Universidad Nacional Autónoma de México

Dirección General de Cómputo de Tecnologías de Información y Comunicación

Diplomado

Desarrollo de Sistemas con Tecnología Java

Ejercicio Trece

“POJO, @Autowired”

Mtro. ISC. Miguel Ángel Sánchez Hernández

Tabla de contenido

1. Copiar un proyecto en IntelliJ IDEA

Para copiar un proyecto, hacemos lo siguiente:

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

2. Borrado de Archivos

Borrar los archivos de las carpetas (**no las carpetas**):

- dgtic.core.modelo
- dgtic.core.servicio
- dgtic.core.xml

3. 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;
public class Estudiante {
		private String matricula;
		private String nombre;
		private int edad;
		private List<Materia> materias=new ArrayList<>();
		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 = Arrays.asList(materias);
		}
		@Override
		public String toString() {
			return "Estudiante [matricula=" + matricula + ", nombre=" + nombre + ", edad=" + edad + ", materias=" + materias
					+ "]";
		}
}
```

4. Clase Materia

Crear la clase Materia con los siguientes datos:

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

```
package dgtic.core.modelo;
public class Materia {
		private String nombre;
		private Integer creditos;
		public Materia(String nombre, Integer creditos) {
			super();
			this.nombre = nombre;
			this.creditos = creditos;
		}
		public String getNombre() {
			return nombre;
		}
		public void setNombre(String nombre) {
			this.nombre = nombre;
		}
		public Integer getCreditos() {
			return creditos;
		}
		public void setCreditos(Integer creditos) {
			this.creditos = creditos;
		}
		@Override
		public String toString() {
			return "Materia [nombre=" + nombre + ", creditos=" + creditos + "]";
		}
}
```

5. Clase BaseDeDatos

Crear la clase BaseDeDatos con los siguientes datos:

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

```
package dgtic.core.repositorio;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import dgtic.core.modelo.Estudiante;
import dgtic.core.modelo.Materia;
public class BaseDeDatos {
		public static Map<String,List<Estudiante>> carreras=new HashMap<>();
		static {
				List<Estudiante> estudiantes=new ArrayList<>();
				Estudiante est=new Estudiante("123","Rosa",20);
				est.setMaterias(new Materia("Cálculo",9),new Materia("Programación",10),
						new Materia("Lógica",10));
				estudiantes.add(est);
				est=new Estudiante("124","Tomas",22);
				est.setMaterias(new Materia("Programación",10),
						new Materia("Lógica",10));
				estudiantes.add(est);
				carreras.put("ico",estudiantes);
				////////////////////////////////
				estudiantes=new ArrayList<>();
				est=new Estudiante("125","Mario",20);
				est.setMaterias(new Materia("Cálculo",9),new Materia("Circuitos Lógicos",10),
						new Materia("Lógica de Autómatas",10));
				estudiantes.add(est);
				est=new Estudiante("126","Esmeralda",22);
				est.setMaterias(new Materia("Circuitos Lógicos",10),
						new Materia("Lógica de Autómatas",10));
				estudiantes.add(est);
				carreras.put("ime", estudiantes);
		}
}
```

6. Interfaz BaseDeDatosDAO

Crear la interfaz BaseDeDatosDAO con los siguientes datos:

- Nombre de la interfaz: BaseDeDatosDAO
- Paquete: dgtic.core.repositorio.intf

```
package dgtic.core.repositorio.intf;
import java.util.List;
import dgtic.core.modelo.Estudiante;
public interface BaseDeDatosDAO {
		public List<Estudiante> getEstudiantes(String carrera);
		public Estudiante getEstudiante(String carrera,String matricula);
}
```

7. Clase BaseDeDatosDAOImpl

Crear la clase BaseDeDatosDAOImpl con los siguientes datos:

- Nombre de la clase: BaseDeDatosDAOImpl
- Paquete: dgtic.core.repositorio.impl

```
package dgtic.core.repositorio.impl;
import java.util.List;
import org.springframework.stereotype.Component;
import dgtic.core.modelo.Estudiante;
import dgtic.core.repositorio.intf.BaseDeDatosDAO;
import dgtic.core.repositorio.BaseDeDatos;
@Repository("baseDeDatosDAO")
public class BaseDeDatosDAOImpl implements BaseDeDatosDAO {
		@Override
		public List<Estudiante> getEstudiantes(String carrera) {
			return BaseDeDatos.carreras.get(carrera);
		}
		@Override
		public Estudiante getEstudiante(String carrera, String matricula) {
			return BaseDeDatos.carreras.get(carrera)
					.stream().filter(est->est.getMatricula().equals(matricula))
					.findFirst()
					.get();
		}
}
```

8. Modificar la clase Inicio (versión 1)

Modificar la clase Inicio como sigue:

