Introducción

Tema 0. Acceso a datos

Resultados aprendizaje

Desarrollar aplicaciones que gestionan información almacenada en ficheros identificando el campo de aplicación de los mismos y utilizando clases específicas.

Desarrollar aplicaciones que gestionan información almacenada en bases de datos relacionales identificando y utilizando mecanismos de conexión.

Gestionar la persistencia de los datos identificando herramientas de mapeo objeto relacional (ORM) y desarrollando aplicaciones que las utilizan.

Resultados aprendizaje

Desarrollar aplicaciones que gestionan la información almacenada en bases de datos objeto relacionales y orientadas a objetos valorando sus características y utilizando los mecanismos de acceso incorporados.

Desarrollar aplicaciones que gestionan la información almacenada en bases de datos nativas XML evaluando y utilizando clases específicas.

Programar componentes de acceso a datos identificando las características que debe poseer un componente y utilizando herramientas de desarrollo.

Roadmap

1. Ficheros
2. JDBC
3. Hibernate
4. ObjectDB
5. BaseX
6. SpringBoot

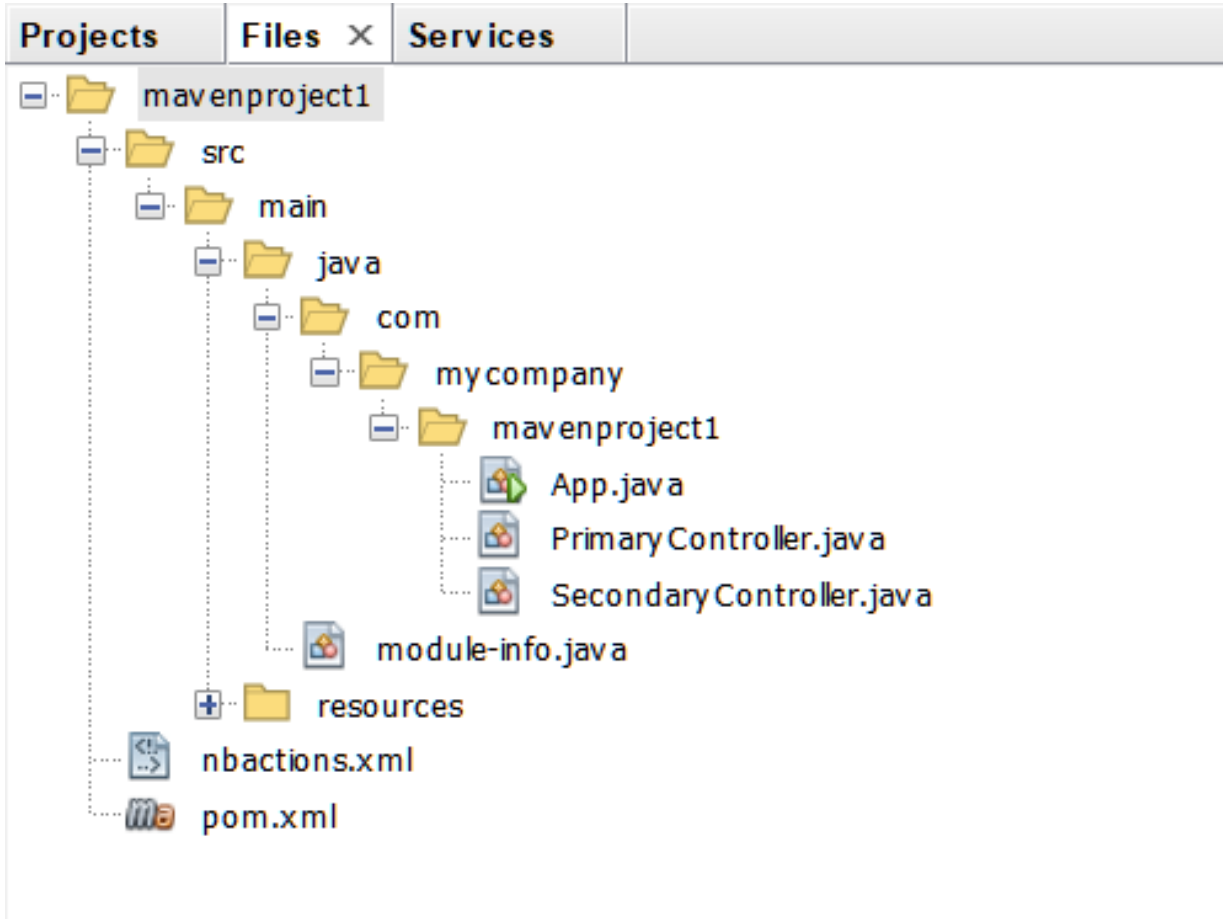
Configuración del entorno de desarrollo

- NetBeans 12
- JDK 16
- System.out.println("Hola Mundo")

Repositorios de código

Gestión de proyectos Java

Gestión de proyectos Java



Gestión de proyectos Java

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany</groupId>
    <artifactId>SwingSample</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>16</maven.compiler.source>
        <maven.compiler.target>16</maven.compiler.target>
    </properties>
```

Documentación de programas

```
/**
 *
 * @author frome
 */
public class NewJFrame extends javax.swing.JFrame {

    /**
     * Creates new form NewJFrame
     */
    public NewJFrame() {
        initComponents();
    }
```

POO

- Clases
- Propiedades
- Métodos
- Herencia (extends)
- Interfaces (implements)
- Templates <T>

Java.util

- ArrayList<T>
- HashMap<T,V>
- HashSet<T>
- Logging

ArrayList

.add() .remove()

.size()

.contains()

.get() .set()

.forEach((p) -> System.out.println(p));

HashSet

.add() .remove()

.size()

.contains()

.get() .set()

HashMap

.put() .get()

.remove()

.size()

.keySet() .values()

.containsKey() .containsValue()

Iteradores

Iterator<T> = .iterator()
.hasNext()
.next()
.remove()

¡¡¡ Iterar siempre con **while**, nunca con **for** !!!

Excepciones en Java

- try {…}
- catch(Exception e){…}
- finally {…}
- Throws

Logs

- **Logger**: objecto que se encarga de toda la gestión.
- **Log / record**: contenido que queremos registrar, organizados por niveles.
- **Handler**: indica donde y como almacenar todos los logs.

Logs

Niveles de registros de logs:

- SEVERE
- WARNING
- INFO
- CONFIG

```
private static Logger LOGGER = Logger.getLogger(NewMain.class.getName());

LOGGER.info("Creando la tabla");
```

Logs

```
static {
    try {
        System.setProperty("java.util.logging.SimpleFormatter.format",
                           "[%1$tF %1$tT] [%4$-7s] %5$s %n");
        LOGGER = Logger.getLogger(NewMain.class.getName());
        FileHandler fileTxt = new FileHandler("log.log");
        SimpleFormatter formatterTxt = new SimpleFormatter();
        fileTxt.setFormatter(formatterTxt);
        LOGGER.addHandler(fileTxt);
    } catch (IOException ex) {
        Logger.getLogger(NewMain.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SecurityException ex) {
        Logger.getLogger(NewMain.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```