# Errores prácticos

## *Repositorio Git*

1. COMO TENER VARIOS PROYECTOS EN EL REPOSITORIO GIT

# Práctica 1: GNU/Linux

## *Práctica 2: Web Estática*

### *1. POSIBLE FALLO AL EJECUTAR ECLIPSE PARA CREAR PAGINAS WEB ESTATICAS: Falta nodejs.*

Puede ser resuelto sencillamente mediante el siguiente comando: sudo apt install nodejs.

## *Práctica 3: Web Dinámica*

### *1. FALLO DE PUERTO OCUPADO EN EL SERVIDOR TOMCAT: Port already in use.*

Lo primero que deberemos hacer es usar el comando ps para consultar los procesos en uso. Podemos usar: ps aux | grep tomcat . Esto nos permitira ver solo el proceso Tomcat para no tener que buscar entre todos los procesos activos. Por ultimo usaremos el comando: kill [id] donde [id] sera el id correspondiente al proceso Tomcat.

## *Práctica 4: Servicios REST y Representación de Datos*

### *1. CREAR EL FICHERO WEB.XML EN UN PROYECTO EXISTENTE.*

En caso de no haber creado el fichero *web.xml* durante la creación del nuevo proyecto se puede realizar los siguientes pasos:
**Botón derecho** sobre el proyecto --> **Java EE Tools** --> **Generate Deployment Descriptor Stub**

## *Práctica 7: JPA*

### *0. Variables de tipo boolean en la base de datos*

Se debe elegir el tipo **tinyint(1)** en la base de datos a las variables que vayamos a referencias como boolean en java. Las claves foráneas hay que ponerlas en CASCADE. Para esto se puede cambiar las preferencias de MySQL desde el principio. Edit -> Preferences -> Modeling -> Default -> ON UPDATE/ON DELETE: CASCADE.

### *1. CONFIGURAR CONEXIÓN A BASE DE DATOS*

En el caso de que al habilitar el Facet de JPA no se ha configurado la conexión a la base de datos, se deberá ir a la pestaña de Data Source Explorer.

Ahí se hace click derecho en Database Connections y seleccionamos la opción New. En la nueva ventana se añadirá nuevo driver, seleccionando la versión 5.1.

Habrá que eliminar el .jar que tiene y añadir el mysql-connector-java (/usr->/share->/java->mysql...jar).

### *2. CREAR JPA ENTITIES*

- Click derecho al proyecto -> New -> JPA Entities From Tables
- Desactivar “List generated classes in persistence.xml”
- Quitar relaciones bidireccionales (deseleccionar las casillas de "Generate a reference to a collection of..." dentro del paso Table associations.
- Key generator: IDENTITY
- Entity Access: Property

### *3. ERROR IMPORTACIONES*

Comprobar que el nombre del servidor WildFly de la pestaña servers, coincide con "click derecho sobre el proyecto" -> Build Path -> Configure Build Path -> Server Library. En caso de que en éste último aparezaca "(unbound)", eliminar los servers y volver a añadirlos.

## *Apuntes mixtos: cód:Flndr*

1.Persistence.xml, standalone.xml y module.xml

2. Apuntes varios (Sesion, nombre del usuario actual, ResultCode, lista desplegable ...

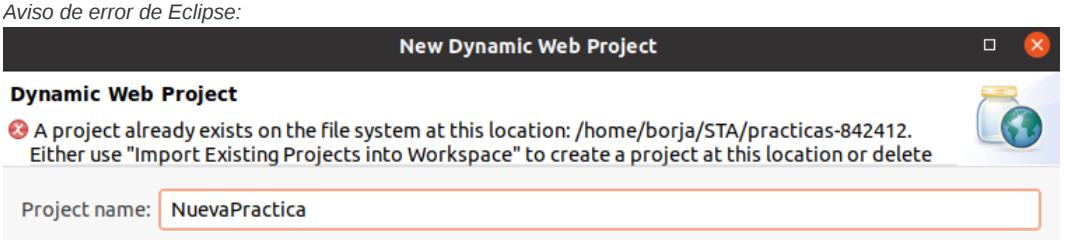
3. JPA --> REST

4. SEGURIDAD EN WEB.XML + STANDALONE

# 1. COMO TENER VARIOS PROYECTOS EN EL REPOSITORIO GIT

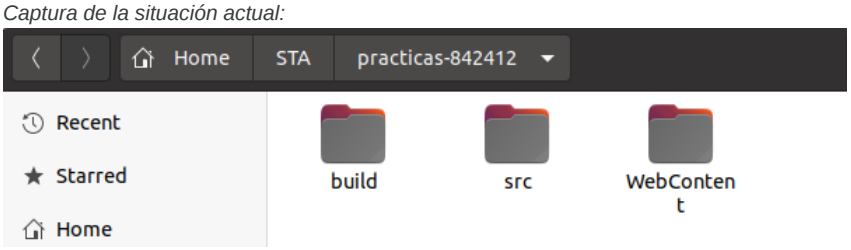
En caso de haber creado el proyecto directamente en la carpeta del repositorio NO se podrá crear más proyectos.

