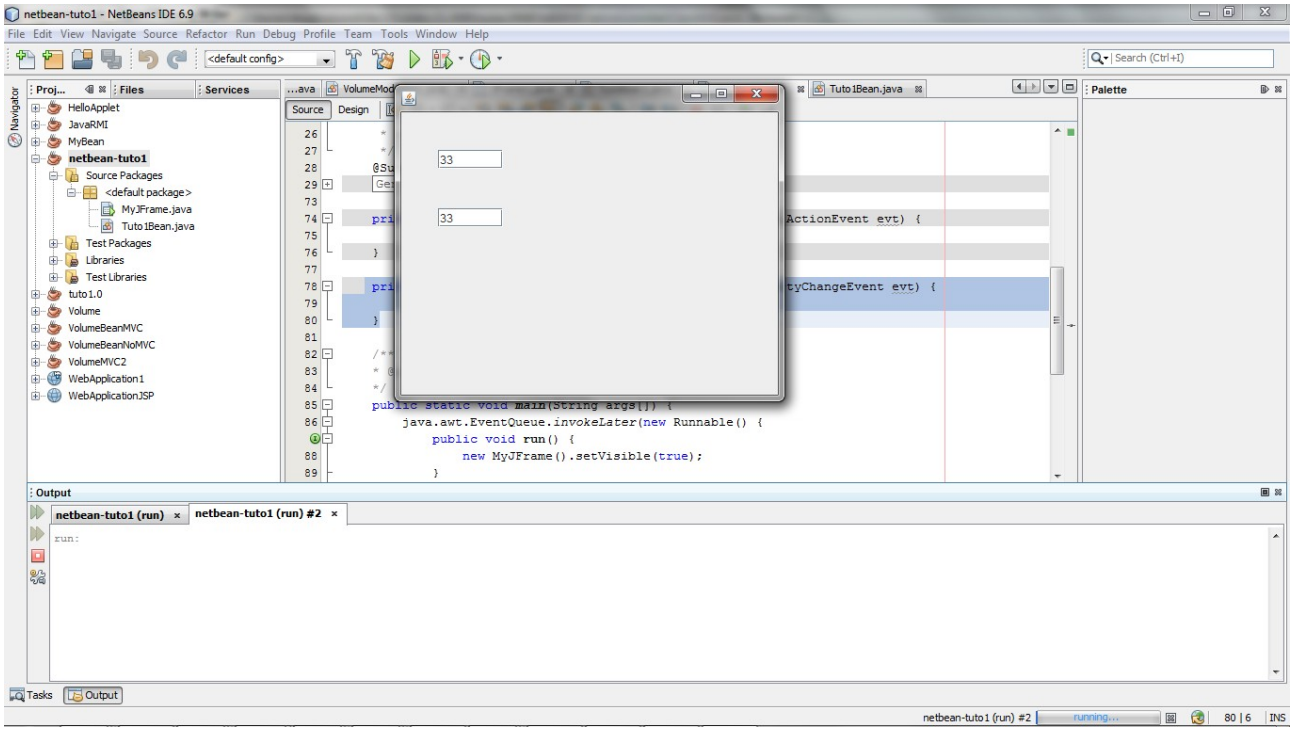
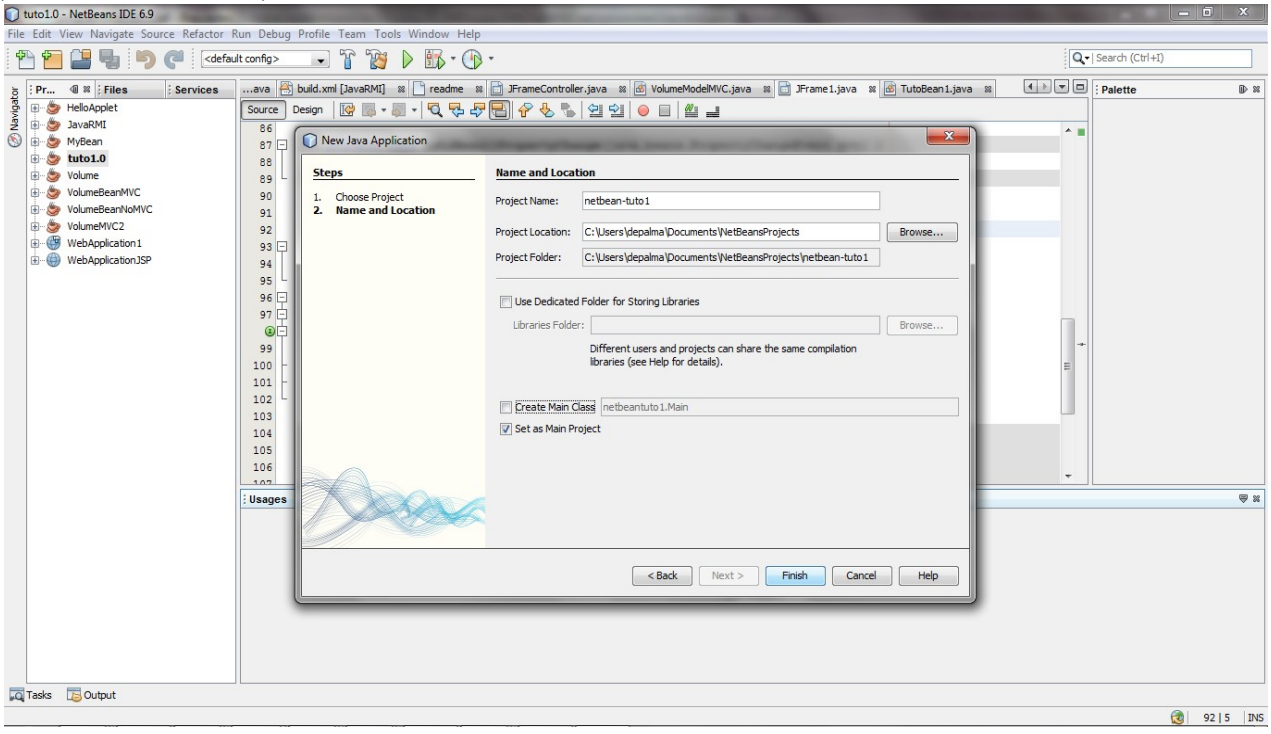
Tutorial JavaBean 1

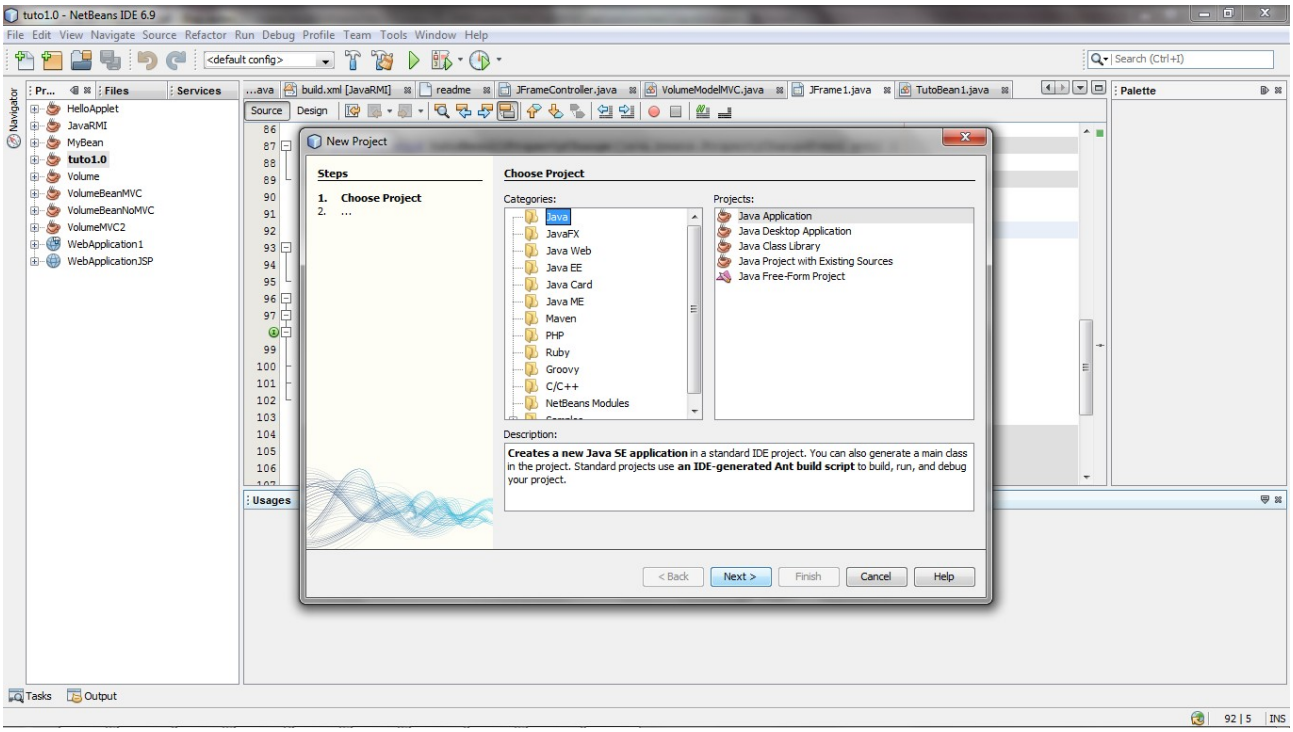
Ce document explique pas à pas la construction d'une application très simple permettant d'illustrer l'utilisation d'un JavaBean et d'une propriété liées vers des composant graphiques (JtextField).
Cette application est constitué de 2 textfields (jTextField1 et jTextField2) et d'un JavaBean (Tuto1Bean). Lorsque l'utilisateur saisi une valeur dans jTextField1, cette valeur est utilisée pour modifier la propriété myString du javaBean. Cette propriété est liées à jTextField2 qui mettra automatiquement son affichage en cohérence. Voici une capture de l'application :



