Universidad Nacional Autónoma de México

Dirección General de Cómputo de Tecnologías de Información y Comunicación

Diplomado

Desarrollo de Sistemas con Tecnología Java

Práctica Cuatro

“JdbcTempate en Spring Boot”

Mtro. ISC. Miguel Ángel Sánchez Hernández

Tabla de contenido

1. Creación del proyecto con Spring Tools 4

Vamos a crear un proyecto Spring con Spring Tools 4, los pasos son los siguientes:

1. Una vez que esté abierto el IDE, presionar las teclas Ctrl + N.
2. Señalamos Spring Boot→Spring Starter Project.
3. Presionamos Next, en la siguiente pantalla llenamos los campos con:
   - Name: spring-core-bd
   - Type: Maven Project
   - Packaging: Jar
   - Java Version: 11
   - Languaje: Java
   - Group: dgtic.core
   - Artifact: spring-core-bd
   - Version: 1.0
   - Description: Data JDBC con Spring Boot
   - Package: dgtic.core
4. Presionamos Next, ahora escogemos lo siguiente:
   - Spring Boot Version: 2.7.1
   - Spring Data JDBC
   - Spring MySQL Driver
5. Presionamos el botón Finish.

2. MySQL crear esquema

Ejecutar el siguiente Script para crear el esquema.

```
SET NAMES 'UTF8MB4';
DROP DATABASE IF EXISTS dgtic;
CREATE DATABASE IF NOT EXISTS dgtic DEFAULT CHARACTER SET UTF8MB4;
USE dgtic;

CREATE TABLE estudiantes(
cuenta_etd                                          VARCHAR(3) NOT NULL ,
nombre                                              VARCHAR(40) NOT NULL,
edad                                                INTEGER NOT NULL,
PRIMARY KEY(cuenta_etd),
CONSTRAINT verificar_cuenta CHECK (REGEXP_LIKE(cuenta_etd,'[a-z]{1}[0-9]{2}'))
)DEFAULT CHARACTER SET UTF8MB4;

CREATE TABLE materias(
id_mtr                                              INTEGER NOT NULL AUTO_INCREMENT,
nombre                                              VARCHAR(40) NOT NULL,
creditos                                            INTEGER NOT NULL,
PRIMARY KEY(id_mtr)
)DEFAULT CHARACTER SET UTF8MB4;

CREATE TABLE estudiante_materia(
cuenta_etd                                          VARCHAR(3) NOT NULL ,
id_mtr                                              INTEGER NOT NULL,
PRIMARY KEY(cuenta_etd,id_mtr),
FOREIGN KEY(cuenta_etd) REFERENCES estudiantes(cuenta_etd) ON DELETE CASCADE,
FOREIGN KEY(id_mtr) REFERENCES materias(id_mtr)
)DEFAULT CHARACTER SET UTF8MB4;

CREATE TABLE usuarios(
```

```
usuario_usa                         VARCHAR(10) NOT NULL ,
clave                               VARCHAR(20) NOT NULL,
alta                                INT(2)  NOT NULL,
PRIMARY KEY(usuario_usa)
)DEFAULT CHARACTER SET UTF8MB4;

CREATE TABLE autorizacion(
usuario_usa                         VARCHAR(10) NOT NULL ,
rol                                 VARCHAR(5) NOT NULL CHECK (rol IN ('admin','user')),
PRIMARY KEY(usuario_usa),
FOREIGN KEY(usuario_usa) REFERENCES usuarios(usuario_usa) ON DELETE CASCADE
)DEFAULT CHARACTER SET UTF8MB4;
```