*Aviso de error de Eclipse:*




Esto es porque no se ha creado un directorio donde guardar cada proyecto.

*Captura de la situación actual:*



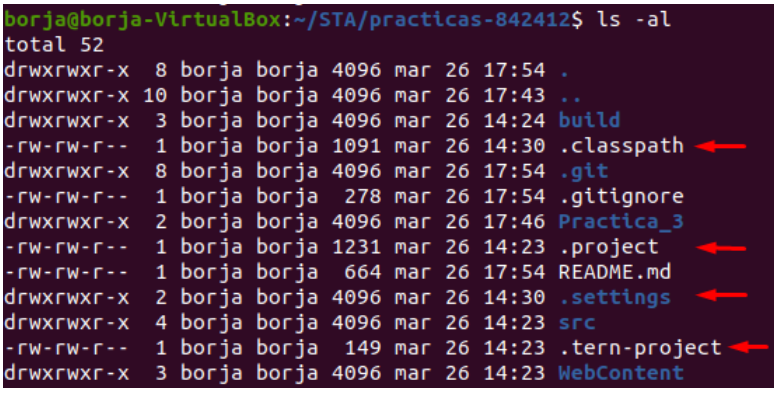

Para solucionarlo es necesario crear una carpeta (p.ej. “Practica_3”) donde almacenar todos los archivos propios del proyecto:

- build
- src
- WebContent
- .classpath
- .project
- .settings
- .tern-proyect

Los tres primeros se pueden arrastrar y soltar en la nueva carpeta, pero los 4 últimos no.

Eso es porque estos ficheros se encuentran en modo oculto (ya que les precede un punto), por lo que es necesario abrir el terminal y desde ahí mover estos ficheros a la nueva carpeta.
A continuación, se muestra los comandos necesarios:

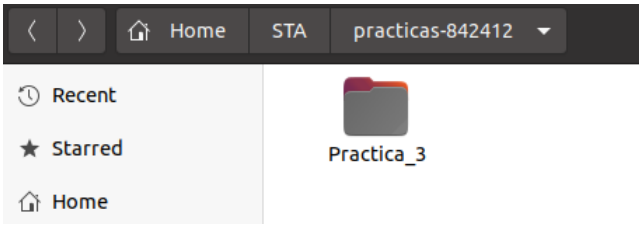
- Mostrar todos los ficheros del directorio: *$ ls -al*




- Mover los ficheros ocultos a la nueva carpeta:
  *$ mv .classpath .project .settings/ .tern-project Practica_3/*

(NOTA: No incluir ningún archivo de git ni el “README.md”)

*Captura del directorio corregido:*

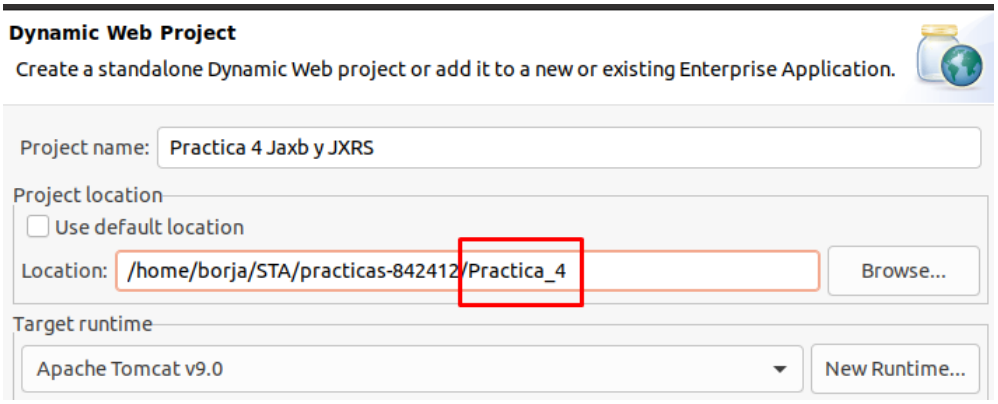



Después, en Eclipse:

1. Eliminar el proyecto que se ha trasladado. Para ello pulsar **botón derecho sobre el proyecto** --> **Delete** --> **OK** (No habilitar la opción de "Delete project contents on disk")
2. Pulsar **File** --> **Import** --> **Existing Projects into Workspace** --> **Browse**… --> seleccionar los projectos que contenga el directorio --> **Finish**

Finalmente, si se desea crear un nuevo proyecto que pertenezca al repositorio git hay que seleccionar correctamente su ubicación, es decir, creando de nuevo una nueva carpeta.

