MyBatis - productos

1. Script para base de datos: **data**

```
DROP TABLE IF EXISTS productos;

-- tabla productos
CREATE TABLE productos (
   idproducto  int(11) NOT NULL AUTO_INCREMENT,
   titulo      varchar(200) NOT NULL,
   tipo        varchar(20) NOT NULL,
   precio      double(10,2) NOT NULL,
   PRIMARY KEY (idproducto),
   UNIQUE KEY IDX_productos_1 (titulo)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


-- insertando filas de prueba
INSERT INTO productos(titulo, tipo, precio)
VALUES('Consulta Paginada con jQuery', 'Separata', 10.0);
INSERT INTO productos(titulo, tipo, precio)
VALUES('Combos Anidados con jQuery', 'Video', 10.0);
INSERT INTO productos(titulo, tipo, precio)
VALUES('Transacciones en MySQL', 'Separata', 15.0);
INSERT INTO productos(titulo, tipo, precio)
VALUES('Store Procedure en Oracle', 'Separata', 15.0);
INSERT INTO productos(titulo, tipo, precio)
VALUES('PDF con Spring', 'Video', 15.0);
```

2. Generar el proyecto **myBatis-productos** y agregar el **driver de MySQL** y **mybatis-3.2.3.jar**

3. En el paquete **config** crear **database.properties**

```
database.driver=com.mysql.jdbc.Driver
database.url=jdbc:mysql://localhost:3306/data
database.username= root
database.password= root
```

4. En el paquete **mybatis** crear **mybatis-config.xml**

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE configuration
PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>
  <properties resource="config/database.properties"/>

  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC"/>
      <dataSource type="POOLED">
        <property name="driver" value="${database.driver}"/>
        <property name="url" value="${database.url}"/>
        <property name="username" value="${database.username}"/>
        <property name="password" value="${database.password}"/>
      </dataSource>
    </environment>
  </environments>
</configuration>
```

5. En el paquete **mybatis** crear la clase **SessionFactory.java**

```
package mybatis;

import java.io.IOException;
import java.io.Reader;
import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

public class SessionFactory {

  protected static final SqlSessionFactory FACTORY;

  static {
    try {
      Reader reader = Resources.getResourceAsReader(
          "mybatis/mybatis-config.xml");
```

```
            FACTORY = new SqlSessionFactoryBuilder().build(reader);

        } catch (IOException e) {
            throw new RuntimeException("Error: " + e, e);
        }
    }

    public static SqlSessionFactory getSqlSessionFactory() {
        return FACTORY;
    }
}
```

6. En el paquete **pruebas** crear la clase **PruebaConexion.java** y ejecute para probar la conexión

```
package pruebas;

import mybatis.SessionFactory;
import org.apache.ibatis.session.SqlSession;

public class PruebaConexion {

    public static void main(String[] args) {
        try (SqlSession session = SessionFactory.getSqlSessionFactory().openSession()) {

            if (session.getConnection() != null) {
                System.out.println("conexion exitosa...");
            } else {
                System.out.println("problemas de conexion...");
            }
        }
    }
}
```

7. En el paquete **dto** genere la clase **Productos**

```
package dto;

public class Productos {

    private Integer idproducto;
    private String titulo;
```

```
    private String tipo;
    private Double precio;

    public Productos() {
    }

    public Integer getIdproducto() {
        return idproducto;
    }

    public void setIdproducto(Integer idproducto) {
        this.idproducto = idproducto;
    }

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public String getTipo() {
        return tipo;
    }

    public void setTipo(String tipo) {
        this.tipo = tipo;
    }

    public Double getPrecio() {
        return precio;
    }

    public void setPrecio(Double precio) {
        this.precio = precio;
    }
}
```

8. En el paquete **mybatis.mapper** gener la interface **ProductosMapper**

```
package mybatis.mapper;

import dto.Productos;
import java.util.List;
import org.apache.ibatis.annotations.Delete;
import org.apache.ibatis.annotations.Insert;
import org.apache.ibatis.annotations.Param;
import org.apache.ibatis.annotations.Select;
import org.apache.ibatis.annotations.Update;

public interface ProductosMapper {

   @Select(PRODUCTOS_QRY)
   public List<Productos> productosQry();

   @Select(PRODUCTOS_GET)
   public Productos productosGet(@Param("idproducto") Integer idproducto);

   @Insert(PRODUCTOS_INS)
   public int productosIns(Productos productos);

   @Update(PRODUCTOS_UPD)
   public int productosUpd(Productos productos);

   @Delete(PRODUCTOS_DEL)
   public int productosDel(@Param("ids") String ids);

