Projecte UF6

Repositori: https://github.com/josepaltadill/MP6UF4Pr3

Index

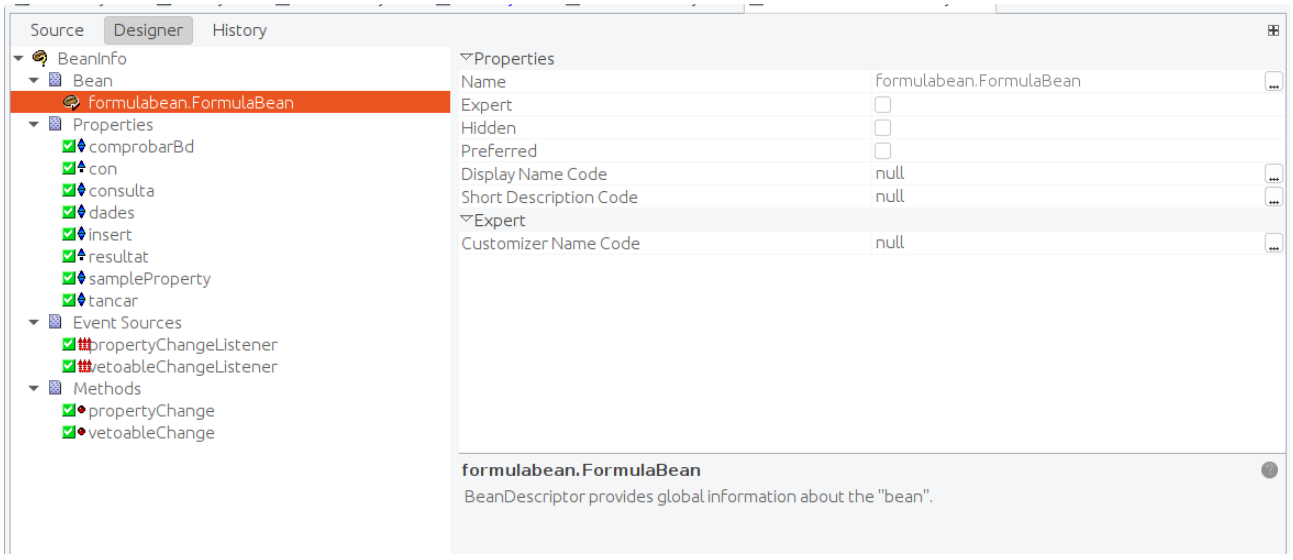
Índex de continguts

Projecte

Per al projecte em baso en el projecte de la UF5.
En aquest cas suprimeixo les pantalles de posar un nom al fitxer i de login.

Pt2 - JavaBean

Llistat de propietats del BeanInfo:



Les propietats les he afegit visibles i la distribució és la següent.

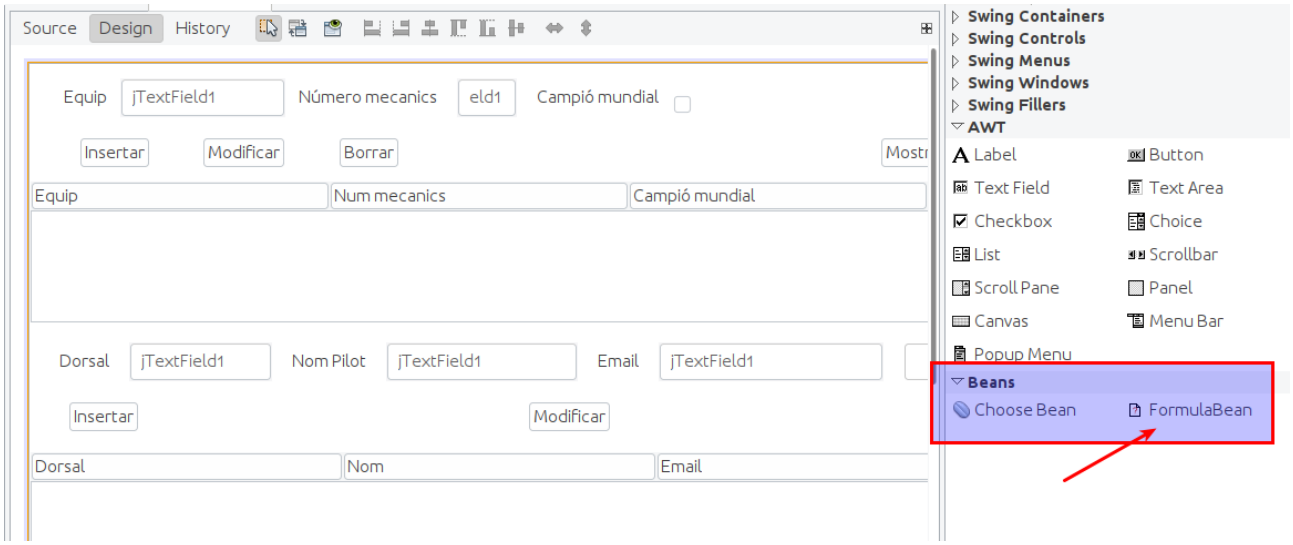
Propietats simples

- private Connection con = null;
- private ResultSet resultat;
- Properties datos;