Por ejemplo: escribiendo "Practica_4" en el path de la ubicación del proyecto:




Para actualizar los cambios en GitHub hacer un **commit + push**.

Imprimir

# Práctica 1: GNU/Linux

## Instalación

### Máquina Virtual

A la hora de instalar los guest aditions puede salir un problema de que no se han instalado correctamente. Para resolver este problema se tiene que instalar los build essentials usando el comando:

```
sudo apt install build-essential
```

### Instalación de programas

## Administración

# 1.Persistence.xml, standalone.xml y module.xml

**PERSISTENCE.XML**

***Ubicación:*** *src/META-INF/persistence.xml*

***Contenido:***

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
/>            <persistence-unit name="ExamenSTA" transaction-type="JTA">
                <jta-data-source>java:jboss/datasources/ExamenSTA</jta-data-source>
        <properties>
                <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect"/>
                <property name="hibernate.hbm2ddl.auto" value="validate"/>
        </properties>
         </persistence-unit>
</persistence>
```

**STANDALONE.XML**

***Ubicación:*** *wildfly/standalone/configuration/standalone.xml*

***Contenido:***

```
<datasource jndi-name="java:jboss/datasources/ExamenSTA" pool-name="ExamenSTA" enabled="true">
<connection-url> jdbc:mysql://localhost:3306/ExamenSTA?useSSL=false</connection-url>
        <driver>com.mysql</driver>
        <security>
            <user-name>alumno</user-name>
            <password>sql</password>
         </security>
</datasource>
```

**MODULE.XML**

***Ubicación:*** *wildfly-19.1.0.Final/modules/system/layers/base/com/mysql/module.xml*

***Contenido:***

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
        <resources>
                <resource-root path="/usr/share/java/mysql-connector-java-8.0.24.jar"/>
        </resources>
        <dependencies>
                <module name="javax.api"/>
                <module name="javax.transaction.api"/>
                <module name="javax.servlet.api" optional="true"/>
        </dependencies>
</module>
```

# 2. Apuntes varios (Sesion, nombre del usuario actual, ResultCode, lista desplegable ...

**Plantilla "UTF-8"**: En head: <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

### Obtener nombre usuario actual:

*@Resource EJBContext ejbContext;*
En el método: *ejbContext.getCallerPrincipal().getName()*

### Obtener sesión: cookies, dir ip, time

**Get:**

```
public Sesion getSesion() {
if(sesion == null) {
ExternalContext context = FacesContext.getCurrentInstance().getExternalContext();
		HttpSession httpsesion = (HttpSession)context.getSession(false);
		String cookie = httpsesion.getId();
		String ip = ((HttpServletRequest)context.getRequest()).getRemoteAddr();
		Date time = new Date(httpsesion.getCreationTime());
		sesion = new Sesion(cookie, ip, time);
		}
return sesion;
}
```

**Terminar sesion:**

```
public String endSesion() {
		HttpSession sesion = (HttpSession) FacesContext
				.getCurrentInstance()
				.getExternalContext()
				.getSession(false);
		sesion.invalidate();
		return "paginaxhtml";

}
```

### ResultCode:

**En el EJB:**

```
public enum ResultCode {OK,USER_EXISTS,SHORT_PASSWD};
public ResultCode anade(Usuario usuario) {
		if(!em.createNamedQuery("Usuario.findAllNombre")
					.setParameter("nombre", usuario.getNombre())
					.getResultList().isEmpty())
				return ResultCode.USER_EXISTS;
		if(usuario.getPassword().length() < 3)
				return ResultCode.SHORT_PASSWD;
		em.persist(usuario);
		return ResultCode.OK;
}
```

**En el MB:**

```
private static final String[] mensaje = {"Usuario dado de alta correctamente",
					"ERROR: Usuario ya existe","ERROR: Contraseña demasiado corta"};
public void anade(Usuario u) {
		ResultCode code = admin.anade(u);
		if(code != ResultCode.OK) {
				FacesContext context = FacesContext.getCurrentInstance();
				context.addMessage(null, new FacesMessage(mensaje[code.ordinal()]));
		}
		usuarios = null;
}
```

