

PROGRAMACION ORIENTADA A OBJETOS II

PRACTICAS DE LABORATORIO

ESTA PRACTICA DE LABORATORIO DEBE SER REALIZADA DE MANERA INDIVIDUAL. LEA DETENIDAMENTE LAS INSTRUCCIONES, PUES EL NO SEGUIRLAS TAL COMO SE INDICA PUEDE RESULTAR EN UNA PRACTICA INCORRECTA. CUANDO LA PRÁCTICA SEA TERMINADA, ASEGURESE DE QUE CUMPLE CON TODOS LOS REQUERIMIENTOS SOLICITADOS Y DE QUE FUNCIONE CORRECTAMENTE ANTES DE ENTREGARLA.

PRACTICA 7. USO DE LA CLASE DataSource EN JDBC

Esta práctica tiene que ver con la API de JDBC en cuanto a la creación de tablas a través de la clase DataSource. Las instrucciones asumen que **NO ESTA USANDO USANDO UN IDE (COMO INTELLIJ IDEA)** y que el programa será compilado y ejecutado desde la línea de comandos, esto con el propósito de aprender a usar librerías externas desde la línea de comando.

Para esta práctica, se asume que todo el trabajo de la práctica 6 fue realizado exitosamente pues es necesario usar las tablas creadas por tal práctica, así como el driver para MySQL.

Para tener los archivos necesarios, deberá crear una copia local del repositorio remoto creado en Github al aceptar la tarea. Para ello, es necesario hacer los siguientes pasos:

- a) Entrar a la página cuyo URL les fue proporcionado al aceptar la tarea, en tal página dé click en el botón Code y copie el URL que aparece en el cuadro de texto de nombre **Clone with HTTPS** (comienza con https://)
- b) En una consola de Git Bash en Windows (o en una terminal en Linux o Mac), cree una carpeta donde quiera que se contengan sus prácticas del semestre (si es que aún no la has creado) y colócate en tal carpeta. La carpeta la puedes crear desde el Git Bash o terminal Linux/Mac usando el comando ``mkdir`` (o con el explorador de archivos de su sistema operativo) y en la consola de Git Bash o terminal de Linux/Mac te puedes cambiar a la carpeta mencionada usando el comando `cd`
- c) Clone el repositorio privado dando el comando **`git clone URL practica07`**
- d) (donde URL es el URL que copió en el paso a)
- e) Este comando creará dentro de la carpeta creada en el paso b) una subcarpeta de nombre **`practica07`** donde estará una copia local del repositorio remoto.
- f) Los comandos posteriores de git, compilación y ejecución tendrán que darse desde tal carpeta, por tanto, cámbiate a la carpeta usando **`cd practica07`**

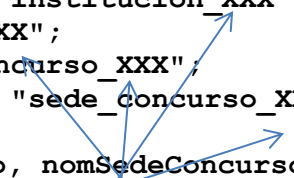
DEL PASO 1 AL PASO 7, SE DAN INDICACIONES DE LOS CAMBIOS NECESARIOS A REALIZAR EN EL ARCHIVO `PonDatosConDataSource.java` PARA QUE PUEDA SER EJECUTADO (MARCADOS CON UN `//TODO` EN EL ARCHIVO). DESPUÉS DE CADA PASO COMPILE PARA VERIFICAR QUE COMPILE PARA DETERMINAR SI NO HAY ERRORES Y SI NO MARCA ERRORES LA COMPILACIÓN HAGA UN `git add PonDatosConDataSource.java` Y UN `git commit -m "MENSAJE"` PARA REGISTRAR LOS CAMBIOS HECHOS ESPECIFICANDO DE MANERA CLARA Y CONCISA EN EL MENSAJE CUALES FUERON LOS CAMBIOS REALIZADOS. ELIMINE TAMBIÉN EL COMENTARIO `//TODO` DE LA PARTE QUE COMPLETÓ.