```
package dgtic.core;
import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import dgtic.core.modelo.Estudiante;
import dgtic.core.repositorio.intf.BaseDeDatosDAO;
import dgtic.core.repositorio.impl.BaseDeDatosDAOImpl;
public class Inicio {
	public static void main(String[] args) {
		ConfigurableApplicationContext contexto =
				new AnnotationConfigApplicationContext(BaseDeDatosDAOImpl.class);

		BaseDeDatosDAO serv=(BaseDeDatosDAO) contexto.getBean("baseDeDatosDAO");
		System.out.println(serv.getEstudiantes("ico"));
		System.out.println("------");
		Estudiante est=serv.getEstudiante("ime", "126");
		System.out.println(est);
		contexto.close();
	}
}
```

Correr la aplicación. Para el siguiente paso vamos a ocupar @Autowired, para que realice la Autoconexión.

9. Clase ServicioDAO

Crear la clase ServicioDAO con los siguientes datos:

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

```
package dgtic.core.servicio;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.stereotype.Component;
import dgtic.core.repositorio.intf.BaseDeDatosDAO;
import java.util.stream.Collectors;
@Service
public class ServicioDAO {
		@Autowired
		private BaseDeDatosDAO servicioDAO;

		public String archivoCSV(String carrera) {
			return servicioDAO.getEstudiantes(carrera).stream()
			.map(alm->alm.getMatricula()+";"+
					(alm.getMaterias().stream()
					.map(mat->(mat.getNombre()+";"+mat.getCreditos()))
					.collect(Collectors.joining(";")))+";"+alm.getNombre())
			.collect(Collectors.joining("\n"));
		}
}
```

10. Modificar la clase Inicio (versión 2)

Modificar la clase Inicio como sigue:

```
package dgtic.core;
import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import dgtic.core.modelo.Estudiante;
import dgtic.core.repositorio.intf.BaseDeDatosDAO;
import dgtic.core.servicio.ServicioDAO;
public class Inicio_v2 {
	public static void main(String[] args) {
		ConfigurableApplicationContext contexto =
				new AnnotationConfigApplicationContext(“dgtic.core”);

		BaseDeDatosDAO serv=(BaseDeDatosDAO) contexto.getBean("baseDeDatosDAO");
		System.out.println(serv.getEstudiantes("ico"));
		System.out.println("------");
		Estudiante est=serv.getEstudiante("ime", "126");
		System.out.println(est);
		System.out.println("------");
		ServicioDAO csv=contexto.getBean(ServicioDAO.class);
		System.out.println(csv.archivoCSV("ime"));
		System.out.println("------");
		contexto.close();
	}
}
```

Correr la aplicación. Ocuparemos @Autowired en un método.

11. Modificar la clase ServicioDAO

Modificamos la clase ServicioDAO como sigue:

```
package dgtic.core.servicio;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.stereotype.Component;
import dgtic.core.repositorio.intf.BaseDeDatosDAO;
import java.util.stream.Collectors;
@Service
public class ServicioDAO {
		private BaseDeDatosDAO servicioDAO;
		@Autowired(required = false)
		public void setServicioDAO(BaseDeDatosDAO servicioDAO) {
			this.servicioDAO=servicioDAO;
		}

		public BaseDeDatosDAO getServicioDAO() {
			return servicioDAO;
		}
		public String archivoCSV(String carrera) {
			return servicioDAO.getEstudiantes(carrera).stream()
			.map(alm->alm.getMatricula()+";"+
					(alm.getMaterias().stream()
					.map(mat->(mat.getNombre()+";"+mat.getCreditos()))
					.collect(Collectors.joining(";")))+";"+alm.getNombre())
			.collect(Collectors.joining("\n"));
		}
}
```

Nota: @Autowired(required=false), indica a Spring sin no puede encontrar un bean que coincida, entonces la propiedad no se establece.

12. Modificar la clase Inicio (versión 3)

Modificar la clase Inicio como sigue:

```
package dgtic.core;
import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import dgtic.core.modelo.Estudiante;
import dgtic.core.repositorio.intf.BaseDeDatosDAO;
import dgtic.core.servicio.ServicioDAO;
public class Inicio_v3 {
	public static void main(String[] args) {
		ConfigurableApplicationContext contexto =
				new AnnotationConfigApplicationContext(“dgtic.core”);

		ServicioDAO csv=contexto.getBean(ServicioDAO.class);
		BaseDeDatosDAO serv=csv.getServicioDAO();
		System.out.println(serv.getEstudiantes("ico"));
		System.out.println("------");
		Estudiante est=serv.getEstudiante("ime", "126");
		System.out.println(est);
		System.out.println("------");
		System.out.println(csv.archivoCSV("ime"));
		System.out.println("------");
		contexto.close();
	}
}
```

Correr la aplicación. Ocuparemos @Autowired en un constructor.

13. Modificar la clase ServicioDAO

Modificamos la clase ServicioDAO como sigue:

```
import org.springframework.stereotype.Component;
import dgtic.core.repositorio.intf.BaseDeDatosDAO;
import java.util.stream.Collectors;
@Service
public class ServicioDAO {
		private BaseDeDatosDAO servicioDAO;
		@Autowired
		public ServicioDAO(BaseDeDatosDAO servicioDAO) {
			this.servicioDAO = servicioDAO;
		}

		public void setServicioDAO(BaseDeDatosDAO servicioDAO) {
			this.servicioDAO = servicioDAO;
		}

		public BaseDeDatosDAO getServicioDAO() {
			return servicioDAO;
		}

		public String archivoCSV(String carrera) {
			return servicioDAO.getEstudiantes(carrera).stream()
			.map(alm->alm.getMatricula()+";"+
					(alm.getMaterias().stream()
					.map(mat->(mat.getNombre()+";"+mat.getCreditos()))
					.collect(Collectors.joining(";")))+";"+alm.getNombre())
			.collect(Collectors.joining("\n"));
		}
}
```

Nota: @Autowired(required=false), no se indica dado que Spring necesita inyectar el bean cuando se ocupa en el constructor. Correr la aplicación.

No tenemos problema en la inyección del bean, como solo tenemos uno que coincide para la Autoconexión, pero que pasa si Spring encuentra dos, para resolverlo ocuparemos @Primary y @Qualifier.

14. Copiar y crear la clase BaseDeDatosExtraDaoImp

En el paquete dgtic.repositorio.impl, copiar el archivo BaseDeDatosDAOImpl.java y pegarlo en el mismo paquete, con el nombre de BaseDeDatosExtraDAOImpl.java. la línea @Repository("baseDeDatosDAO"), cambiarla por @Component("baseDeDatosDAOExtra").

Ahora ya tenemos dos clases que implementan la misma interfaz, por lo que Spring no sabe cuál inyectar. Correr la Aplicación. Tenemos una excepción:

ADVERTENCIA: Exception encountered during context initialization - cancelling refresh attempt: org.springframework.beans.factory.UnsatisfiedDependencyException

Para resolver el problema, ocupamos @Primary, indicando que clase queremos que se inyecte porque le estamos dando preferencia.

15. Modificar la clase BaseDeDatosExtraDAOImpl

Modificar la clase BaseDeDatosExtraDAOImpl como sigue:

```
package dgtic.core.repositorio.impl;
import java.util.List;
import org.springframework.context.annotation.Primary;
import org.springframework.stereotype.Component;
import dgtic.core.modelo.Estudiante;
import dgtic.core.repositorio.BaseDeDatos;
import dgtic.core.repositorio.intf.BaseDeDatosDAO;
@Repository("baseDeDatosDAOExtra")
@Primary
public class BaseDeDatosExtraDAOImpl implements BaseDeDatosDAO {
		@Override
		public List<Estudiante> getEstudiantes(String carrera) {
			return BaseDeDatos.carreras.get(carrera);
		}
		@Override
		public Estudiante getEstudiante(String carrera, String matricula) {
			return BaseDeDatos.carreras.get(carrera)
					.stream().filter(est->est.getMatricula().equals(matricula))
					.findFirst()
					.get();
		}
}
```

Correr la aplicación. Si nosotros queremos indicar el bean que se inyectara, lo podemos hacer por medio de su nombre que se asigna, para eso hacemos el uso de la anotación @Qualifier

16. Modificar de nuevo la clase BaseDeDatosExtraDAOImpl

Modificar la clase BaseDeDatosExtraDAOImpl como sigue:

```
package dgtic.core.repositorio.impl;
import java.util.List;
import org.springframework.context.annotation.Primary;
import org.springframework.stereotype.Component;
import dgtic.core.modelo.Estudiante;
import dgtic.core.repositorio.BaseDeDatos;
import dgtic.core.repositorio.intf.BaseDeDatosDAO;
@Repository("baseDeDatosDAOExtra")
public class BaseDeDatosExtraDAOImpl implements BaseDeDatosDAO {
		@Override
		public List<Estudiante> getEstudiantes(String carrera) {
			return BaseDeDatos.carreras.get(carrera);
		}
		@Override
		public Estudiante getEstudiante(String carrera, String matricula) {
			return BaseDeDatos.carreras.get(carrera)
						.stream().filter(est->est.getMatricula().equals(matricula))
						.findFirst()
						.get();
		}
}
```

17. Modificar la clase ServicioDAO

Modificar la clase ServicioDAO como sigue:

```
package dgtic.core.servicio;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.stereotype.Component;
import dgtic.core.repositorio.intf.BaseDeDatosDAO;
import java.util.stream.Collectors;
@Service
public class ServicioDAO {
		private BaseDeDatosDAO servicioDAO;
		@Autowired
		public ServicioDAO(@Qualifier("baseDeDatosDAOExtra") BaseDeDatosDAO servicioDAO) {
			this.servicioDAO = servicioDAO;
		}

		public void setServicioDAO(BaseDeDatosDAO servicioDAO) {
			this.servicioDAO = servicioDAO;
		}

		public BaseDeDatosDAO getServicioDAO() {
			return servicioDAO;
		}
		public String archivoCSV(String carrera) {
			return servicioDAO.getEstudiantes(carrera).stream()
			.map(alm->alm.getMatricula()+";"+
					(alm.getMaterias().stream()
					.map(mat->(mat.getNombre()+";"+mat.getCreditos()))
					.collect(Collectors.joining(";")))+";"+alm.getNombre())
			.collect(Collectors.joining("\n"));
		}
}
```

Correr la aplicación. Si tuviéramos la inyección por propiedad en la clase ServicioDAO el cambio seria como sigue.

18. Si la inyección es por propiedad en la ServicioDAO

Modificar la clase ServicioDAO como sigue:

```
package dgtic.core.servicio;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.stereotype.Component;
import dgtic.core.repositorio.intf.BaseDeDatosDAO;
import java.util.stream.Collectors;
@Service
public class ServicioDAO {
		@Autowired
		@Qualifier("baseDeDatosDAOExtra")
		private BaseDeDatosDAO servicioDAO;

		public void setServicioDAO(BaseDeDatosDAO servicioDAO) {
			this.servicioDAO = servicioDAO;
		}

		public BaseDeDatosDAO getServicioDAO() {
			return servicioDAO;
		}

		public String archivoCSV(String carrera) {
			return servicioDAO.getEstudiantes(carrera).stream()
			.map(alm->alm.getMatricula()+";"+
					(alm.getMaterias().stream()
					.map(mat->(mat.getNombre()+";"+mat.getCreditos()))
					.collect(Collectors.joining(";")))+";"+alm.getNombre())
			.collect(Collectors.joining("\n"));
		}

}
```

19. Si la inyección es por método en la ServicioDAO

Modificar la clase ServicioDAO como sigue:

```
package dgtic.core.servicio;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.stereotype.Component;
import dgtic.core.repositorio.intf.BaseDeDatosDAO;
import java.util.stream.Collectors;
@Service
public class ServicioDAO {
		private BaseDeDatosDAO servicioDAO;
		@Autowired
		public void setServicioDAO(
					@Qualifier("baseDeDatosDAOExtra")BaseDeDatosDAO servicioDAO) {
			this.servicioDAO = servicioDAO;
		}

		public BaseDeDatosDAO getServicioDAO() {
			return servicioDAO;
		}

		public String archivoCSV(String carrera) {
			return servicioDAO.getEstudiantes(carrera).stream()
			.map(alm->alm.getMatricula()+";"+
						(alm.getMaterias().stream()
						.map(mat->(mat.getNombre()+";"+mat.getCreditos()))
						.collect(Collectors.joining(";")))+";"+alm.getNombre())
			.collect(Collectors.joining("\n"));
		}

}
```