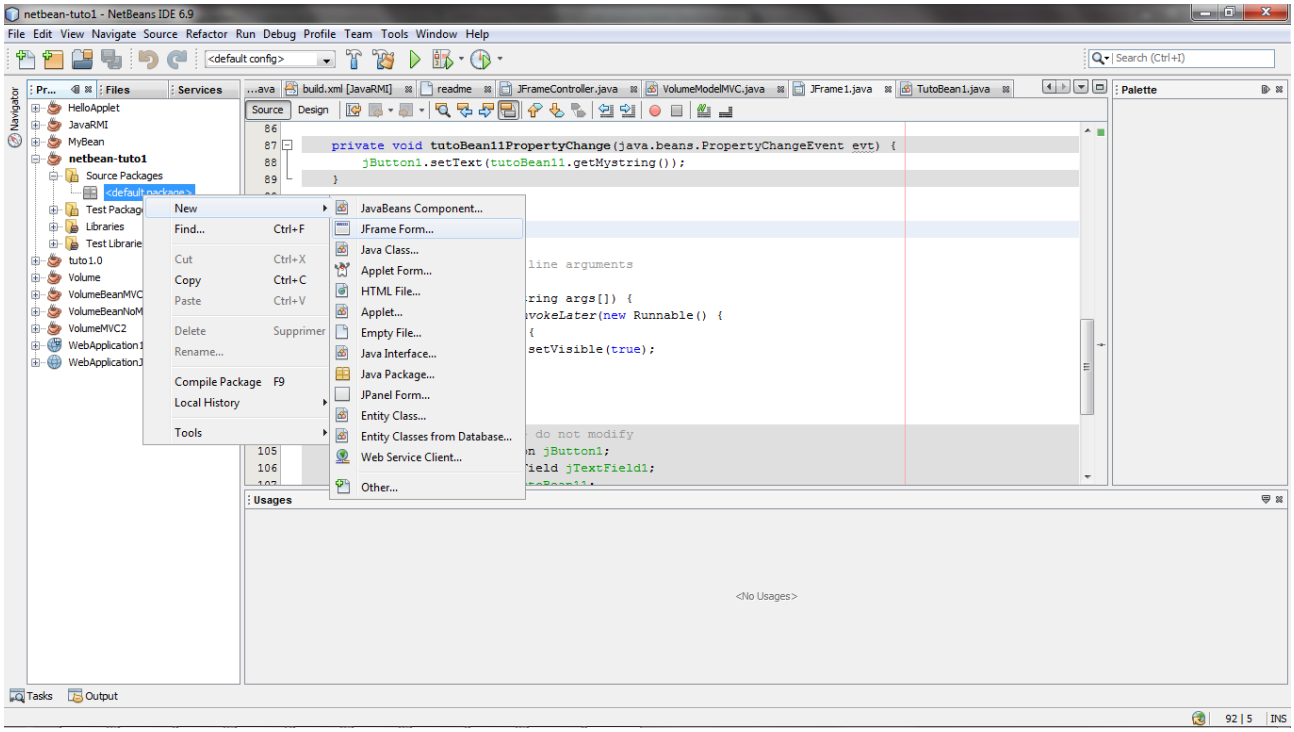
Etape1 : Création du projet sous NetBean
(décocher main class)



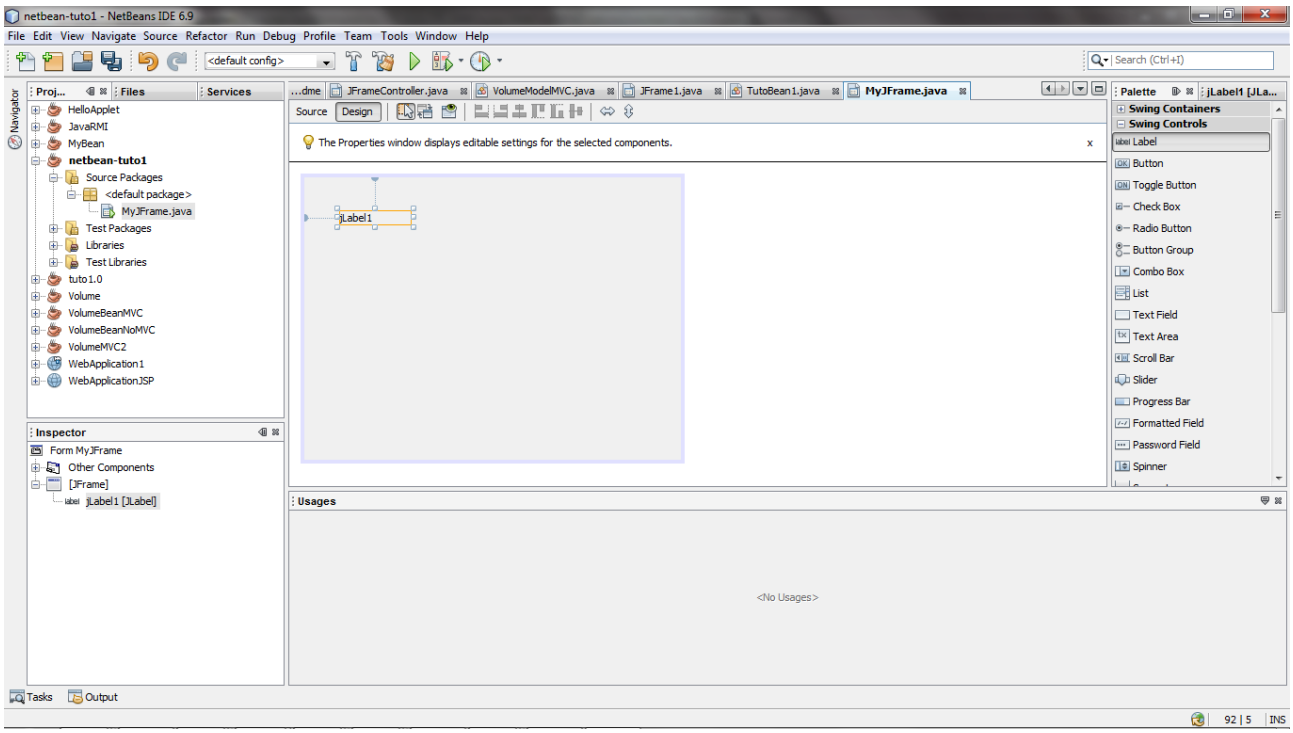
Etape 2 : Sélectionner projet Java/java application