   // sentecias SQL
   String PRODUCTOS_QRY
         = "SELECT idproducto, titulo, tipo, precio "
         + "FROM productos ORDER BY titulo";

   String PRODUCTOS_GET
         = "SELECT idproducto, titulo, tipo, precio FROM productos "
         + "WHERE idproducto=#{idproducto}";

   String PRODUCTOS_INS
         = "INSERT INTO productos(titulo, tipo, precio) "
         + "VALUES(#{titulo}, #{tipo}, #{precio})";
```

```
    String PRODUCTOS_UPD
        = "UPDATE productos SET "
        + "titulo=#{titulo}, tipo=#{tipo}, precio=#{precio} "
        + "WHERE idproducto=#{idproducto}";

    String PRODUCTOS_DEL
        = "DELETE FROM productos WHERE idproducto IN(${ids})";
}
```

9. Mapear productos creando en el paquete **mybatis.mapper** el documento **ProductosMapper.xml**

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >

<mapper namespace="mybatis.mapper.ProductosMapper"> <!-- interface java -->

  <resultMap id="productos" type="dto.Productos">
    <id column="idproducto" property="idproducto" jdbcType="INTEGER"/>
    <result column="titulo" property="titulo" jdbcType="VARCHAR"/>
    <result column="tipo" property="tipo" jdbcType="VARCHAR"/>
    <result column="precio" property="precio" jdbcType="DOUBLE" />
  </resultMap>
</mapper>
```

10. En **mybatis-config.xml** configurar para acceder al mapeo:

```
<configuration>
  <properties resource="config/database.properties"/>

  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC"/>
      <dataSource type="POOLED">
        <property name="driver" value="${database.driver}"/>
        <property name="url" value="${database.url}"/>
        <property name="username" value="${database.username}"/>
        <property name="password" value="${database.password}"/>
      </dataSource>
    </environment>
```

```
    </environments>

    <mappers>
      <mapper resource="mybatis/mapper/ProductosMapper.xml"/>
    </mappers>

</configuration>
```

11. En el paquete **pruebas** crear la clase **PruebaQry**

```
package pruebas;

import dto.Productos;
import java.util.List;
import mybatis.SessionFactory;
import mybatis.mapper.ProductosMapper;
import org.apache.ibatis.session.SqlSession;

public class PruebaQry {

   public static void main(String[] args) {
      List<Productos> list;

      try (SqlSession session = SessionFactory.getSqlSessionFactory().openSession()) {
         ProductosMapper mapper = session.getMapper(ProductosMapper.class);

         list = mapper.productosQry();
      }

      if (list != null) {
         for (Productos p : list) {
            System.out.println(String.format("%-4d%-40s%-10s%8.2f",
                  p.getIdproducto(), p.getTitulo(),
                  p.getTipo(), p.getPrecio()));
         }
      } else {
         System.out.println("consulta sin resultados...");
      }
   }
}
```

12. En el paquete **pruebas** crear la clase **PruebaIns**

```
package pruebas;

import dto.Productos;
import mybatis.SessionFactory;
import mybatis.mapper.ProductosMapper;
import org.apache.ibatis.exceptions.PersistenceException;
import org.apache.ibatis.session.SqlSession;

public class PruebaIns {

   public static void main(String[] args) {
      Productos p1 = new Productos();
      Productos p2 = new Productos();

      p1.setTitulo("aaa");
      p1.setTipo("Separata");
      p1.setPrecio(12D);

      p2.setTitulo("bbb");
      p2.setTipo("Video");
      p2.setPrecio(15d);
      //

      SqlSession session = SessionFactory.getSqlSessionFactory().openSession();
      ProductosMapper mapper = session.getMapper(ProductosMapper.class);

      try {
         mapper.productosIns(p1);
         mapper.productosIns(p2);

         session.commit();
         System.out.println("operacion exitosa...");

      } catch (PersistenceException e) {
         System.out.println(e.getMessage());
         session.rollback();
      }
   }
```

```
}
```

13. Continuar probando **productosGet**, **ProductosUpd** y **productosDel**

Parte 2: full anotaciones

1. Eliminar **ProductosMapper.xml** de **mybatis-config.xml** , de tal forma que este, se vea así:

```
<configuration>
  <properties resource="config/database.properties"/>

  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC"/>
      <dataSource type="POOLED">
        <property name="driver" value="${database.driver}"/>
        <property name="url" value="${database.url}"/>
        <property name="username" value="${database.username}"/>
        <property name="password" value="${database.password}"/>
      </dataSource>
    </environment>
  </environments>
</configuration>
```

