Universidad Nacional Autónoma de México

Dirección General de Cómputo de Tecnologías de Información y Comunicación

Diplomado

Desarrollo de Sistemas con Tecnología Java

Ejercicio Quince

“Mockito”

Mtro. ISC. Miguel Ángel Sánchez Hernández

Tabla de contenido

1. Copiar un proyecto en JntelliJ IDEA

Para copiar un proyecto, hacemos lo siguiente:

1. Señalamos el proyecto spring-core-pojodaotest
2. Apretamos Ctrl + C
3. Luego Ctrl + V
4. Project Name: spring-core-pojodaotestmockito
5. En el archivo pom.xml cambiar las siguientes líneas por lo siguiente
    - <artifactId>spring-core-javabean</artifactId>
        - <artifactId>spring-core-pojodaotestmockito</artifactId>
    - Cambiar las etiquetas <name> con
        - <name> spring-core-pojodaotestmockito</name>
    - Cambiar la etiqueta <description> con
        - <description> Ejemplo de configuración Mockito en Spring </description>
6. Actualizar maven

2. Clase Estudiante

Crear la clase Estudiante con los siguientes datos:

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

```
package dgtic.core.modelo;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Objects;
public class Estudiante {
	private String matricula;
	private String nombre;
	private int edad;
	private List<Materia> materias=new ArrayList<>();
	public Estudiante() {
		// TODO Auto-generated constructor stub
	}
	public Estudiante(String matricula, String nombre, int edad) {
		super();
		this.matricula = matricula;
		this.nombre = nombre;
		this.edad = edad;
	}
	public String getMatricula() {
		return matricula;
	}
	public void setMatricula(String matricula) {
		this.matricula = matricula;
	}
	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(Materia ...materias ) {
		this.materias.addAll(Arrays.asList(materias));
	}
	@Override
	public String toString() {
		return "Estudiante [matricula=" + matricula + ", nombre=" + nombre + ", edad=" + edad + ", materias=" + materias
				+ "]";
	}

	@Override
	public int hashCode() {
		return Objects.hash(edad, materias, matricula, nombre);
	}
	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Estudiante other = (Estudiante) obj;
		return edad == other.edad && Objects.equals(materias, other.materias)
				&& Objects.equals(matricula, other.matricula) && Objects.equals(nombre, other.nombre);
	}
}
```

3. Clase Materia

Crear la clase Materia con los siguientes datos:

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

```
package dgtic.core.modelo;
import java.util.Objects;
import dgtic.core.excepciones.CreditosMenores;
public class Materia {
		private Long id;
		private String nombre;
		private Integer creditos;
		public Materia() {
			// TODO Auto-generated constructor stub
		}

		public Materia(Long id, String nombre, Integer creditos) {
			super();
			this.id = id;
			this.nombre = nombre;
			this.creditos = creditos;
		}

		public Long getId() {
			return id;
		}


		public void setId(Long id) {
			this.id = id;
		}

		public String getNombre() {
			return nombre;
		}
		public void setNombre(String nombre) {
			this.nombre = nombre;
		}
		public Integer getCreditos() {
			return creditos;
		}
		public void setCreditos(Integer creditos) {
			if(creditos<0) {
				throw new CreditosMenores("No creditos negativos");
			}else {
				this.creditos = creditos;
			}

		}
		@Override
		public String toString() {
			return "Materia [nombre=" + nombre + ", creditos=" + creditos + "]";
		}
		@Override
		public int hashCode() {
			return Objects.hash(creditos, nombre);
		}
```

```
	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Materia other = (Materia) obj;
		return Objects.equals(creditos, other.creditos) && Objects.equals(nombre, other.nombre);
	}

}
```

4. Clase BaseDeDatos

Crear la clase Materia con los siguientes datos:

- Nombre de la clase: BaseDeDatos
- Paquete: dgtic.core.repositorio

```
package dgtic.core.repositorio;
import java.util.ArrayList;
import java.util.List;
import dgtic.core.modelo.Estudiante;
import dgtic.core.modelo.Materia;
public class BaseDeDatos {
	public static List<Estudiante> estudiantes=new ArrayList<>();
	static {
		Estudiante est=new Estudiante("123","Rosa",20);
		est.setMaterias(new Materia(1l,"Cálculo",9),new Materia(2l,"Programación",10),
				new Materia(3l,"Lógica",10));
		estudiantes.add(est);
		est=new Estudiante("124","Tomas",22);
		est.setMaterias(new Materia(2l,"Programación",10),
				new Materia(3l,"Lógica",10));
		estudiantes.add(est);
		est=new Estudiante("125","Mario",20);
		est.setMaterias(new Materia(1l,"Cálculo",9),new Materia(4l,"Circuitos Lógicos",10),
				new Materia(5l,"Lógica de Autómatas",10));
		estudiantes.add(est);
		est=new Estudiante("126","Esmeralda",22);
		est.setMaterias(new Materia(4l,"Circuitos Lógicos",10),
				new Materia(5l,"Lógica de Autómatas",10));
		estudiantes.add(est);

	}
}
```

5. Interfaz EstudianteRepositorio

Crear la interfaz EstudianteRespositorio con los siguientes datos:

- Nombre de la clase: EstudianteRespositorio
- Paquete: dgtic.core.repositorio.intf

```
package dgtic.core.repositorio.intf;
import java.util.List;
import dgtic.core.modelo.Estudiante;
public interface EstudianteRepositorio {
		public List<Estudiante> findAll();
		public Estudiante findById(String matricula);
		public void save(Estudiante estudiante);
}
```

6. Interfaz MateriaRepositorio

Crear la interfaz MateriaRepositorio con los siguientes datos:

- Nombre de la clase: MateriaRepositorio
- Paquete: dgtic.core.repositorio.intf

```
package dgtic.core.repositorio.intf;
import java.util.List;
import dgtic.core.modelo.Materia;
public interface MateriaRepositorio {
		public List<Materia> findAll();
		public Materia findById(Long id);
		public void save(Materia materia);
}
```

7. Interfaz EstudianteServicio

Crear la interfaz EstudianteServicio con los siguientes datos:

- Nombre de la clase: EstudianteServicio
- Paquete: dgtic.core.servicio

```
package dgtic.core.servicio;
import java.util.List;
import dgtic.core.modelo.Estudiante;
public interface EstudianteServicio {
		public List<Estudiante> findAll();
		public Estudiante findById(String matricula);
		public String archivoCSV(String matricula);
		public int creditos(String matricula);
}
```

8. Clase EstudianteServicioImpl

Crear la clase EstudianteServicioImpl con los siguientes datos:

- Nombre de la clase: EstudianteServicioImpl
- Paquete: dgtic.core.servicio

```
package dgtic.core.servicio;
import java.util.List;
import java.util.stream.Collectors;
import org.springframework.stereotype.Service;
import dgtic.core.modelo.Estudiante;
import dgtic.core.repositorio.intf.EstudianteRepositorio;
import dgtic.core.repositorio.intf.MateriaRepositorio;
//@Service
public class EstudianteServicioImpl implements EstudianteServicio{
		private EstudianteRepositorio estudianteRepositorio;
		private MateriaRepositorio materiaRepositorio;
		public EstudianteServicioImpl(EstudianteRepositorio estudianteRepositorio, MateriaRepositorio materiaRepositorio) {
			super();
			this.estudianteRepositorio = estudianteRepositorio;
			this.materiaRepositorio = materiaRepositorio;
		}
		@Override
		public List<Estudiante> findAll() {
			return estudianteRepositorio.findAll();
		}

		@Override
		public Estudiante findById(String matricula) {
			return estudianteRepositorio.findById(matricula);

		}

		@Override
		public String archivoCSV(String matricula) {
			// TODO Auto-generated method stub
			return estudianteRepositorio.findAll().stream()
					.filter(est->(est.getMatricula().equals(matricula)))
					.map(est->(est.getMatricula()+";"+
							(est.getMaterias().stream()
								.map(mat->(mat.getId()+";"+mat.getNombre()+";"
										+mat.getCreditos()))
								.collect(Collectors.joining(";")))
							+";"+est.getNombre()+";"+est.getEdad()))
					.collect(Collectors.joining("\n"));

		}
		@Override
		public int creditos(String matricula) {
			return estudianteRepositorio.findAll().stream()
					.filter(est->(est.getMatricula().equals(matricula)))
					.findFirst()
					.map(est->(est.getMaterias().stream()
							.map(xx->xx.getCreditos()).reduce(0,Integer::sum)))
					.get();
		}
}
```