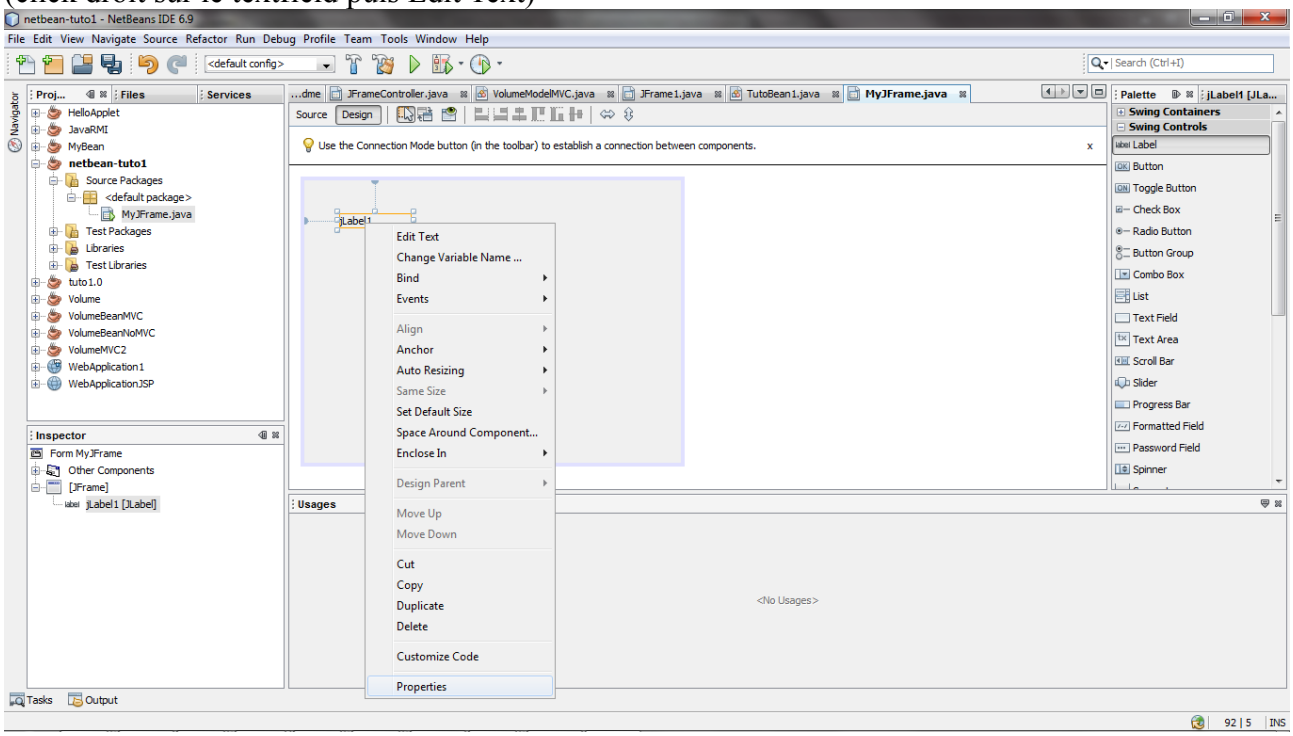
Etape 3 : creation d'une Jframe



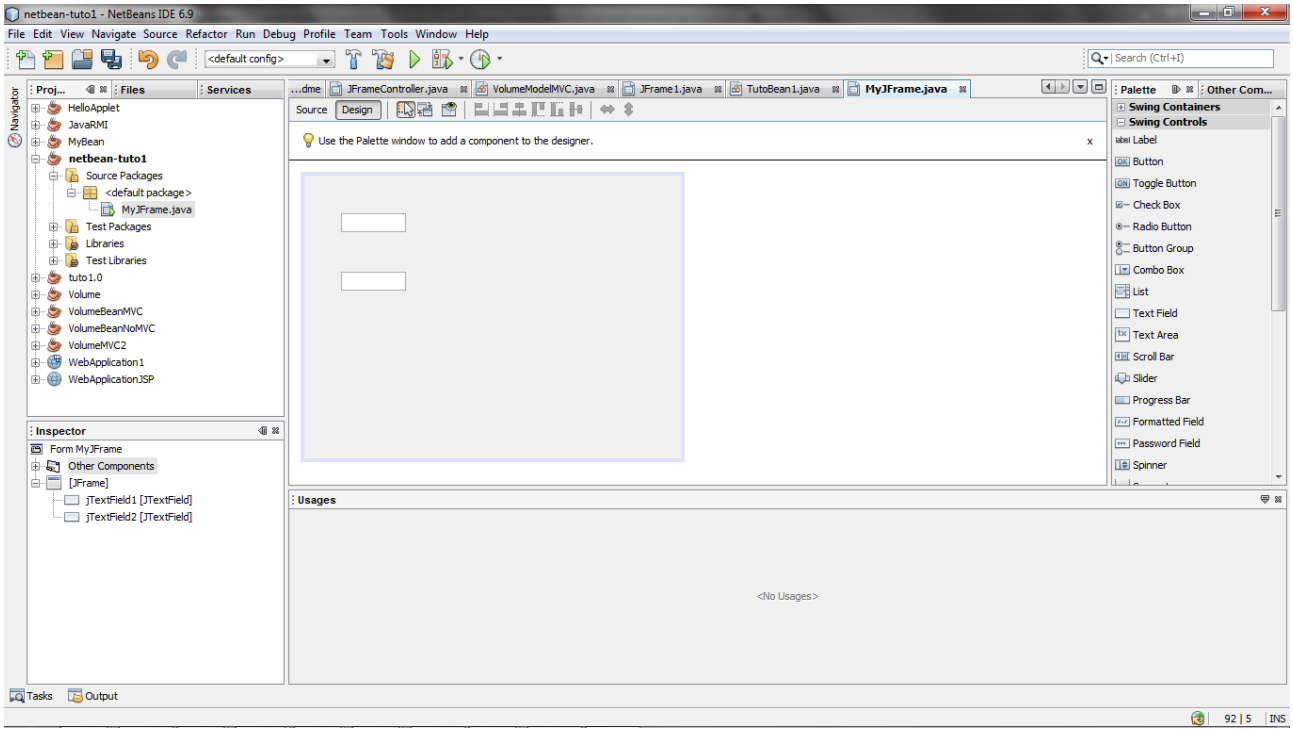
Etape4 : Ajout d'un jTextfield (jTextField1)



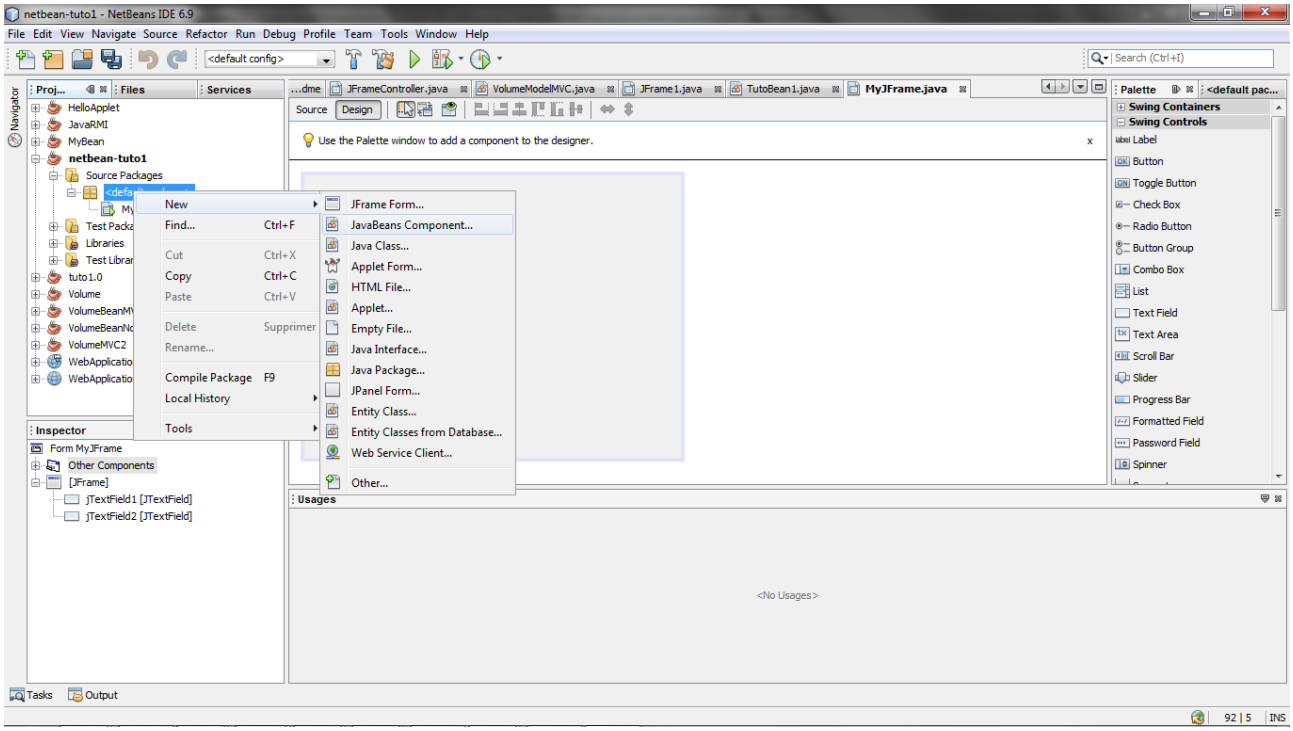
Etape 5 : Edition du texte du Jtextfield1
(click droit sur le textfield puis Edit Text)



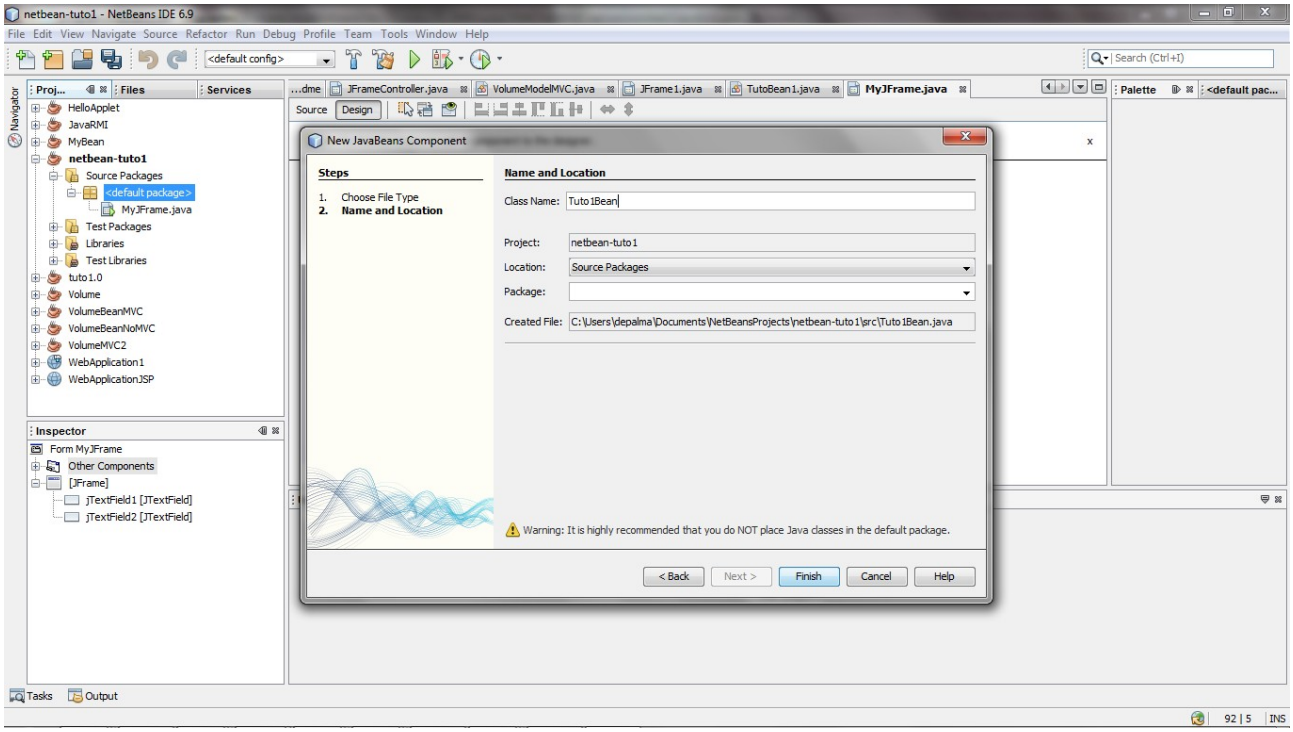
Etape 6 : idem pour jTextField2



Etape 7 : Creation du javabean (tuto1Bean1)