Paso 1. Sustituya en las cláusulas @author del comentario Javadoc de la clase **PonDatosConDataSource** las **NNNNNNNN** por su nombre completo y las **XXXXXXXXXX** por su matrícula:

```
/** ...
    @author NNNNNNNN
    @author XXXXXXXX
*/
```

Paso 2 Sustituya las XXX en los valores de los atributos de la clase PonDatosConDataSource:

```
private final String nomInstitucion = "institucion_XXX";
private final String nomSede = "sede_XXX";
private final String nomConcurso = "concurso_XXX";
private final String nomSedeConcurso = "sede_concurso_XXX";
private final String[] nomtablas =
    { nomInstitucion, nomSede, nomConcurso, nomSedeConcurso};
```



NOTA: En los valores de estas cuatro últimas constantes deben reemplazar XXX por su matrícula.

Paso 3. El método main de la clase PonDatosConDataSource tiene como trabajo verificar que se hayan pasado 4 argumentos que representan la dirección del servidor de base de datos MySQL, el nombre de la base de datos, así como el usuario y clave de acceso; y una vez hecha esta verificación carga el driver correspondiente, llama al método creaConexion y si tiene éxito, llamará a tres métodos para colocar datos en tres tablas. **Sustituya las NNNNNNNNN en el println de la última línea del main por su nombre completo:**

```
public static void main(String[] args) {
    ...
    System.out.println("PRACTICA DE NNNNNNNNN");
}
```

Paso 4. Complete el código del método creaConexion() en la clase PonDatosConDataSource, el cual creará conexión a la base de datos con los argumentos recibidos y regresará tal conexión si fue posible crearla, o generará una SQLException en caso de no haberla podido crear. El código se proporciona a continuación:

```
public Connection creaConexion(String direccionServidor,
                               String nomBD, String usuario,
                               String clave)
    throws SQLException {
    MysqlDataSource fuente = new MysqlDataSource();
    fuente.setServerName(direccionServidor);
    fuente.setUser(usuario);
    fuente.setPassword(clave);
    fuente.setDatabaseName(nomBD);
    return fuente.getConnection();
}
```

Paso 5. Complete el código del método llenaTablaSede() en la clase PonDatosConDataSource, el cual recibe un objeto que representa la conexión a la base de datos y tiene como trabajo insertar datos en la tabla sede_XXX (primero eliminando los datos existentes en tal tabla) demostrando como se hace a

través del uso de un objeto `ResultSet` actualizable. Los datos son tomados del archivo `datosSede.txt`. El código se proporciona a continuación (note que en algunas sentencias SQL involucradas en este paso y pasos posteriores puede haber espacios al inicio o fin de los strings entre las comillas, estos espacios son necesarios, de otra manera su programa no funcionará):

```
public void llenaTablaSede(Connection con) {
    System.out.println("Colocando datos en tabla sede:");
    try {
        Statement sentencia = con.createStatement();
        sentencia.executeUpdate("DELETE FROM "+nomSede);
        sentencia.executeUpdate("ALTER TABLE "+nomSede+
            " AUTO_INCREMENT=1");
        Scanner lector = new Scanner(new File("datosSede.txt"));
        lector.useLocale(Locale.US);
        //Delimitador de los datos en archivo
        lector.useDelimiter("[,\\n\\r]+");
        sentencia = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
            ResultSet.CONCUR_UPDATABLE);
        ResultSet resultado = sentencia.executeQuery("SELECT * FROM "
            + nomSede);

        String valor;
        while (lector.hasNext()) {
            resultado.moveToInsertRow();
            valor = lector.next();
            System.out.println("Agregando la sede "+valor+" ...");
            resultado.updateString("nombre_sede",valor);
            resultado.updateInt(3,lector.nextInt());
            valor = lector.next();
            resultado.updateString(4,valor);
            resultado.insertRow();
            resultado.last();
            idSedes.add(resultado.getInt("id_sede"));

        }
        lector.close();
        sentencia.close();
    } catch (IOException eio) {
        // Excepcion que se al abrir/leer el archivo de datos
        System.err.println("Error al cargar datos del archivo: " +
            eio.getMessage());
    } catch (SQLException exsql) {
        System.err.println("Error al ejecutar sentencias SQL: " +
            exsql.getMessage());
    }
} // fin del metodo llenaTablaSede()
```