9. Clase EstudianteServicioImplTest

Crear la clase EstudianteServicioImplTest con los siguientes datos:

- Nombre de la clase: EstudianteServicioImplTest
- Paquete (test/java): dgtic.core.servicio

```
package dgtic.core.servicio;
import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.context.annotation.ComponentScan;
import dgtic.core.excepciones.CreditosMenores;
import dgtic.core.modelo.Estudiante;
import dgtic.core.modelo.Materia;
import dgtic.core.repositorio.BaseDeDatos;
import dgtic.core.repositorio.intf.EstudianteRepositorio;
import dgtic.core.repositorio.intf.MateriaRepositorio;
import dgtic.core.servicio.EstudianteServicio;
import dgtic.core.servicio.EstudianteServicioImpl;
@SpringBootTest(classes = {EstudianteServicioImplTest.class})
@ComponentScan(basePackages = "dgtic.core")
class EstudianteServicioImplTest {
		private EstudianteRepositorio estudianteRespositorio;
		private MateriaRepositorio materiaRepositorio;
		private EstudianteServicio estudianteServicio;

		@BeforeEach
		public void inicio() {
			estudianteRespositorio=mock(EstudianteRepositorio.class);
			materiaRepositorio=mock(MateriaRepositorio.class);
			estudianteServicio=new EstudianteServicioImpl(estudianteRespositorio, materiaRepositorio);
		}

		@Test
		void testUno() {

			when(estudianteRespositorio.findAll()).thenReturn(BaseDeDatos.estudiantes);
			when(estudianteRespositorio.findById(anyString())).thenReturn(BaseDeDatos.estudiantes.stream()
					.filter(e->e.getMatricula().equals("123"))
					.findFirst()
					.get());
			assertEquals(4, estudianteServicio.findAll().size());
			assertEquals(3,estudianteRespositorio.findById("123").getMaterias().size());
			assertEquals("Rosa", estudianteServicio.findById("123").getNombre());
			assertEquals("123;1;Cálculo;9;2;Programación;10;3;Lógica;10;Rosa;20",
					estudianteServicio.archivoCSV("123"));
			assertEquals(29, estudianteServicio.creditos("123"));

		}
		@Test
		void testDos() {
			when(estudianteRespositorio.findAll()).thenReturn(BaseDeDatos.estudiantes);
			when(estudianteRespositorio.findById("123")).thenReturn(BaseDeDatos.estudiantes.stream()
					.filter(e->e.getMatricula().equals("123"))
					.findFirst()
					.get());
			//assertEquals(3,estudianteRespositorio.findById("123").getMaterias().size());
			//verify(estudianteRespositorio).findById(anyString());
```

```
                //assertEquals(29, estudianteServicio.creditos("123"));
                //verify(estudianteRespositorio).findAll();

                /*assertEquals(4, estudianteServicio.findAll().size());
                assertEquals("123;1;Cálculo;9;2;Programación;10;3;Lógica;10;Rosa;20",
                                estudianteServicio.archivoCSV("123"));
                verify(estudianteRespositorio,times(2)).findAll();*/

        }

        @Test
        void testTres() {
                when(estudianteRespositorio.findAll()).thenReturn(BaseDeDatos.estudiantes);
                when(estudianteRespositorio.findById("123")).thenReturn(BaseDeDatos.estudiantes.stream()
                                .filter(e->e.getMatricula().equals("123"))
                                .findFirst()
                                .get());
                //correcto se lanza la excepcion
                assertThrows(CreditosMenores.class, ()->{
                        estudianteRespositorio.findById("123").getMaterias().get(0)
                        .setCreditos(-11);
                });
                //correcto, nunca se ejecuta
                //verify(estudianteRespositorio,never()).findAll();

                //correcto, nunca se ejecuta una sola vez.
                //verify(materiaRepositorio,times(0)).save(any(Materia.class));

                //simulando insertar
                //materiaRepositorio.save(any(Materia.class));
                //comprobar cuantas veces se llama
                //verify(materiaRepositorio,times(1)).save(any());

        }

        @Test
        void testCuatro() {
                when(estudianteRespositorio.findById("124")).thenReturn(BaseDeDatos.estudiantes.stream()
                                .filter(e->e.getMatricula().equals("124"))
                                .findFirst()
                                .get());
                Estudiante estudianteUno=estudianteServicio.findById("124");
                Estudiante estudianteDos=estudianteServicio.findById("124");

                //verificar que sea la misma instancia (dos maneras distintas)
                //assertSame(estudianteUno,estudianteDos);
                //assertTrue(estudianteUno==estudianteDos);

                //assertEquals(estudianteUno.getNombre(), estudianteDos.getNombre());-
                //verify(estudianteRespositorio,times(3)).findById("124");
        }
}
```