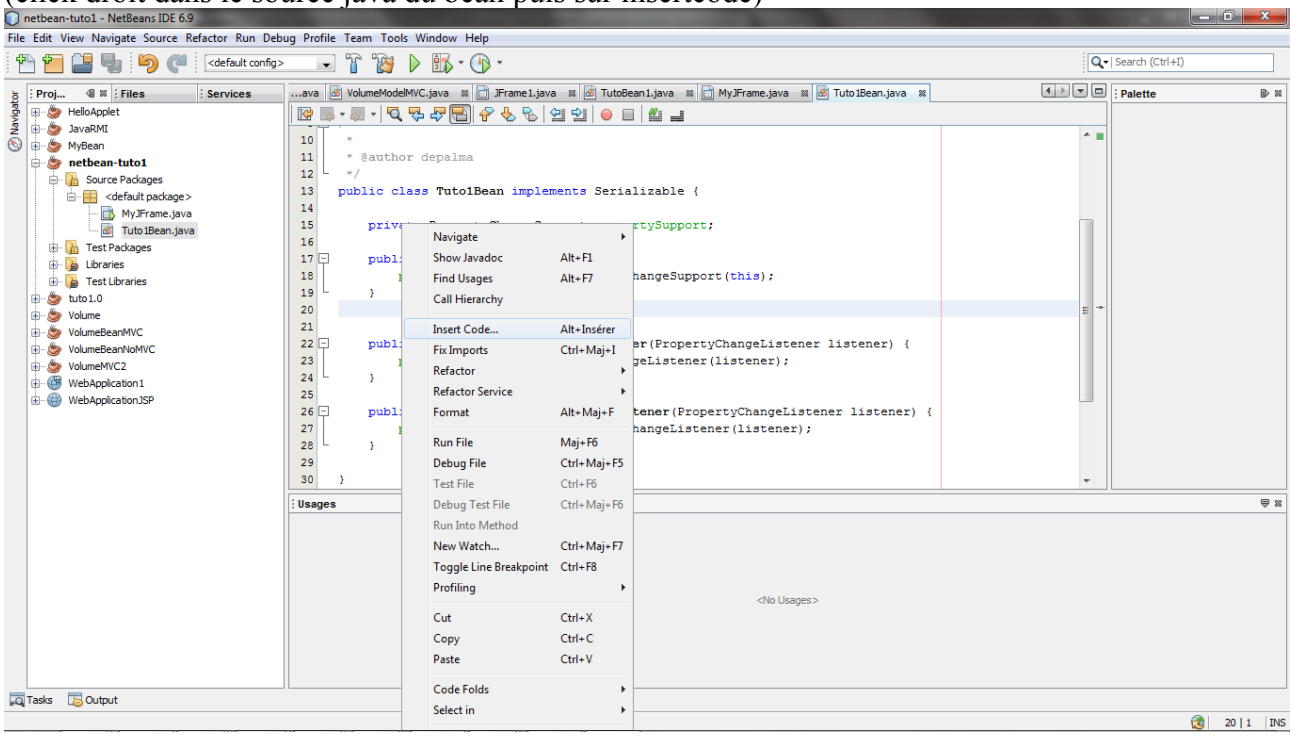
Fixer le nom du bean : tuto1Bean1



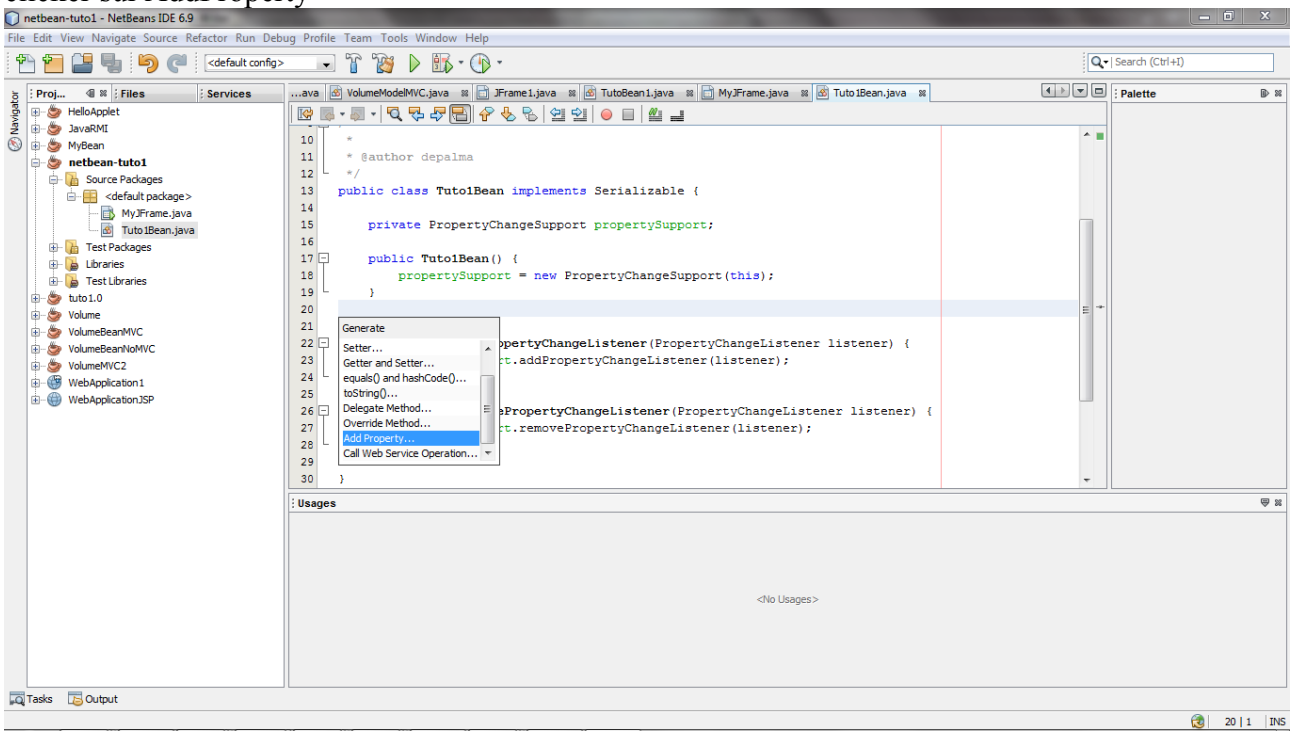
Dans le source java du bean, supprimmer les ligne en gras :

```
public class Tuto1Bean implements Serializable {
    public static final String PROP_SAMPLE_PROPERTY = "sampleProperty";
    private String sampleProperty;

    private PropertyChangeSupport propertySupport;

    public Tuto1Bean() {
        propertySupport = new PropertyChangeSupport(this);
    }

    public String getSampleProperty() {
        return sampleProperty;
    }

    public void setSampleProperty(String value) {
        String oldValue = sampleProperty;
        sampleProperty = value;
        propertySupport.firePropertyChange(
           PROP_SAMPLE_PROPERTY, oldValue, sampleProperty);
    }

    public void addPropertyChangeListener(PropertyChangeListener
listener) {
        propertySupport.addPropertyChangeListener(listener);
    }

    public void removePropertyChangeListener(PropertyChangeListener
listener) {
        propertySupport.removePropertyChangeListener(listener);
    }
}
```

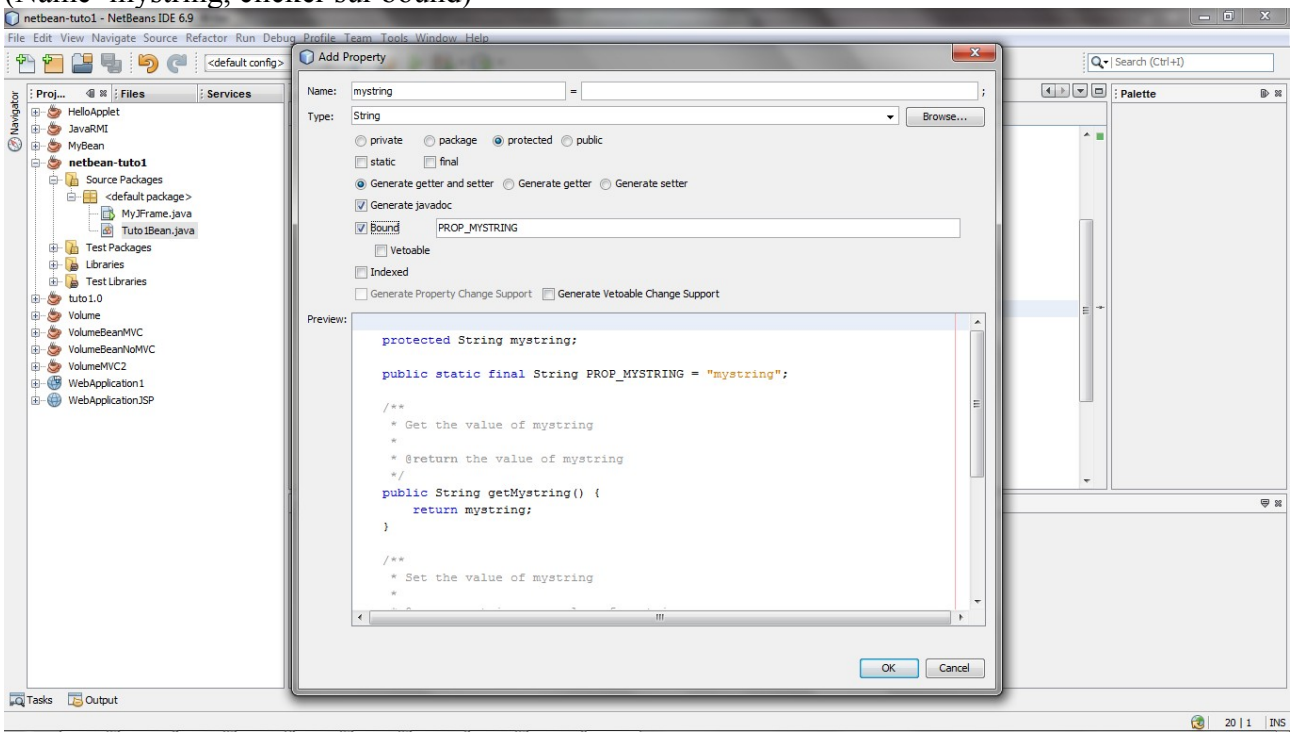
Etape 8 : ajout d'une propriété au bean
(click droit dans le source java du bean puis sur insertcode)



clicker sur AddProperty



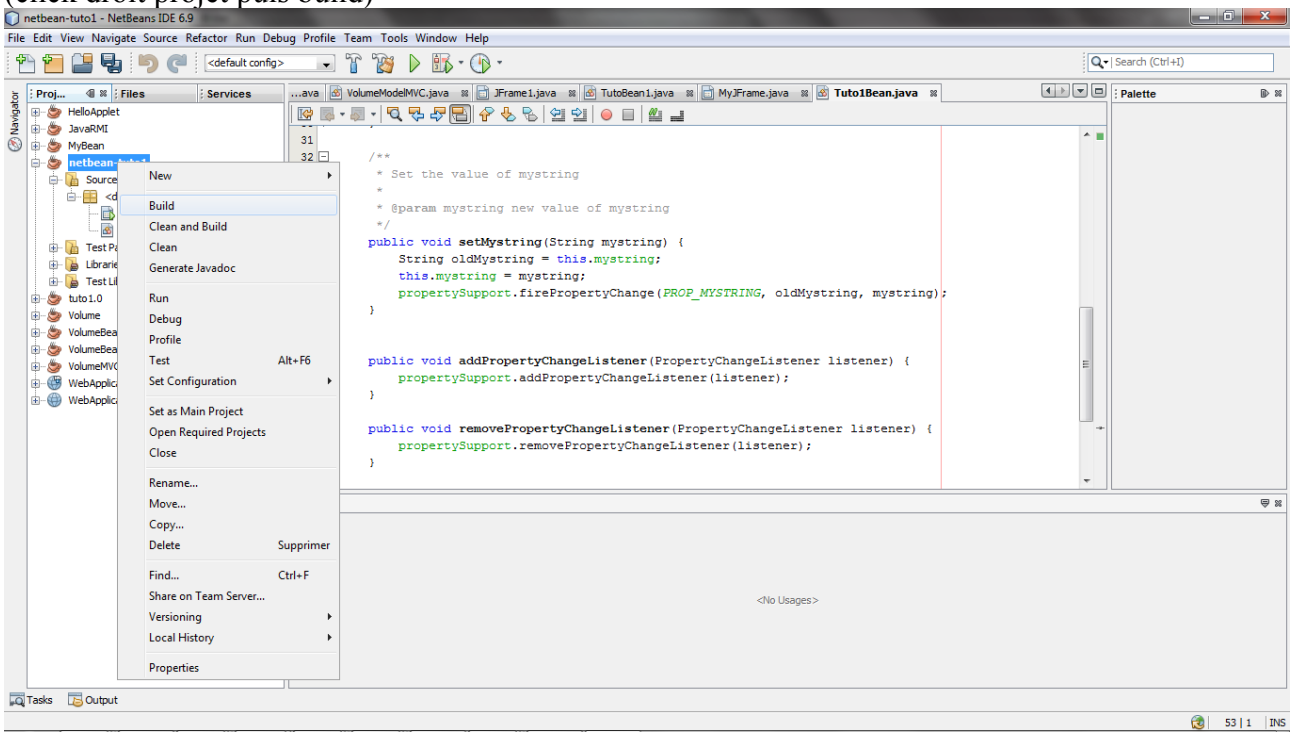
Renseigner les information sur la propriété
(Name=mystring, clicker sur bound)