3. MySQL insertar datos en el esquema

Ejecutar el siguiente Script para insertar dato en el esquema.

```
truncate table estudiante_materia;
delete from usuarios;
delete from autorizacion;
delete from materias;
delete from estudiantes;
insert into estudiantes(cuenta_etd,nombre,edad) values('A00','Rosa',19);
insert into estudiantes(cuenta_etd,nombre,edad) values('A01','Pedro',22);
insert into estudiantes(cuenta_etd,nombre,edad) values('A02','Samuel',19);
insert into estudiantes(cuenta_etd,nombre,edad) values('A03','Sara',21);
insert into estudiantes(cuenta_etd,nombre,edad) values('A04','Alfredo',24);
alter table materias auto_increment=1;
insert into materias(nombre,creditos) values('Lógica',9);
insert into materias(nombre,creditos) values('Programación',10);
insert into materias(nombre,creditos) values('Circuitos lógicos',9);
insert into materias(nombre,creditos) values('Estructura de Datos',9);
insert into materias(nombre,creditos) values('Spring',10);
insert into materias(nombre,creditos) values('Diseño y Análisis de Algoritmos',9);
insert into estudiante_materia value('A00',2);
insert into estudiante_materia value('A01',1);
insert into estudiante_materia value('A01',2);
insert into estudiante_materia value('A01',5);
insert into estudiante_materia value('A02',1);
insert into estudiante_materia value('A02',2);
insert into estudiante_materia value('A03',3);
insert into estudiante_materia value('A03',4);
insert into estudiante_materia value('A04',5);
insert into usuarios(usuario_usa,clave,alta) values('dgtic','$2a$10$5OPg9MiA6NsHLqqso2KsOeJonhpkAc4iNTzwNCi1BbyvqkoycXO6',1);
insert into usuarios(usuario_usa,clave,alta) values('usuario','$2a$10$FqCXhbXkUuYsBF.Y9zcNNuYtyxjuVtft3AqVnZxhYmDn8opw0G3Pu',1);
insert into autorizacion(usuario_usa,rol) values('dgtic','admin');
insert into autorizacion(usuario_usa,rol) values('usuario','user');
```

4. Archivo application.properties

En la carpeta src/main/resources modificar el archivo application.properties con lo siguiente:

```
spring.datasource.url=jdbc:mysql://localhost:3306/dgtic?useSSL=false&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=1234
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

5. Clase Estudiante

Crear la clase Estudiante con los siguientes datos:

- Nombre de la clase: Estudiante
- Paquete: dgtic.core.persistence.modelo

```
package dgtic.core.persistence.modelo;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
public class Estudiante {
	private String cuenta;
	private String nombre;
	private int edad;
	private List<Materia> materias=new ArrayList<>();
	public Estudiante() {
		// TODO Auto-generated constructor stub
	}
	public Estudiante(String cuenta, String nombre, int edad, Materia ... materia) {
		super();
		this.cuenta = cuenta;
		this.nombre = nombre;
		this.edad = edad;
		this.materias.addAll(Arrays.asList(materia));
	}
	public String getCuenta() {
		return cuenta;
	}
	public void setCuenta(String cuenta) {
		this.cuenta = cuenta;
	}
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public int getEdad() {
		return edad;
	}
	public void setEdad(int edad) {
		this.edad = edad;
	}
	public List<Materia> getMaterias() {
		return materias;
	}
	public void setMaterias(List<Materia> materias) {
		this.materias = materias;
	}
	@Override
	public String toString() {
		return "Estudiante [cuenta=" + cuenta + ", nombre=" + nombre + ", edad=" + edad + ", materias=" + materias
				+ "]";
	}
}
```

6. Clase Materia

Crear la clase Materia con los siguientes datos:

- Nombre de la clase: Materia
- Paquete: dgtic.core.persistence.modelo

```
package dgtic.core.persistence.modelo;
public class Materia {
	private Integer id;
	private String nombre;
	private int creditos;
	public Materia() {
		// TODO Auto-generated constructor stub
	}
	public Materia(String nombre, int creditos) {
		super();
		this.nombre = nombre;
		this.creditos = creditos;
	}
	public Integer getId() {
		return id;
	}
	public void setId(Integer id) {
		this.id = id;
	}
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public int getCreditos() {
		return creditos;
	}
	public void setCreditos(int creditos) {
		this.creditos = creditos;
	}
	@Override
	public String toString() {
		return "Materia [id=" + id + ", nombre=" + nombre + ", creditos=" + creditos + "]";
	}
}
```

7. Clase EstudianteMateria

Crear la clase EstudianteMateria con los siguientes datos:

- Nombre de la clase: EstudianteMateria
- Paquete: dgtic.core.persistence.modelo

```
package dgtic.core.persistence.modelo;
public class EstudianteMateria {
	private String idEstudiante;
	private int idMateria;
	public EstudianteMateria() {
		// TODO Auto-generated constructor stub
	}
	public EstudianteMateria(String idEstudiante, int idMateria) {
		super();
		this.idEstudiante = idEstudiante;
		this.idMateria = idMateria;
	}
	public String getIdEstudiante() {
		return idEstudiante;
	}
	public void setIdEstudiante(String idEstudiante) {
		this.idEstudiante = idEstudiante;
	}
	public int getIdMateria() {
		return idMateria;
	}
	public void setIdMateria(int idMateria) {
		this.idMateria = idMateria;
	}
	@Override
	public String toString() {
		return "EstudianteMateria [idEstudiante=" + idEstudiante + ", idMateria=" + idMateria + "]";
	}
}
```

8. Interfaz EstudianteDao

Crear la interfaz EstudianteDao con los siguientes datos:

- Nombre de la interfaz: EstudianteDao
- Paquete: dgtic.core.persistence.dao