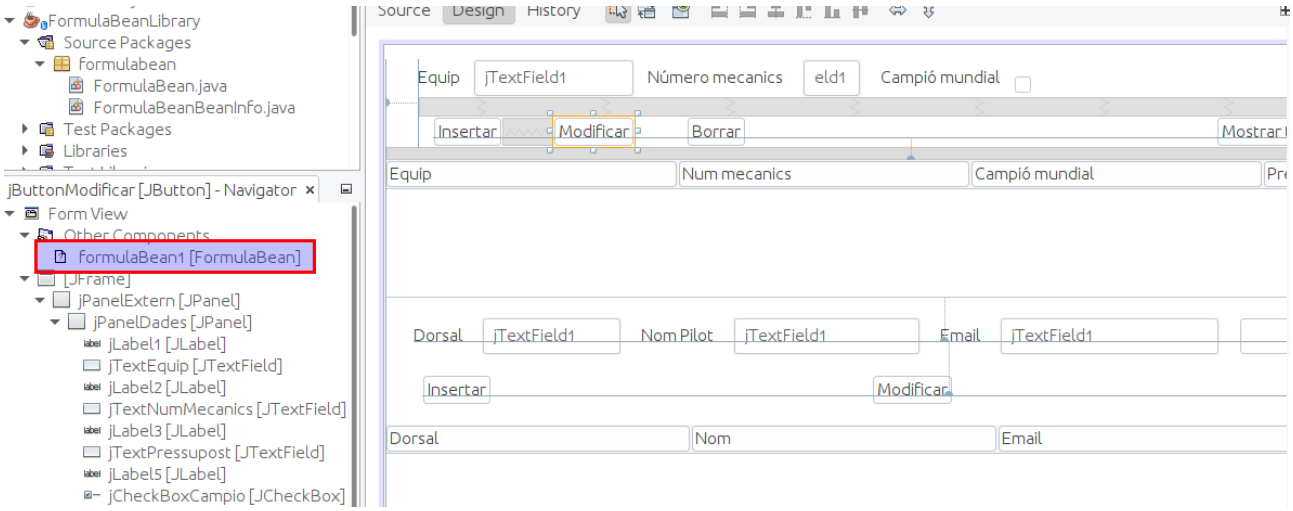
Propietats restringides

- private String dades;
- private String consulta;
- private String insert;
- private String tancar;
- private String comprobarBd;

Afegeixo el JavaBean a la paleta



I l'afegeixo a la vista.



PT3

He creat un fitxer «dades.properties» a l'arrel de la carpeta «MP6UF4Projecte» on he afegit 4 propietats enlloc de 3 ja que també tenim que afegir una url de connexió directament a l'arrel del mysql per poder crear la base de dades en cas que no s'hagi creat abans quedant de la següent manera:

url1=jdbc:mysql://localhost/formula1 → accés a la base de dades formula1 una vegada creada.

url2=jdbc:mysql://localhost → accés a l'arrel del mysql si no està la bd formula1

user=josep → usuari d'accés al mysql

password=josep → contrasenya d'accés al mysql

Controlador

Al controlador només faig un canvi i és que ara al arrancar l'aplicació el primer que fa és comprovar si la base de dades formula1 existeix mitjançant el mètode model.comprobarBd();

```
private void controlador() {
    model.comprobarBd();
    model.conexion();

    itemsEnBlanc();
    carregarTaula();
```

Model i JavaBean

Comprovar la bd

Aquest mètode és nou i com he explicat abans s'encarrega de controlar si la bd està creada cada vegada que posem en marxa l'aplicació, en cas de no estar-ho la crea juntament amb les seues taules i afegeix unes dades de mostra.