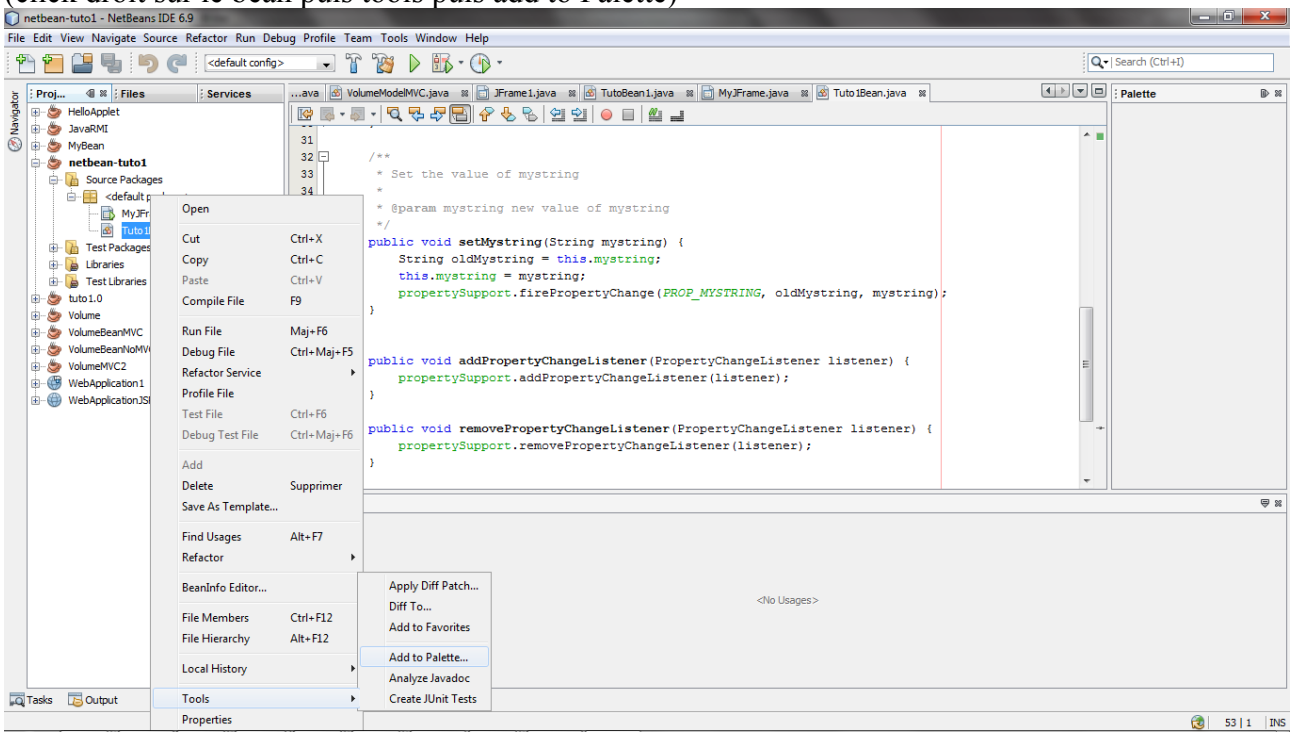
La classe du bean doit ressembler à ca :

```
import java.beans.*;
import java.io.Serializable;
public class Tuto1Bean implements Serializable {
    private PropertyChangeSupport propertySupport;

    public Tuto1Bean() {
        propertySupport = new PropertyChangeSupport(this);
    }
    protected String mystring;
    public static final String PROP_MYSTRING = "mystring";

    public String getMystring() {
        return mystring;
    }

    public void setMystring(String mystring) {
        String oldMystring = this.mystring;
        this.mystring = mystring;
        propertySupport.firePropertyChange(PROP_MYSTRING,
                oldMystring, mystring);
    }

    public void addPropertyChangeListener(
                    PropertyChangeListener listener) {
        propertySupport.addPropertyChangeListener(listener);
    }

    public void removePropertyChangeListener(
                    PropertyChangeListener listener) {
        propertySupport.removePropertyChangeListener(listener);
    }
}
```

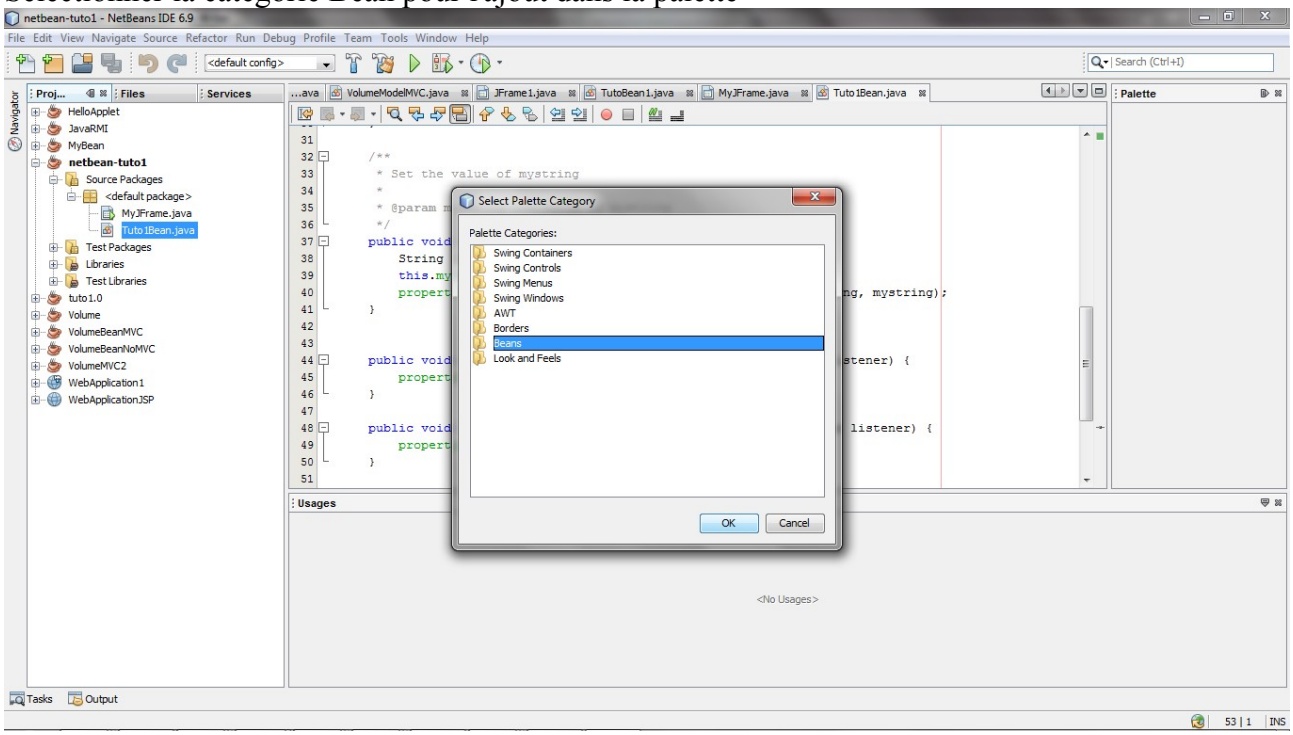
Etape 9 : Lancer le build du projet
(click droit projet puis build)



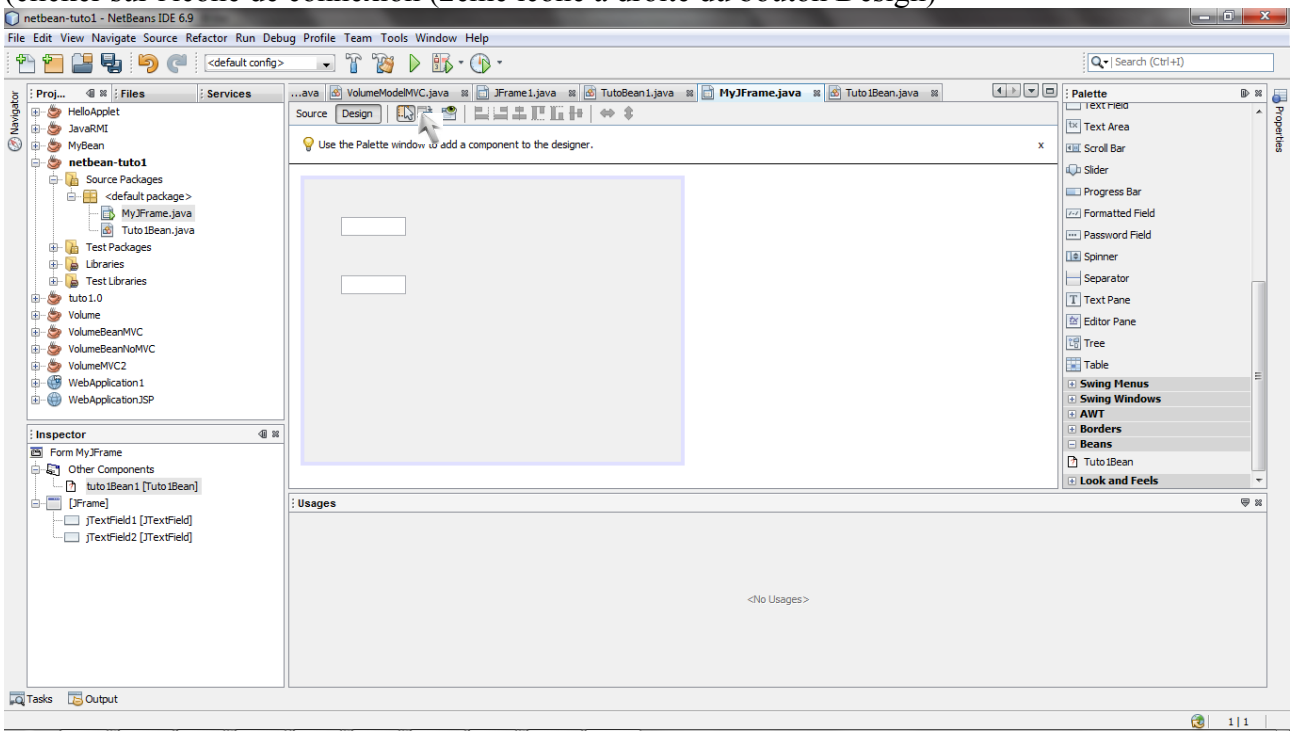
Etape 10 : Ajout du bean dans la palette
(click droit sur le bean puis tools puis add to Palette)