```
package dgtic.core.persistence.dao;
import java.util.List;
import dgtic.core.persistence.modelo.Estudiante;
public interface EstudianteDao {
	public List<Estudiante> consulta();
	public Estudiante consultaId(String cuenta);
	public void insertar(Estudiante estudiante);
	public void cambiar(Estudiante estudiante);
	public void borrar(Estudiante estudiante);
}
```

9. Clase EstudianteDaoRepository

Crear la clase EstudianteDaoRepository con los siguientes datos:

- Nombre de la clase: EstudianteDaoRepository
- Paquete: dgtic.core.persistence.dao

```
package dgtic.core.persistence.dao;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;
import dgtic.core.persistence.modelo.Estudiante;
@Repository
public class EstudianteDaoRepository implements EstudianteDao {
	private JdbcTemplate jdbc;

	@Autowired
	private MateriaDao materia;

	@Autowired
	public EstudianteDaoRepository(JdbcTemplate jdbc) {
		super();
		this.jdbc = jdbc;
	}

	@Override
	public List<Estudiante> consulta() {
		String query = "select cuenta_etd,nombre,edad\r\n"
				+ "from estudiantes \r\n"
				+ "order by cuenta_etd;";
		return jdbc.query(query, this::mapRowtoEstudiante);
	}

	@Override
	public Estudiante consultaId(String cuenta) {
		String query = "select cuenta_etd,nombre,edad\r\n"
				+ "from estudiantes \r\n"
				+ " where cuenta_etd='"+cuenta +"' order by cuenta_etd;";
```

```
                return jdbc.query(query, this::mapRowtoEstudiante).get(0);

        }

        @Override
        public void insertar(Estudiante estudiante) {
                jdbc.update("insert into estudiantes(cuenta_etd,nombre,edad) "
                                + "values(?,?,?)",estudiante.getCuenta(),
                                estudiante.getNombre(),estudiante.getEdad());
        }

        @Override
        public void cambiar(Estudiante estudiante) {
                jdbc.update("update estudiantes set nombre=?,edad=? where "
                                + "cuenta_etd=?",estudiante.getNombre()
                                ,estudiante.getEdad(),estudiante.getCuenta());
        }

        @Override
        public void borrar(Estudiante estudiante) {
                jdbc.update("delete from estudiantes where cuenta_etd=?",
                                estudiante.getCuenta());
        }

        private Estudiante mapRowtoEstudiante(ResultSet rs, int row) throws SQLException {
                Estudiante est = new Estudiante();
                est.setCuenta(rs.getString(1));
                est.setNombre(rs.getString(2));
                est.setEdad(rs.getInt(3));
                est.getMaterias().addAll(((MateriaDaoRepository)materia)
                                .consultaMateria(est.getCuenta()));
                return est;
        }
}
```

10. Interfaz MateriaDao

Crear la interfaz MateriaDao con los siguientes datos:

- Nombre de la interfaz: MateriaDao
- Paquete: dgtic.core.persistence.dao

```
package dgtic.core.persistence.dao;
import java.util.List;
import dgtic.core.persistence.modelo.Materia;
public interface MateriaDao {
        public List<Materia> consulta();
        public Materia consultaId(int id);
        public int insertar(Materia materia);
        public void cambiar(Materia materia);
        public void borrar(Materia materia);
}
```

11. Clase MateriaDaoRepository

Crear la clase MateriaDaoRepository con los siguientes datos:

- Nombre de la clase: MateriaDaoRepository
- Paquete: dgtic.core.persistence.dao

```
package dgtic.core.persistence.dao;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.sql.DataSource;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
import org.springframework.stereotype.Repository;
import dgtic.core.persistence.modelo.Materia;

@Repository
public class MateriaDaoRepository implements MateriaDao{
	private JdbcTemplate jdbc;
	private SimpleJdbcInsert insertar;

	@Autowired
	public MateriaDaoRepository(DataSource source) {
		this.jdbc = new JdbcTemplate(source);
		this.insertar=new SimpleJdbcInsert(this.jdbc).
					withTableName("materias").usingGeneratedKeyColumns("id_mtr");
	}

	@Override
	public List<Materia> consulta() {
		String query = "select id_mtr,nombre,creditos\r\n"
					+ "from materias \r\n"
					+ "order by id_mtr;";
		return jdbc.query(query, this::mapRowtoMateria);
	}

