Spring JDBC

Spring JDBC .-

- Este modulo de Spring trabaja en la capa de persistencia.

- Permite acceder a la BBDD a través del estándar JDBC, como también posibilita la integración con otros framework de persistencia como Hibernate, Ibatis, JPA, …etc.

Recursos necesarios .-

- Necesitamos los siguientes componentes:
- **DataSource**; contiene los datos de la conexión.
- **Template**; plantilla para la reutilización de código.
- **Mapeador**; clase que transforma el ResultSet en la instancia del modelo.
- **Dao**; clase que recoge todos los métodos de acceso a la BBDD.

DataSource .-

- Se declara en el archivo XML como un bean de una de estas clases:

- **DriverManagerDataSource**; esta clase me proporciona una conexión nueva cada vez.

- **SingleConnectionDataSource**; actúa como un Singleton, esto es, siempre devuelve la misma conexión.

Crear un DataSource .-

```
<bean id="dataSource"
      class="org.springframework.jdbc.datasource.DriverManagerDataSource">
      <property name="driverClassName" value="com.mysql.jdbc.Driver" />
      <property name="url" value="jdbc:mysql://localhost/curso_spring" />
      <property name="username" value="root" />
      <property name="password" value="root" />
</bean>
```

Template .-

- Spring proporciona 3 tipos de plantilla:
- **JdbcTemplate**; permite acceso a JDBC. Es la plantilla más básica.
- **NamedParametersJdbcTemplate**; Se utiliza para parámetros con nombre.
- **SimpleJdbcTemplate**; Recoge las novedades de java 5 como por ejemplo los tipos genéricos

Crear la plantilla .-

```
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
      <property name="dataSource" ref="dataSource" />
</bean>
```

```
<bean id="namedTemplate"
      class="org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate">
      <constructor-arg ref="dataSource" />
</bean>
```

```
<bean id="jdbcTemplate"
      class="org.springframework.jdbc.core.simple.SimpleJdbcTemplate">
   <constructor-arg ref="dataSource" />
</bean>
```

Crear el mapeador .-

```
public class ProductoMapeador implements RowMapper<Producto> {

    public Producto mapRow(ResultSet rs, int rowNum) throws SQLException {
        Producto producto = new Producto();
        producto.setCodigo(rs.getInt("codigo"));
        producto.setDescripcion(rs.getString("descripcion"));

        return producto;
    }
}
```

```
<bean id="mapeador" class="es.prac.spring.ProductoMapeador" />
```

Crear el DAO .-

```
public class DaoProducto {

        private JdbcTemplate jdbcTemplate;
        private RowMapper<Producto> mapeador;

        public List<Producto> findAllProductos() {
                String sql = "select codigo, descripcion from productos";
                return (List<Producto>) (jdbcTemplate.query(sql, mapeador));
        }

        public void insertarProducto(int codigo, String descripcion) {
                jdbcTemplate.update("insert into productos values(" + codigo +
                        ",'"+ descripcion + "')");
        }
```

Configurar el DAO .-

```
<bean id="daoProducto" class="es.prac.spring.DaoProducto">
      <property name="jdbcTemplate" ref="jdbcTemplate" />
      <property name="mapeador" ref="mapeador" />
</bean>
```

DAO con patrón Factory Method .-

```
<bean id="daoProducto" class="datos.DAOFactory"
                       factory-method="getProductoDAO">
      <property name="jdbcTemplate" ref="namedTemplate" />
      <property name="mapeador" ref="mapeador" />
</bean>
```