10. Clase EstudianteServicioImplTestDos

Copiar y pegar la clase EstudianteServicioImplTest, cambiando el nombre ha EstudianteServicioImplTestDos, en el mismo paquete. Hacer los siguientes cambios.

```
package dgtic.pruebas;
import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.context.annotation.ComponentScan;

import dgtic.core.excepciones.CreditosMenores;
import dgtic.core.modelo.Estudiante;
import dgtic.core.modelo.Materia;
import dgtic.core.repositorio.BaseDeDatos;
import dgtic.core.repositorio.intf.EstudianteRepositorio;
import dgtic.core.repositorio.intf.MateriaRepositorio;
import dgtic.core.servicio.EstudianteServicio;
import dgtic.core.servicio.EstudianteServicioImpl;
@SpringBootTest(classes = {EstudianteServicioImplTestDos.class})
@ComponentScan(basePackages = "dgtic.core")
class EstudianteServicioImplTestDos {
		@Mock
		private EstudianteRepositorio estudianteRespositorio;
		@Mock
		private MateriaRepositorio materiaRepositorio;
		@InjectMocks
		private EstudianteServicioImpl estudianteServicio;

		@Test
		void testUno() {
		.
		.
		}
.
.
.
}
```

11. Clase EstudianteServicioImplTestTres

Copiar y pegar la clase EstudianteServicioImplTestDos, cambiando el nombre ha EstudianteServicioImplTestTres, en el mismo paquete. Hacer los siguientes cambios.

```
package dgtic.pruebas;
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertThrows;
import static org.mockito.ArgumentMatchers.anyString;
import static org.mockito.Mockito.when;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.context.annotation.ComponentScan;
import dgtic.core.excepciones.CreditosMenores;
import dgtic.core.modelo.Estudiante;
import dgtic.core.repositorio.BaseDeDatos;
import dgtic.core.repositorio.intf.EstudianteRepositorio;
import dgtic.core.repositorio.intf.MateriaRepositorio;
import dgtic.core.servicio.EstudianteServicio;
import dgtic.core.servicio.EstudianteServicioImpl;


@SpringBootTest(classes = {EstudianteServicioImpl.class})
@ComponentScan(basePackages = "dgtic.core")
class EstudianteServicioImplTestTres {

		@MockBean
		private EstudianteRepositorio estudianteRespositorio;
		@MockBean
		private MateriaRepositorio materiaRepositorio;
		@Autowired
		private EstudianteServicio estudianteServicio;

		@Test
		void testUno() {
		.
		.
		}
}
```