Primer fa una crida al mètode del JabaBean «bean.setComprobarBd("dades.properties");» passant-li el nom del fitxer com a string i dintre el JavaBen tenim les «PropertyChangeEvent» de les propietats restringides que s'encarrega de vigilar que l'string sigui correcte, si és així connecta a la bd i sinó retorna una « PropertyVetoException» per a que tornem a accedir a l'aplicació.

```
@Override
public void vetoableChange(PropertyChangeEvent evt) throws PropertyVetoException {
    switch (evt.getPropertyName()) {
        case FormulaBean.PROP_COMPROBARBD:
            datos = new Properties();
            try {
                datos.load(new FileInputStream((String) evt.getNewValue()));
                String url1 = (String)datos.get("url1");
                String url2 = (String)datos.get("url2");
                String user = (String)datos.get("user");
                String password = (String)datos.get("password");

                con = DriverManager.getConnection(url2, user, password);
                System.out.println("Conectat en exit");

            } catch (IOException | SQLException ex) {
                throw new PropertyVetoException("Error", evt);
            }
            break;
```

Si la connexió és correcta el mètode «comprobarBd» llança una consulta a la bd a través del JavaBean «bean.setConsulta("show databases");». Aquest té una propietat consulta encarregada de retornar totes les consultes que es fan a la bd i que retorna una «PropertyVetoException» en cas d'haver algo incorrecte, a la vegada controlem que l'error no sigui d'escriptura a la consulta error 1064.

```
case FormulaBean.PROP_CONSULTA:
    try {
        String SQL = evt.getNewValue().toString();
          System.out.println(SQL);
        Statement st;
        st = con.createStatement();
        ResultSet rs = st.executeQuery(SQL);
        resultat = rs;
    } catch (SQLException ex) {
        if (ex.getErrorCode() == 1064) {
            throw new PropertyVetoException("Hi ha un error d'escriptura en la consulta SQL", evt);
        } else {
            throw new PropertyVetoException(ex.toString(), evt);
        }
    }
    break;
```

Amb el resultat de la consulta dintre el mètode «comprobarBd» comprovo si la bd formula1 existeix, en cas de no existir la crea i li afegeix les taules i dades de mostra.

```
public void comprobarBd() {
    try {
        bean.setComprobarBd("dades.properties");
        bean.setConsulta("show databases");
        boolean existeix = false;
        ResultSet rs = bean.getResultat();
        try {
            while (rs.next()) {
                if (rs.getString(1).equalsIgnoreCase("formula1")) {
                    existeix = true;
                    System.out.println("La base de dades " + rs.getString(1) + " existeix ");
                }
            }
            if (!existeix) {
                System.out.println("creo base dades");
                bean.setInsert("CREATE DATABASE IF NOT EXISTS formula1");
                bean.setInsert("USE formula1;");
                bean.setInsert("CREATE TABLE IF NOT EXISTS equip("
                        + "nom varchar(40) NOT NULL,"
                        + "numMecanics int DEFAULT NULL,"
                        + "pressupost double DEFAULT NULL,"
                        + "campioMundial tinyint(1) DEFAULT NULL,"
                        + "PRIMARY KEY (nom));");
                bean.setInsert("CREATE TABLE IF NOT EXISTS pilot("
                        + "dorsal int NOT NULL,"
                        + "nom varchar(40) DEFAULT NULL,"
                        + "email varchar(40) DEFAULT NULL,"
                        + "nom_equipf1 varchar(40) DEFAULT NULL,"
                        + "PRIMARY KEY (dorsal),"
                        + "CONSTRAINT pilot_ibfk_1 FOREIGN KEY (nom_equipf1) REFERENCES equip (nom) ON DELETE SET NULL ON UPDATE CASCADE);");
                bean.setInsert("INSERT INTO equip VALUES ('Ferrari',28,1234567,0),('Mercedes',13,1234556,0),('Renault',25,1234568,0);");
                bean.setInsert("INSERT INTO pilot VALUES (1,'pilot4','email4@emai.com','Ferrari'),(2,'pilot32','email@emai.com','Ferrari'),(3,'pilot3','emai3@email.com','Mercedes'),(4,'pil
            }
        } catch (SQLException ex) {

        }
    } catch (PropertyVetoException ex) {
        JOptionPane.showMessageDialog(null, ex.getMessage());
        System.exit(0);
    }
}
```

Connexió a la bd

Tenim el mètode conexion() al model que modifica la propietat del bean passant-li un string amb el nom del fitxer de propietats «bean.setDades()». Aquesta al ser una propietat restringida passa pel mètode «vetoableChange» i si es correcta mitjançant les dades que compté el fitxer connecta a la base de dades, si no llença una «PropertyVetoException».

```
case FormulaBean.PROP_DADES:
    datos = new Properties();
    try {
        datos.load(new FileInputStream((String) evt.getNewValue()));
        String url1 = (String)datos.get("url1");
        String url2 = (String)datos.get("url2");
        String user = (String)datos.get("user");
        String password = (String)datos.get("password");

        con = DriverManager.getConnection(url1, user, password);
        System.out.println("Conectat en exit");

    } catch (IOException | SQLException ex) {
        throw new PropertyVetoException("Error", evt);
    }
    break;
```