# 3. JPA --> REST

AVISO! QUITAR TODO LO RELACIONADO CON SEGURIDAD DEL WEB.XML Y LAS ANOTACIONES: *@SecurityDomain, @RolesAllowed*

**En EJB:** @Path("admin")
**En MB:** private final WebTarget client = ClientBuilder.newClient().target(""">http://localhost:8080/ExamenSta/rest/admin");

| Rest Provider (EJB) | Rest Client (MB) |
|---|---|
| @Path(“usuariosLista”)<br>@**GET**<br>@Produces(MediaType.APPLICATION_JSON)<br>public List<UsuarioBL> getUsuarios(){} | usuarios = client.path("usuariosLista")<br>.request(MediaType.APPLICATION_JSON)<br>.**get**(new GenericType<List<UsuarioBL>>(){}); |
| @Path(“delete/{uid}”)<br>@**DELETE**<br>public int delete(@PathParam(“uid”) int uid){} | client.path("elimina")<br>.path(String.valueOf(uid))<br>.request().**delete**(); |
| @Path(“anade”)<br>@**PUT** //@POST<br>@Consumes(MediaType.APPLICATION_JSON)<br>public ResultCode anade(Usuario usuario){} | ResultCode code = client.path("anade")<br>.request(MediaType.TEXT_PLAIN)<br>.**put**(Entity.json(u),ResultCode.class); |

Lo que consuma o produzca obligará a poner la anotación @XMLRootElement en las clases correspondientes
(En este caso: UsuarioBL y Usuario)

*Si la tabla anterior se desconfigura se repite el mismo contenido a continuación:*

**Rest Provider (EJB)**
**y**
**Rest Client (MB)**

@Path(“usuariosLista”)
@**GET**
@Produces(MediaType.APPLICATION_JSON)
public List<UsuarioBL> getUsuarios(){}
y
usuarios = client.path("usuariosLista")
.request(MediaType.APPLICATION_JSON)
.**get**(new GenericType<List<UsuarioBL>>(){});

---

@Path(“delete/{uid}”)
@**DELETE**
public int delete(@PathParam(“uid”) int uid){}
y
client.path("elimina")
.path(String.valueOf(uid))
.request().**delete**();

---

@Path(“anade”)
@**PUT** //@POST
@Consumes(MediaType.APPLICATION_JSON)
public ResultCode anade(Usuario usuario){}
y
ResultCode code = client.path("anade")
.request(MediaType.TEXT_PLAIN)
.**put**(Entity.json(u),ResultCode.class);

# 4. SEGURIDAD EN WEB.XML + STANDALONE

**En web.xml:**

```
<login-config>
        <auth-method>BASIC</auth-method>
        <realm-name>ExamenSTA</realm-name>
</login-config>

<security-role>
        <description>Usuarios</description>
        <role-name>usuario</role-name>
</security-role>

<security-role>
        <description>Administradores</description>
        <role-name>admin</role-name>
</security-role>

<security-constraint>
        <display-name>Administradores</display-name>
        <web-resource-collection>
                <web-resource-name>AdminXHTML</web-resource-name>
                <url-pattern>/admin.xhtml</url-pattern>
        </web-resource-collection>
        <auth-constraint>
                <role-name>admin</role-name>
        </auth-constraint>
         <user-data-constraint>
                <transport-guarantee>CONFIDENTIAL</transport-guarantee>
         </user-data-constraint>
</security-constraint>

<security-constraint>
         <display-name>Usuarios</display-name>
        <web-resource-collection>
                <web-resource-name>UsuarioXHTML</web-resource-name>
                <url-pattern>/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
                <role-name>usuario</role-name>
                <role-name>admin</role-name>
        </auth-constraint>
         <user-data-constraint>
                <transport-guarantee>CONFIDENTIAL</transport-guarantee>
         </user-data-constraint>
</security-constraint>
```

**En standalone.xml (al final del archivo, en security-domains)**

```
<security-domain name="db-82" cache-type="default">

    <authentication>
      <login-module code="Database" flag="required">
        <module-option name="password-stacking" value="useFirstPass"/>
        <module-option name="dsJndiName" value="java:jboss/datasources/ExamenSta"/>
        <module-option name="principalsQuery" value="SELECT password FROM Usuario WHERE nombre = ?"/>
        <module-option name="rolesQuery" value="SELECT rol, 'Roles' FROM Usuario WHERE nombre = ?"/>
      </login-module>
    </authentication>
</security-domain>
```

Anotación para la lógica EJB: *@SecurityDomain*("**db-82**")