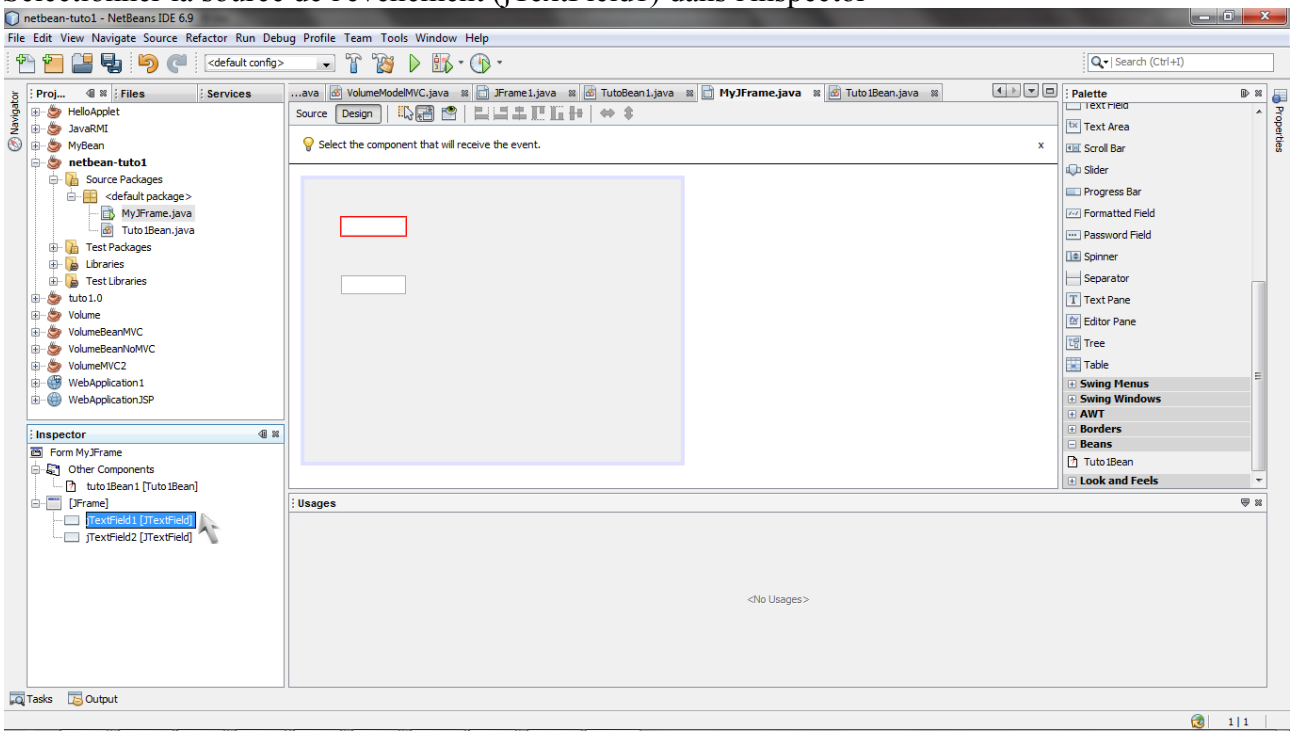
Sélectionner la catégorie Bean pour l'ajout dans la palette



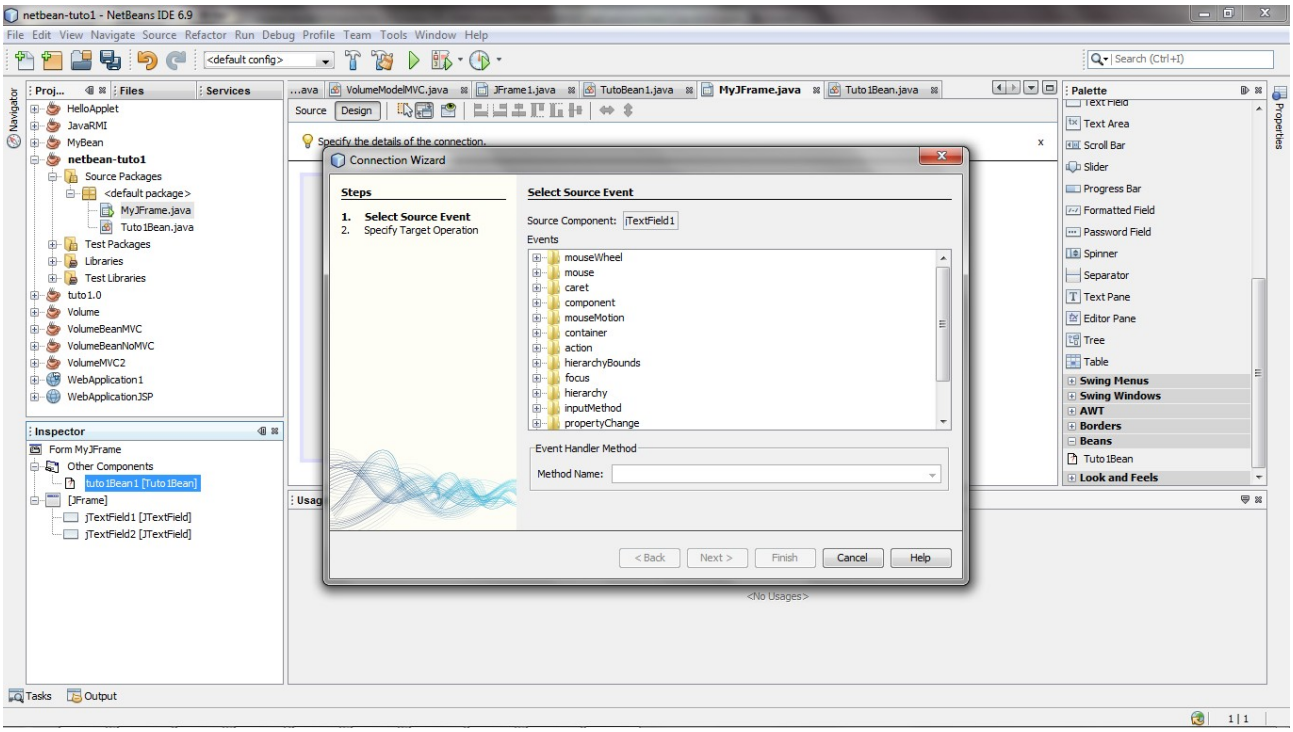
Etape 11 : connexion de jTextField1 au bean
(clicker sur l'icone de connexion (2eme icône a droite du bouton Design)



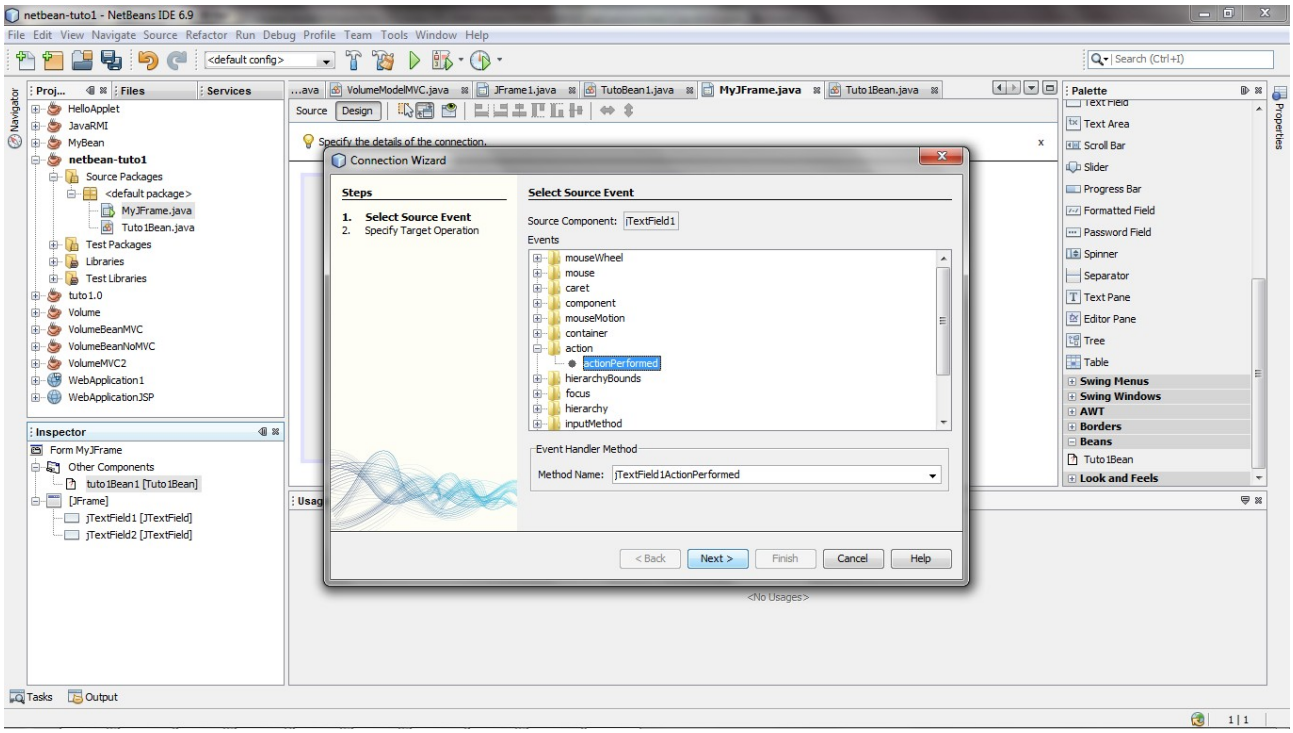
Sélectionner la source de l'événement (jTextField1) dans l'inspector