Si no hi ha error al model assignem la nova connexió: «con=bean.getCon().

```
public void conexion() {
    try {
        bean.setDades("dades.properties");
        JOptionPane.showMessageDialog(null, "Acces en un nom de fitxer correcte");
    } catch (PropertyVetoException ex) {
        JOptionPane.showMessageDialog(null, "Hi ha alguna dada incorrecta. Torna a accedir a l'aplicació");
        System.exit(0);
    }
    con = bean.getCon();
}
```

Una vegada connectats el controlador executa el mètode carregarTaula() que a la vegada crida la mètode del model mostrarDades(). Aquest últim modifica la propietat «consulta» del bean que està dintre un « PropertyChangeEvent» enviant un select en format d'string, si la consulta és correcta el resultat es guarda a la propietat del bean «resultat» i es capturada desde el model per omplir els equips i pilots. Si no és correcta tirem una «PropertyVetoException».

```java
public void mosrtrarDades() {
    pilots.removeAll(pilots);
    equips.removeAll(equips);

    try {
        bean.setConsulta("select * from equip;");
        ResultSet rs = bean.getResultat();

        while (rs.next()) {
            String nom = rs.getNString("nom");
            int numMecanics = rs.getInt("numMecanics");
            double pressupost = rs.getDouble("pressupost");
            boolean campioMundial = rs.getBoolean("campioMundial");
            EquipF1 e = new EquipF1(nom, numMecanics, pressupost, campioMundial);
            this.<EquipF1>insertar(e, equips);
        }

        bean.setConsulta("select * from pilot;");
        ResultSet rsEquip = bean.getResultat();

        while (rsEquip.next()) {
            int dorsal = rsEquip.getInt("dorsal");
            String nom = rsEquip.getNString("nom");
            String email = rsEquip.getNString("email");
            String nom_equipf1 = rsEquip.getNString("nom_equipf1");

            Pilot a = new Pilot(dorsal, nom, email, nom_equipf1);
            this.<Pilot>insertar(a, pilots);
        }
    } catch (PropertyVetoException ex) {
        JOptionPane.showMessageDialog(null, "Error en la consulta SQL");
    } catch (SQLException ex) {
        System.out.println(ex.toString());
    }
}
```

```java
case FormulaBean.PROP_CONSULTA:
    try {
        String SQL = evt.getNewValue().toString();
          System.out.println(SQL);
        Statement st;
        st = con.createStatement();
        ResultSet rs = st.executeQuery(SQL);
        resultat = rs;
    } catch (SQLException ex) {
        if (ex.getErrorCode() == 1064) {
            throw new PropertyVetoException("Hi ha un error d'escriptura en la consulta SQL", evt);
        } else {
            throw new PropertyVetoException(ex.toString(), evt);
        }
    }
    break;
```

Inserir

A l'hora d'inserir, al model generem una consulta en forma d'string que l'enviem al bean modificant la seva propietat «insert». Aquesta l'utilitzarem tant per fer els inserts, els updates i els delete ja que no necessitem guardar

cap resultSet.

Inserir equips

```
public void insertarEquipsBd(String nom, int numMecanics, double pressupost, boolean campioMundial) throws Excepcio {
    if (nom.trim().isEmpty()) {
        throw new Excepcio(1);
    }

    try {
        bean.setInsert("insert into equip (nom, numMecanics, pressupost, campioMundial) values "
                + "('" + nom + "'," + numMecanics + ",'" + pressupost + "'," + campioMundial + ")");
    } catch (PropertyVetoException ex) {
        JOptionPane.showMessageDialog(null, ex.getMessage());
    }
}
```

Inserir pilots

```
public void insertarPilotBd(int dorsal, String nom, String email, String nom_equipf1, EquipF1 equip) throws Excepcio {
    if (nom.trim().isEmpty()) {
        throw new Excepcio(1);
    }
    for(Pilot pilot : pilots) {
        if (pilot.get2_nom().equalsIgnoreCase(nom)) {
            throw new Excepcio(2);
        }
    }
    Matcher matcher = pattern.matcher(email);
    if(matcher.matches()) {
        try {
            bean.setInsert("insert into pilot (dorsal, nom, email, nom_equipf1) values "
                    + "(" + dorsal + ",'" + nom + "','" + email + "','" + nom_equipf1 + "')");
        } catch (PropertyVetoException ex) {
            JOptionPane.showMessageDialog(null, ex.getMessage());
        }
    } else {
        throw new Excepcio(3);
    }
}
```