2. Eliminar el archivo **ProductosMapper.xml**

3. Editar **ProductosMapper.java** con el siguiente código:

```
package mybatis.mapper;

import dto.Productos;
import java.util.List;
import org.apache.ibatis.annotations.Delete;
import org.apache.ibatis.annotations.Insert;
import org.apache.ibatis.annotations.Options;
import org.apache.ibatis.annotations.Param;
import org.apache.ibatis.annotations.Result;
import org.apache.ibatis.annotations.Results;
import org.apache.ibatis.annotations.Select;
import org.apache.ibatis.annotations.Update;

public interface ProductosMapper {
```

```
@Select(PRODUCTOS_QRY)
@Results(value = {
   @Result(property = "idproducto", column = "idproducto"),
   @Result(property = "titulo", column = "titulo"),
   @Result(property = "tipo", column = "tipo"),
   @Result(property = "precio", column = "precio")
})
public List<Productos> productosQry();

@Select(PRODUCTOS_GET)
@Results(value = {
   @Result(property = "idproducto", column = "idproducto"),
   @Result(property = "titulo", column = "titulo"),
   @Result(property = "tipo", column = "tipo"),
   @Result(property = "precio", column = "precio")
})
public Productos productosGet(@Param("idproducto") Integer idproducto);

@Insert(PRODUCTOS_INS)
@Options(useGeneratedKeys = true, keyProperty = "idproducto")
public int productosIns(Productos productos);

@Update(PRODUCTOS_UPD)
public int productosUpd(Productos productos);

@Delete(PRODUCTOS_DEL)
public int productosDel(@Param("ids") String ids);

// sentecias SQl
String PRODUCTOS_QRY
      = "SELECT idproducto, titulo, tipo, precio "
      + "FROM productos ORDER BY titulo";

String PRODUCTOS_GET
      = "SELECT idproducto, titulo, tipo, precio FROM productos "
      + "WHERE idproducto=#{idproducto}";

String PRODUCTOS_INS
      = "INSERT INTO productos(titulo, tipo, precio) "
      + "VALUES(#{titulo}, #{tipo}, #{precio})";

String PRODUCTOS_UPD
```

```
        = "UPDATE productos SET "
        + "titulo=#{titulo}, tipo=#{tipo}, precio=#{precio} "
        + "WHERE idproducto=#{idproducto}";

    String PRODUCTOS_DEL
        = "DELETE FROM productos WHERE idproducto IN(${ids})";

}
```

4. Registrar el mapper **ProductosMapper.class** en **SessionFactory.java**

```
package mybatis;

import java.io.IOException;
import java.io.Reader;
import mybatis.mapper.ProductosMapper;
import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

public class SessionFactory {

  protected static final SqlSessionFactory FACTORY;

  static {
    try {
      Reader reader = Resources.getResourceAsReader(
          "mybatis/mybatis-config.xml");

      FACTORY = new SqlSessionFactoryBuilder().build(reader);
      // se adicionan los mapper
      FACTORY.getConfiguration().addMapper(ProductosMapper.class);
      // ---

    } catch (IOException e) {
      throw new RuntimeException("Error: " + e, e);
    }
  }

  public static SqlSessionFactory getSqlSessionFactory() {
    return FACTORY;
```

```
    }
}
```

5. Ejecute **PruebaQry**

6. Editar **PruabaIns**

```
package pruebas;

import dto.Productos;
import mybatis.SessionFactory;
import mybatis.mapper.ProductosMapper;
import org.apache.ibatis.exceptions.PersistenceException;
import org.apache.ibatis.session.SqlSession;

public class PruebaIns {

   public static void main(String[] args) {

      Productos p1 = new Productos();
      Productos p2 = new Productos();

      p1.setTitulo("aaa");
      p1.setTipo("Separata");
      p1.setPrecio(12D);

      p2.setTitulo("bbb");
      p2.setTipo("Video");
      p2.setPrecio(15d);
      //

      SqlSession session = SessionFactory.getSqlSessionFactory().openSession();
      ProductosMapper mapper = session.getMapper(ProductosMapper.class);

      try {
         mapper.productosIns(p1);
         System.out.println("ID generado: " + p1.getIdproducto());

         mapper.productosIns(p2);
         System.out.println("ID generado: " + p2.getIdproducto());
```

```
            session.commit();
            System.out.println("operacion exitosa...");

        } catch (PersistenceException e) {
            System.out.println(e.getMessage());
            session.rollback();
        }
    }
}
```

7. Seguir con los demás métodos