	@Override
	public Materia consultaId(int id) {
		String query = "select id_mtr,nombre,creditos\r\n"
					+ "from materias \r\n"
					+ " where id_mtr='"+id +"' order by id_mtr;";
		return jdbc.query(query, this::mapRowtoMateria).get(0);
	}

	@Override
	public int insertar(Materia materia) {
		final Map<String, Object> parametros = new HashMap<>();
		parametros.put("nombre", materia.getNombre());
		parametros.put("creditos",materia.getCreditos());
		Number llave=insertar.executeAndReturnKey(parametros);
		return  llave.intValue();
	}

	@Override
	public void cambiar(Materia materia) {
		jdbc.update("update materias set nombre=?,creditos=? where "
					+ "id_mtr=?",materia.getNombre()
					,materia.getCreditos(),materia.getId());
```

```
	}

	@Override
	public void borrar(Materia materia) {
		jdbc.update("delete from materias where id_mtr=?",
					materia.getId());
	}
	public List<Materia> consultaMateria(String cuenta) {
		String query = "select b.id_mtr,b.nombre,b.creditos \r\n"
					+ "from estudiante_materia a,materias b \r\n"
					+ "where a.cuenta_etd='"+cuenta+"' and a.id_mtr=b.id_mtr;";
		return jdbc.query(query, this::mapRowtoMateria);
	}
	private Materia mapRowtoMateria(ResultSet rs, int row) throws SQLException {
		Materia mat = new Materia();
		mat.setId(rs.getInt(1));
		mat.setNombre(rs.getString(2));
		mat.setCreditos(rs.getInt(3));
		return mat;
	}
}
```

12. Interfaz EstudianteMateriaDao

Crear la interfaz EstudianteMateriaDao con los siguientes datos:

- Nombre de la interfaz: EstudianteMateriaDao
- Paquete: dgtic.core.persistence.dao

```
package dgtic.core.persistence.dao;
import java.util.List;
import dgtic.core.persistence.modelo.EstudianteMateria;
public interface EstudianteMateriaDao {
	public List<EstudianteMateria> consulta();
	public void insertar(EstudianteMateria estudianteMateria);
	public void cambiar(EstudianteMateria viejoEstudianteMateria,
				EstudianteMateria nuevoEstudianteMateria);
	public void borrar(EstudianteMateria estudianteMateria);
}
```

13. Clase EstudianteMateriaDaoRepository

Crear la clase EstudianteMateriaDaoRepository con los siguientes datos:

- Nombre de la clase: EstudianteMateriaDaoRepository
- Paquete: dgtic.core.persistence.dao

```
package dgtic.core.persistence.dao;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;
import dgtic.core.persistence.modelo.EstudianteMateria;
```

```
@Repository
public class EstudianteMateriaDaoRepository implements EstudianteMateriaDao {
	private JdbcTemplate jdbc;

	@Autowired
	public EstudianteMateriaDaoRepository(JdbcTemplate jdbc) {
		super();
		this.jdbc = jdbc;
	}

	@Override
	public List<EstudianteMateria> consulta() {
		String query = "select cuenta_etd,id_mtr\r\n"
					+ "from estudiante_materia \r\n"
					+ "order by cuenta_etd;";
		return jdbc.query(query, this::mapRowtoEstMat);
	}


	@Override
	public void insertar(EstudianteMateria estudianteMateria) {
		jdbc.update("insert into estudiante_materia(cuenta_etd,id_mtr) "
					+ "values(?,?)",estudianteMateria.getIdEstudiante(),
					estudianteMateria.getIdMateria());
	}

	@Override
	public void cambiar(EstudianteMateria viejoEstudianteMateria,
				EstudianteMateria nuevoEstudianteMateria) {
		jdbc.update("update estudiante_materia set cuenta_etd=?,"
					+ "id_mtr=? where "
					+ "cuenta_etd=? and id_mtr=?"
					,nuevoEstudianteMateria.getIdEstudiante()
					,nuevoEstudianteMateria.getIdMateria()
					,viejoEstudianteMateria.getIdEstudiante()
					,viejoEstudianteMateria.getIdMateria());

	}

	@Override
	public void borrar(EstudianteMateria estudianteMateria) {
		jdbc.update("delete from estudiante_materia where "
					+ "cuenta_etd=? and id_mtr=?",
					estudianteMateria.getIdEstudiante()
					,estudianteMateria.getIdMateria());