Dintre del bean controlem que l'string de la consulta sigui correcte i si no retornem una «PropertyVetoException» segons l'error que obtinguem: error d'escriptura, clau primària duplicada o altres.

```
case FormulaBean.PROP_INSERT:
    try {
        String SQL = evt.getNewValue().toString();
          System.out.println(SQL);
        Statement st;
        st = con.createStatement();
        st.executeUpdate(SQL);
    } catch (SQLException ex) {
        if (ex.getErrorCode() == 1064) {
            throw new PropertyVetoException("Hi ha un error d'escriptura en la consulta SQL", evt);
        } else if (ex.getErrorCode() == 1062) {
            throw new PropertyVetoException("Clau primària duplicada", evt);
        } else {
            throw new PropertyVetoException(ex.toString(), evt);
        }
    }
    break;
```

Borrar

Igual que en el cas anterior canviem els mètodes que teníem per borrar per unes consultes en format string i modifiquem l'atribut insert del bean que ja he explicat en le cas anterior.

```
public void eliminarEquipBd(EquipF1 equip, String nom) {
    try {
        bean.setInsert("delete from equip where nom = '" + nom + "'");
        this.<EquipF1>borrar(equip, equips);
    } catch (PropertyVetoException ex) {
        JOptionPane.showMessageDialog(null, ex.getMessage());
    }
}

public void eliminarPilotBd(Pilot pilot, int dorsal) {
    try {
        bean.setInsert("delete from pilot where dorsal = " + dorsal);
        this.<Pilot>borrar(pilot, pilots);
    } catch (PropertyVetoException ex) {
        JOptionPane.showMessageDialog(null, ex.getMessage());
    }
}
```

Modificar

Per fer les modificacions tornem a utilitzar la propietat del bean «insert», al model la modifiquem en un setter que li passa un string en la consulta a fer. Si la consulta és correcta es modifica la base de dades i si no el bean retorna una «PropertyVetoException».

```
public void modificarEquipsBd(EquipF1 equip, String nom_antic, String nom, int numMecanics, double pressupost,
        boolean campioMundial) throws Excepcio {
    if (nom.trim().isEmpty()) {
        throw new Excepcio(1);
    }
    try {
        bean.setInsert("update equip set nom = '" + nom + "', numMecanics = " + numMecanics + ", pressupost = '" +
                pressupost + "', campioMundial = " + campioMundial + " where nom = '"+ nom_antic + "'");
        this.<EquipF1>borrar(equip, equips);
    } catch (PropertyVetoException ex) {
        JOptionPane.showMessageDialog(null, ex.getMessage());
    }
}

public void modificarPilotBd(Pilot pilot, int dorsal_antic, int dorsal, String nom, String email, String nom_equipf1)
        throws Excepcio {
    this.<Pilot>borrar(pilot, pilots);
    if (nom.trim().isEmpty()) {
        throw new Excepcio(1);
    }
    for(Pilot pilots : pilots) {
        if (pilots.get2_nom().equalsIgnoreCase(nom)) {
            throw new Excepcio(2);
        }
    }
    Matcher matcher = pattern.matcher(email);
    if(matcher.matches()) {
        try {
            bean.setInsert("update pilot set dorsal = " + dorsal + ", nom = '" + nom + "', email = '" + email
                    + "', nom_equipf1 = '" + nom_equipf1 + "' where dorsal = " + dorsal_antic);
        } catch (PropertyVetoException ex) {
            JOptionPane.showMessageDialog(null, ex.getMessage());
        }
    } else {
        throw new Excepcio(3);
    }
}
```

Tancar connexió

Per tancar la connexió al apagar l'aplicació tenim la propietat al bean restringida «tancar».

Desde el model la modifiquem passant-li un string.

```
public void tancarConexio() {
    try {
        bean.setTancar("Tanca");
    } catch (PropertyVetoException ex) {
        JOptionPane.showMessageDialog(null, ex.getMessage());
    }
}
```

I al bean tanquem la connexió «con.close()», si hi ha algun error retrona una «PropertyVetoException».

```
case FormulaBean.PROP_TANCAR:
    try {
        con.close();
    } catch (SQLException e) {
        throw new PropertyVetoException("Error al tancar", evt);
    }
    break;
```