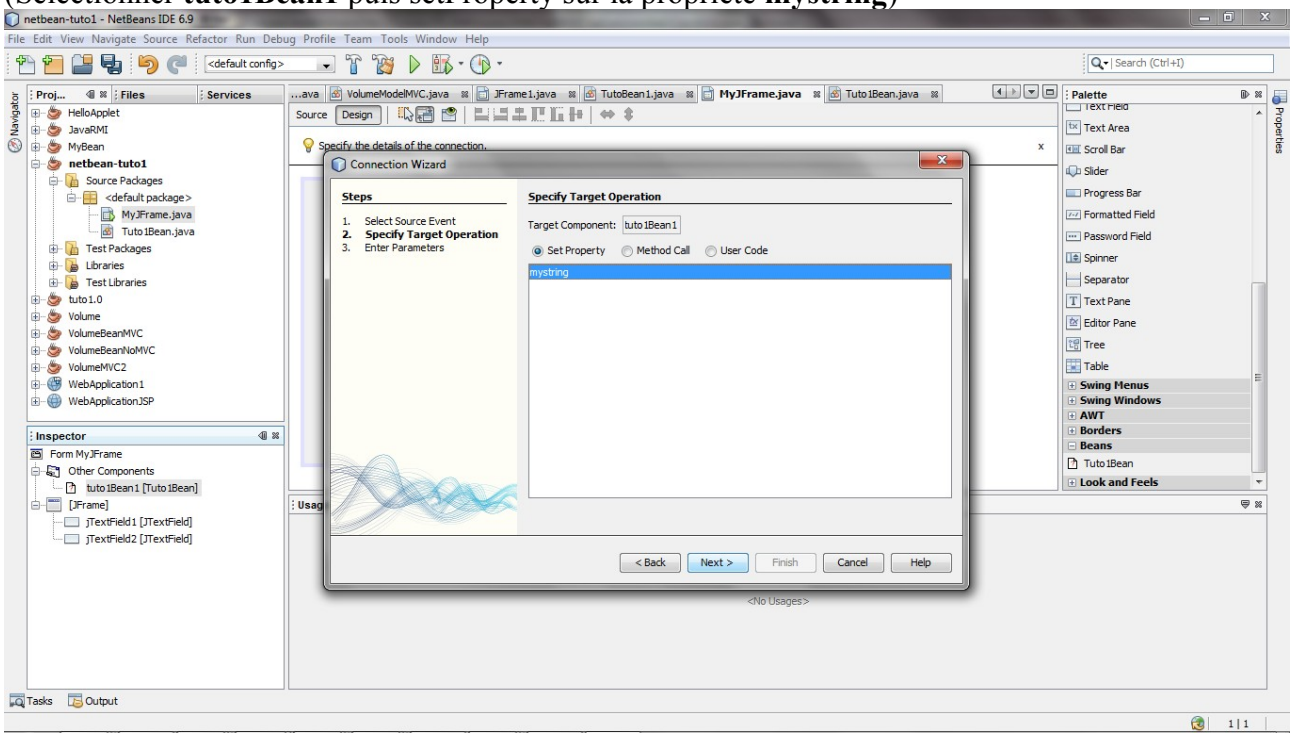
Sélectionner le recepteur de l'événement (tuto1Bean1) dans l'inspector
Une fenêtre apparaît pour sélectionner le type d'évènement que la source (jTextField1) va envoyer au recepteur (bean).



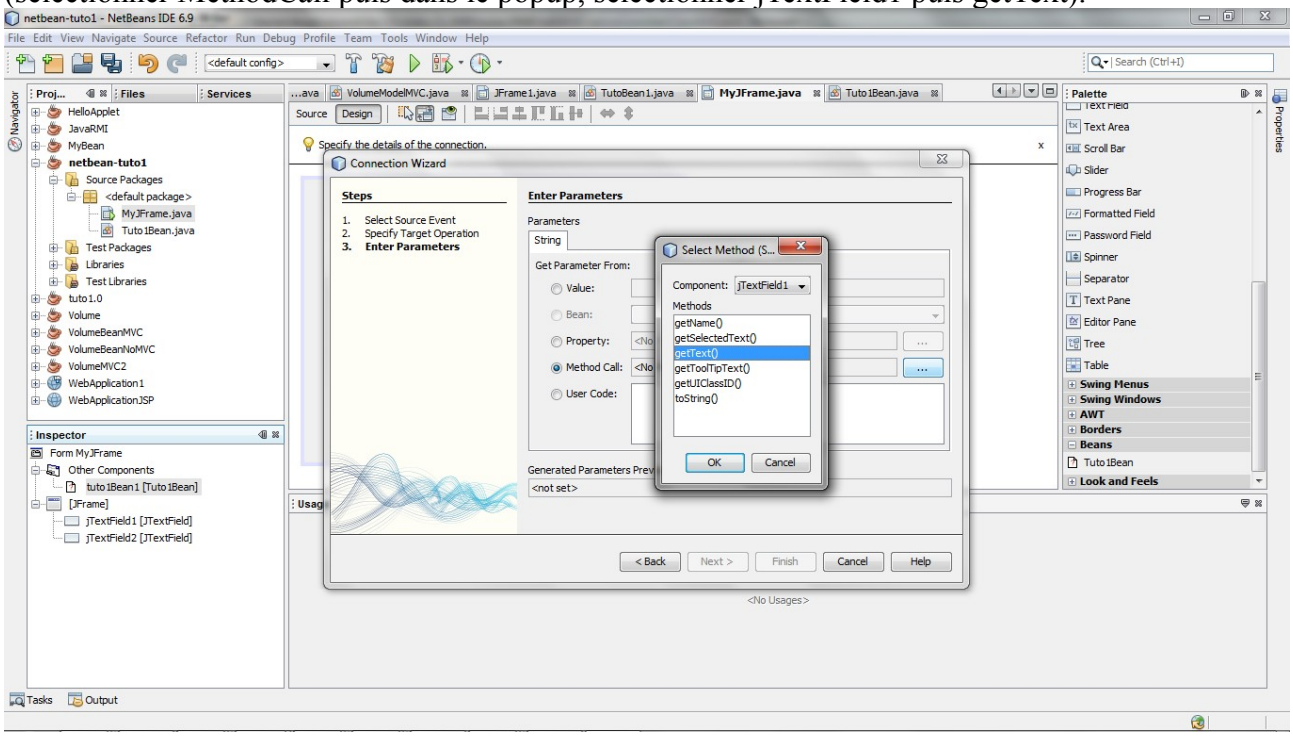
Sélectionner le composant jTextField1 puis action=>actionPerformed



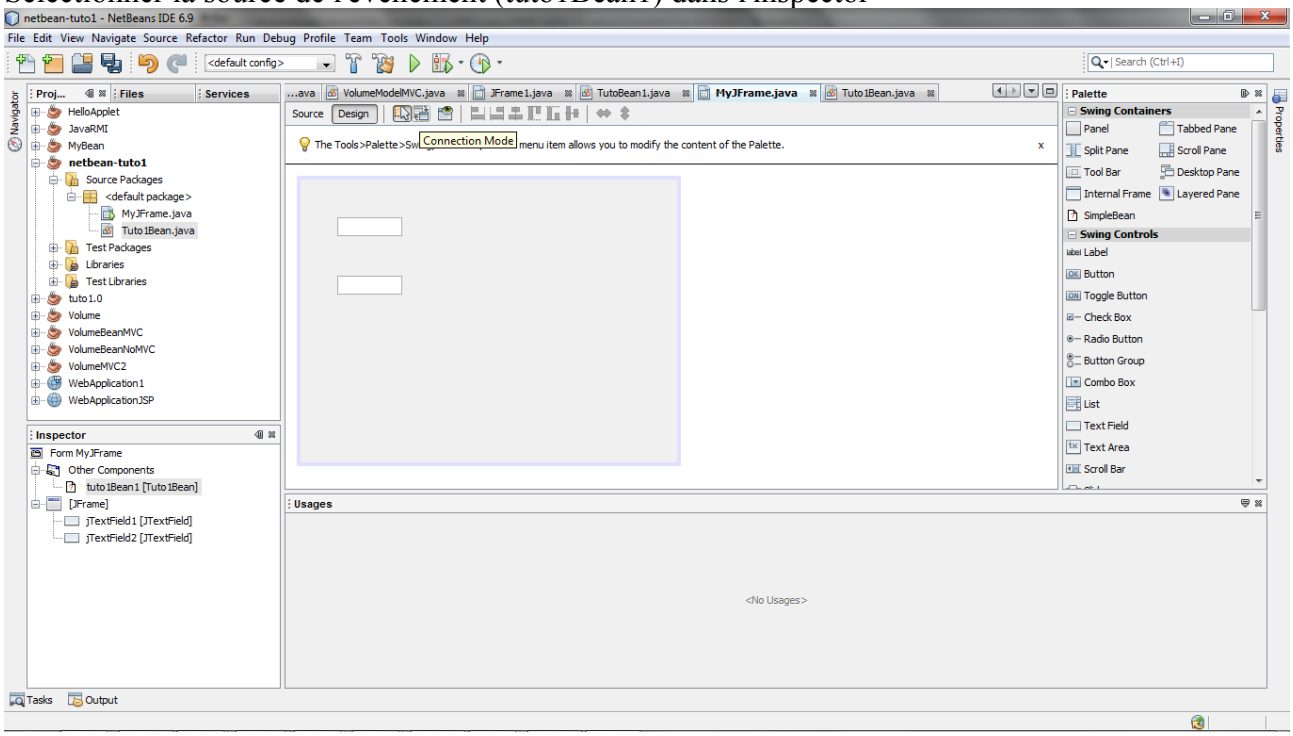
Sélectionner l'opération à effectuer sur le récepteur de l'évènement (tuto1Bean1)
(Sélectionner **tuto1Bean1** puis setProperty sur la propriété **mystring**)



