:

MyPersistence: es un framework de persistencia que nace como desafío personal en el año 2008; luego de experimentar muy buenos resultados con este en ambiente de laboratorio y tras someterlo a innumerables pruebas, decido pasarlo a producción en uno de mis proyectos de mediana envergadura (120 tablas) experimentando nuevamente muy buenos resultados.

Al trabajar en proyectos con frameworks como Openlaszlo (www.openlaszlo.org) y Flex (www.adobe.com/flex) comienzo a agregarle soporte para estos dos frameworks, enfocándome en Openlaszlo, dado que la mayoría de mis proyectos los realizo con este framework.

El framework: si bien existe una enorme cantidad de frameworks de persistencia, y cada uno de ellos con ventajas y desventajas, lo que intenta hacer MyPersistence es realizar el mismo trabajo de una manera simple y sin archivos xml de configuración, además de trabajar lo más cerca posible de JDBC para perder la menor performance posible.



Overview

Proceso de Mapping y persistencia en una base de datos.

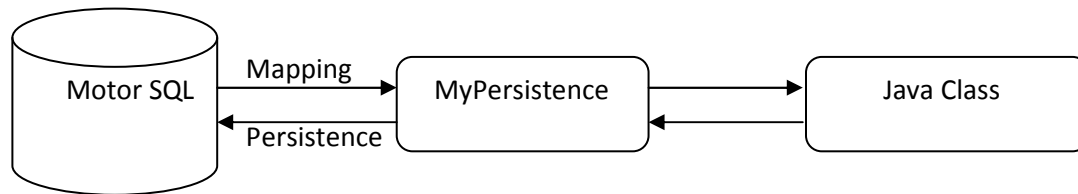


Figura 1

1. Conexión a base de datos.

1.1 Configurando el Pool de conexiones.

```
// Obtengo la instancia de MySqlConnection
```

```
MyConnection mycon = MySqlConnection.getMyConnection();
mycon.setDatabaseName("database_name");
mycon.setDiverClassName("com.mysql.jdbc.Driver");
mycon.setHost("localhost");
mycon.setPass("mypassword");
mycon.setUser("root");
mycon.setPort("3306");
mycon.setSubprotocol("jdbc:mysql");
mycon.setConnectionName("my_conexion");
mycon.setMaxConnection(30); // ← Configuro la cantidad máxima del Pool.
mycon.setComments("MyPersistence v 1.0");
```

1.2 Estableciendo al pool conexiones seguras - setSafeConnection(true)

Estableciendo el valor a "true" el pool verificará cada conexión solicitada antes de devolverla, si ésta estuviese cerrada, el pool la elimina y crea una nueva.

```
try {
    mycon.setSafeConnection(true);
    con = mycon.getConnection(); // Obtengo una conexión del Pool.
    mycon.releaseConnection(con); // devuelvo la conexión al Pool
} catch (SQLException ex) {
    ex.printStackTrace();
}
```

1.3 cargando el archivo xml de configuración.

```
try {
    mycon. loadDBPropertiesFromXml ("/home/miguel/database.xml");
} catch (MyConnectionException ex) {
    ex.printStackTrace();
}
```

2. Mapping desde Tablas SQL a objetos Java.

Una vez que diseñamos la base de datos con todas sus tablas, realizamos el proceso de mapeo.

```
try {
    JavaMapping jm = new JavaMapping(con.getConnection());
    try {
        int i = jm.mapping("C:\\Users\\MIGUEL\\Documents\\NetBeansProjects\\Parking\\src",
"com.parking.modelo");
        System.out.println("cantidad tablas mapeadas: " + i);
    } catch (MyPersistenceException ex) {
        ex.printStackTrace();
    }
} catch (SQLException ex) {
    ex.printStackTrace();
}
```

El método Mapping recibe dos parámetros, el primero es el path donde se guardarán las clases java, mientras que el segundo es el “package” de las clases, en el ejemplo “com.parking.modelo”, el sistema creará de forma automática los directorios.

```
Salida: cantidad tablas mapeadas: 108
```

3. Operaciones SQL con MyPersistence.

3.1 Insert

```
Persona per = new Persona();  
(setters ...)  
MyPersistence myper = MyPersistence.getMyPersistence();  
try {  
    myper.insert(per);  
} catch (SQLException ex) {  
  
} catch (MyPersistenceException ex) {  
}
```

Realizar un insert en la base de datos es muy sencillo, obtenemos la instancia de MyPersistence, luego en un bloque try, invocamos al método “insert” y le pasamos como parámetro el objeto a persistir.

3.2 Update

```
Persona per = new Persona();  
(setters ...)  
MyPersistence myper = MyPersistence.getMyPersistence();  
try {  
    myper.update(per);  
} catch (SQLException ex) {  
  
} catch (MyPersistenceException ex) {  
}
```

Para realizar un update, invocamos el metodo “update” y le pasamos el objeto a actualizar.

3.3 Delete

```
Persona per = new Persona();  
(setters ...)  
MyPersistence myper = MyPersistence.getMyPersistence();  
try {  
    myper.delete(per);  
} catch (SQLException ex) {  
  
} catch (MyPersistenceException ex) {  
}
```

Para realizar un delete, invocamos el metodo “delete” y le pasamos el objeto a eliminar.

4. Transacciones

```
try {
    myper.beginTransaction();
    myper.insert(per);
    user.setIdPersona(myper.getGeneratedKey());
    myper.insert(user);
    myper.commit();
} catch (SQLException ex) {

} catch (MyPersistenceException ex) {
}
```

Para realizar una transacción se debe inicializar la misma invocando el método `beginTransaction()`. Luego invocamos las operaciones SQL que necesitemos realizar y al final invocamos el método `commit()` para efectivizar la transacción.