Paso 6. Complete el código del método `llenaTablaConcurso()` en la clase `PonDatosConDataSource`, el cual recibe un objeto que representa la conexión a la base de datos y tiene como trabajo meter datos en la tabla `concurso_XXX` a través del uso sentencias SQL INSERT. El código se proporciona a continuación:

```

public void llenaTablaConcurso(Connection con) {
    String insertaDatos = "INSERT INTO " + nomConcurso +
        "(nombre_concurso, fecha_concurso, fecha_inicio_registro, " +
        " fecha_fin_registro) VALUES(";
    System.out.println("Colocando datos en la tabla concurso:");
    try {
        Statement sentencia = con.createStatement();
        sentencia.executeUpdate("DELETE FROM " + nomConcurso);
        sentencia.executeUpdate("ALTER TABLE " + nomConcurso +
            " AUTO_INCREMENT=1");
        System.out.println("Agregando 'Gran Premio' ...");
        String sql = insertaDatos + "'Gran Premio', '2020-10-31', " +
            "'2020-09-30', '2020-10-28')";
        sentencia.executeUpdate(sql,
            Statement.RETURN_GENERATED_KEYS);
        idConcurso = -1;

        ResultSet rs = sentencia.getGeneratedKeys();
        if (rs.next()) {
            idConcurso = rs.getInt(1);
        }

    } catch (SQLException ex) {
        System.err.println("***Error al ejecutar sentencias SQL: " +
            ex.getMessage());
    }
} // fin del metodo llenaTablaConcurso()

```

Paso 7. Complete el código del método `llenaTablaSedeConcurso()` en la clase `PonDatosConDataSource`, el cual recibe un objeto que representa la conexión a la base de datos y tiene como trabajo meter datos a la tabla `sede_concurso_XXX` a través del uso sentencias SQL INSERT. El código se proporciona a continuación:

```

public void llenaTablaSedeConcurso(Connection con) {
    System.out.println("Colocando datos en tabla sede_concurso:");
    String sql;
    try {
        Statement sentencia = con.createStatement();
        sentencia.executeUpdate("DELETE FROM " + nomSedeConcurso);
        sentencia.executeUpdate("ALTER TABLE " + nomSedeConcurso +
            " AUTO_INCREMENT=1");
        for (Integer idSede: idSedes) {
            sql = "INSERT INTO " + nomSedeConcurso +
                " SET id_sede=" + idSede + ", id_concurso=" + idConcurso;
            sentencia.executeUpdate(sql);
            System.out.printf("%d,%d\n", idSede, idConcurso);
        }
    }
    catch (SQLException ex) {
        System.err.println("***Error al ejecutar sentencias SQL: " +

```

```

        ex.getMessage() );
    }

} // fin del método llenaTablaSedeConcurso()

```

Paso 8. En este punto, ya está completo el código, compilamos nuevamente:

```
javac PonDatosConDataSource.java
```

Paso 9. Ejecutamos el programa con el siguiente comando:

```
java PonDatosConDataSource localhost IngSW UAZsw2020
```

La salida del programa debe ser algo como lo siguiente:

```

Conexion a MySQL exitosa!
Colocando datos en tabla sede:
Agregando la sede UAIE ...
Agregando la sede TIC-UTZAC ...
Colocando datos en la tabla concurso:
Agregando 'Gran Premio' ...
Colocando datos en tabla sede_concurso:
1,1
2,1
PRACTICA DE NNNNNNNNNN

```

Paso 10. Verifique entrando a la consola de MySQL que la base de datos controlconcursos_ad2020 contenga las tablas sede_XXX, concurso_XXX y sede_concurso_XXX y que cada una de ellas contengan los datos insertados por las sentencias SQL ejecutadas por el programa. (Nuevamente, recuerde que las XXX deben ser sustituidas por su matrícula)

Paso 11. Haga un `git push` para subir los cambios a su repositorio privado

Recuerda que cada vez que hagas **git push** se realizarán automáticamente pruebas sobre tu programa para verificar si funciona correctamente. Verifica que las pruebas que se hacen no marquen errores. Recuerda que en la página de tu repositorio en la sección **Pull Requests**, se encuentra una subsección de nombre **Feedback**, donde podrás encontrar los resultados de las pruebas en la pestaña denominada **Check** (expandiendo la parte que dice **Run education/autograding@v1**), y cualquier comentario general que el profesor tenga sobre tu código en la pestaña **Conversation**.