	}
	private EstudianteMateria mapRowtoEstMat(ResultSet rs, int row) throws SQLException {
		EstudianteMateria estMat = new EstudianteMateria();
		estMat.setIdEstudiante(rs.getString(1));
		estMat.setIdMateria(rs.getInt(2));
		return estMat;
	}

}
```

14. Test en la clase SpringCoreBdApplicationTests

Modificar la clase SpringCoreBdApplicationTests con los siguientes datos:

- Nombre de la clase: SpringCoreBdApplicationTests
- Paquete (src/test/java): dgtic.core

```
package dgtic.core;
import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import dgtic.core.persistence.dao.EstudianteDao;
import dgtic.core.persistence.modelo.Estudiante;
@SpringBootTest
class SpringCoreBdApplicationTests {
	@Autowired
	private EstudianteDao estudiante;

	@Test
	void estudiante() {
			assertEquals(5,estudiante.consulta().size());
			String actual=estudiante.consultaId("A00").getNombre();
			assertEquals("Rosa", actual);
			Estudiante es=new Estudiante();
			es.setCuenta("A05");
			es.setNombre("DGTIC");
			es.setEdad(23);
			//insertar
			estudiante.insertar(es);
			assertEquals(6,estudiante.consulta().size());

			//actualizar
			es.setNombre("DGTIC 2");
			estudiante.cambiar(es);
			assertEquals("DGTIC 2", estudiante.consultaId(es.getCuenta()).getNombre());

			//borrar
			estudiante.borrar(es);
			assertEquals(5,estudiante.consulta().size());

	}

}
```

15. Clase SpringCoreBdApplication

Modificar la clase SpringCoreBdApplication con los siguientes datos:

- Nombre de la clase: SpringCoreBdApplication
- Paquete: dgtic.core

```
package dgtic.core;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import dgtic.core.persistence.dao.EstudianteDao;
import dgtic.core.persistence.dao.EstudianteMateriaDao;
import dgtic.core.persistence.dao.MateriaDao;
import dgtic.core.persistence.modelo.Estudiante;
import dgtic.core.persistence.modelo.EstudianteMateria;
import dgtic.core.persistence.modelo.Materia;

@SpringBootApplication
public class SpringCoreBdApplication implements CommandLineRunner {
	@Autowired
	private EstudianteDao estudiante;

	@Autowired
	private MateriaDao materia;

	@Autowired
	private EstudianteMateriaDao estMat;

	public static void main(String[] args) {
		SpringApplication.run(SpringCoreBdApplication.class, args);
	}

	@Override
	public void run(String... args) throws Exception {
		// probando estudiantes
		/*
		 * Estudiante est=new Estudiante(); est.setCuenta("A06");
		 * est.setNombre("DGTIC"); est.setEdad(20); //insertar
		 * //estudiante.insertar(est);
		 *
		 * //actualizar //est.setNombre("DGTIC-2"); //estudiante.cambiar(est);
		 *
		 * //borrar //estudiante.borrar(est);
		 *
		 * for(Estudiante e:estudiante.consulta()) { System.out.println(e); }
		 */
		/*
		 * //probando materia Materia mat=new Materia();
		 * mat.setNombre("Ingeniería de Software"); mat.setCreditos(9);
		 *
		 * //insertar mat.setId(materia.insertar(mat)); for(Materia
		 * m:materia.consulta()) { System.out.println(m); } System.out.println("-----");
		 * //actualizar mat.setNombre("DGTIC"); materia.cambiar(mat); for(Materia
		 * m:materia.consulta()) { System.out.println(m); } System.out.println("-----");
		 * //borrar materia.borrar(mat); for(Materia m:materia.consulta()) {
		 * System.out.println(m); }
		 */

		// probando estudiante_materia
		EstudianteMateria e = new EstudianteMateria();
```

```
                e.setIdEstudiante("A00");
                e.setIdMateria(1);

                /*//insertar
                try {
                        estMat.insertar(e);
                } catch (Exception exp) {
                        System.out.println("error");
                }*/
                /*
                //actualizar
                EstudianteMateria nuevo = new EstudianteMateria();
                nuevo.setIdEstudiante("A01");
                nuevo.setIdMateria(4);
                estMat.cambiar(e,nuevo);
                */
                /*
                //borrar
                EstudianteMateria nuevo = new EstudianteMateria();
                nuevo.setIdEstudiante("A01");
                nuevo.setIdMateria(4);
                estMat.borrar(nuevo);

                for (EstudianteMateria datos : estMat.consulta()) {
                        System.out.println(datos);
                }*/

        }

}
```