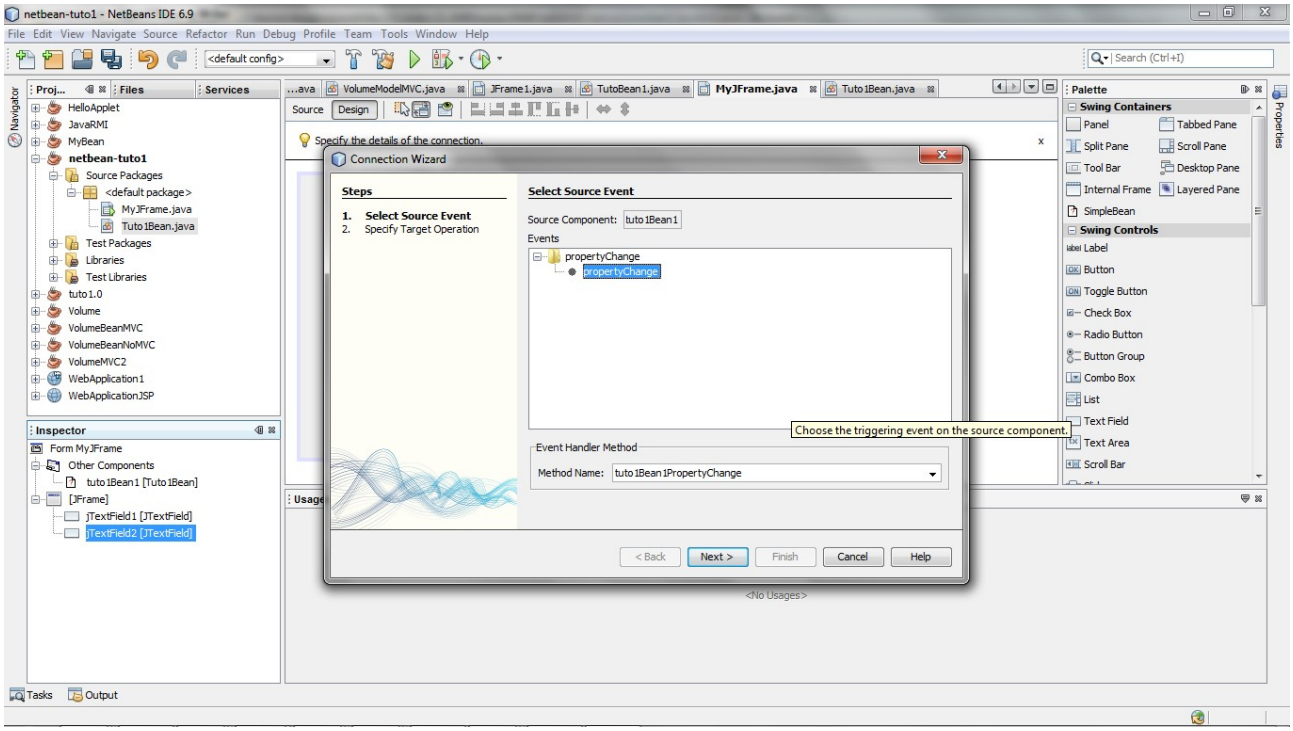
Sélectionner comment les données sont extraites de la source (jTextField1)
(sélectionner MethodCall puis dans le popup, selectionner jTextField1 puis getText).



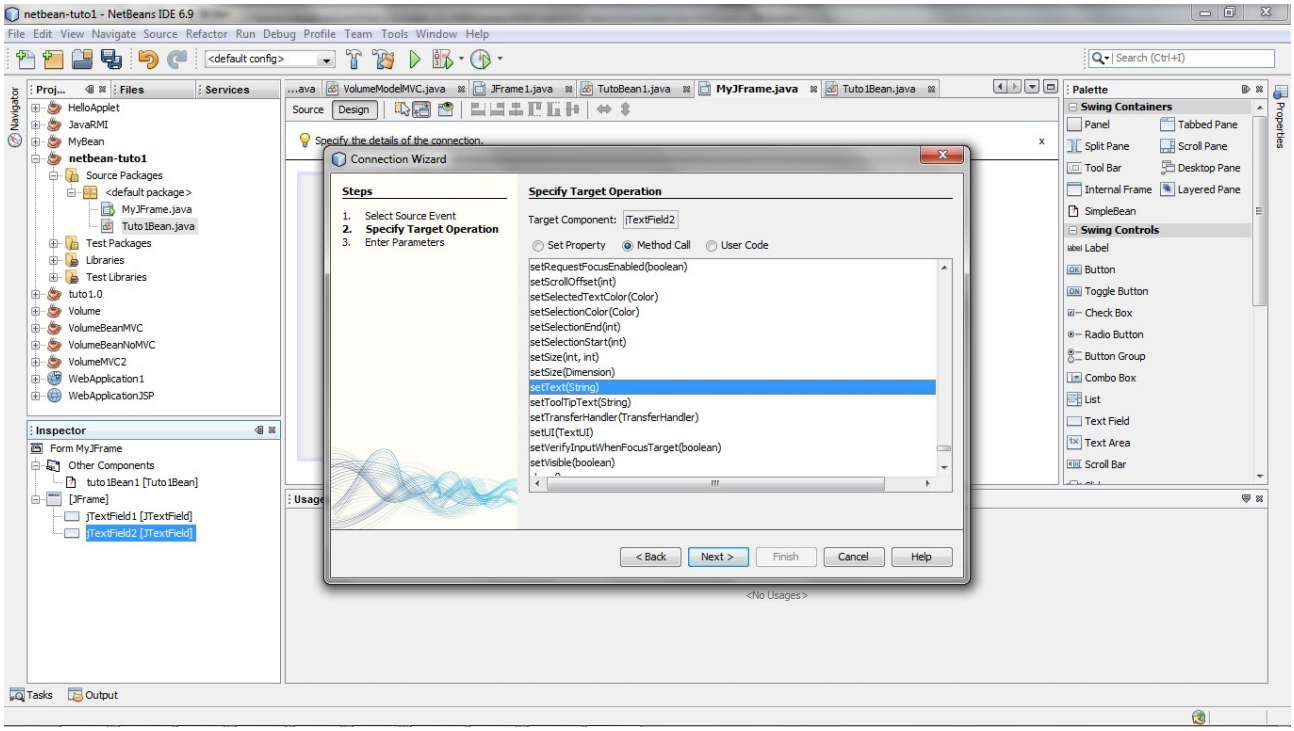
Etape 12 : Connexion du bean et du jTextField2
Sélectionner la source de l'événement (tuto1Bean1) dans l'inspector



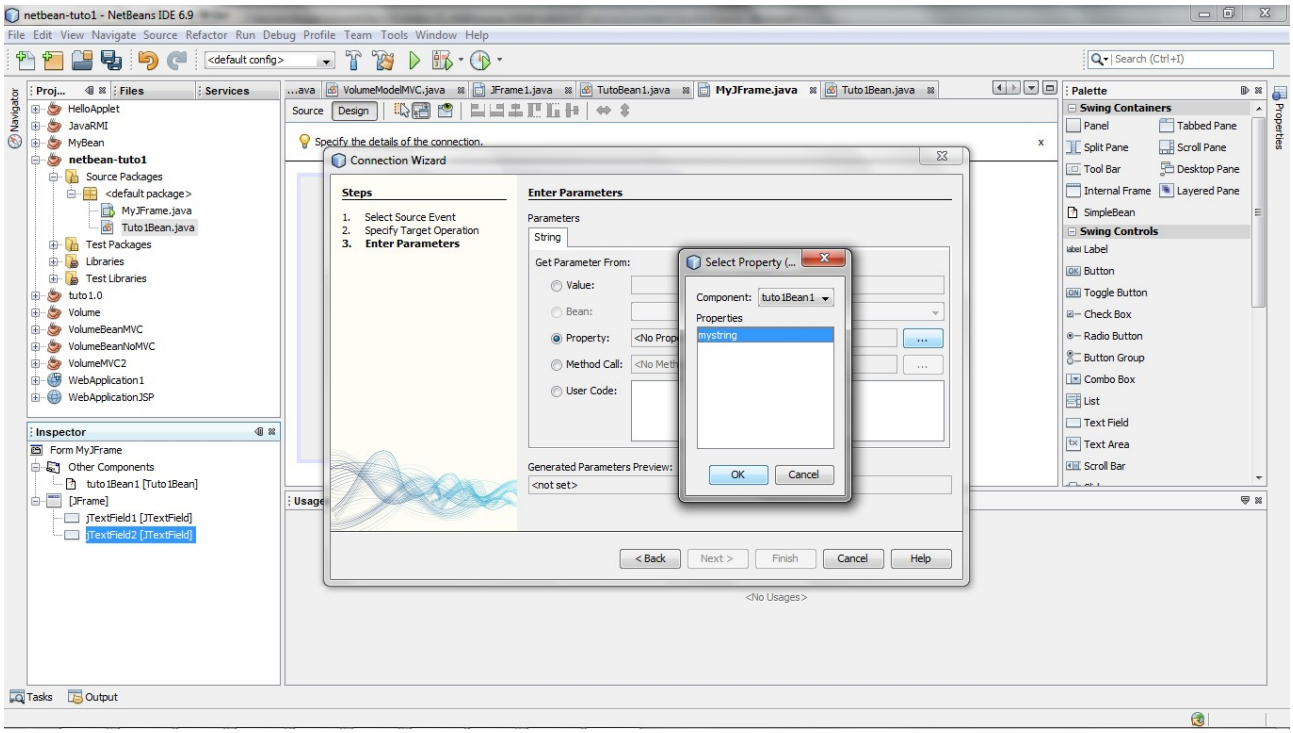
Sélectionner le recepteur de l'événement (JtextField2) dans l'inspector
Une fenêtre apparaît pour sélectionner le type d'évènement que la source (bean) va envoyer au recepteur (jTextField2). Selectionner tuto1Bean1 puis propertyChange



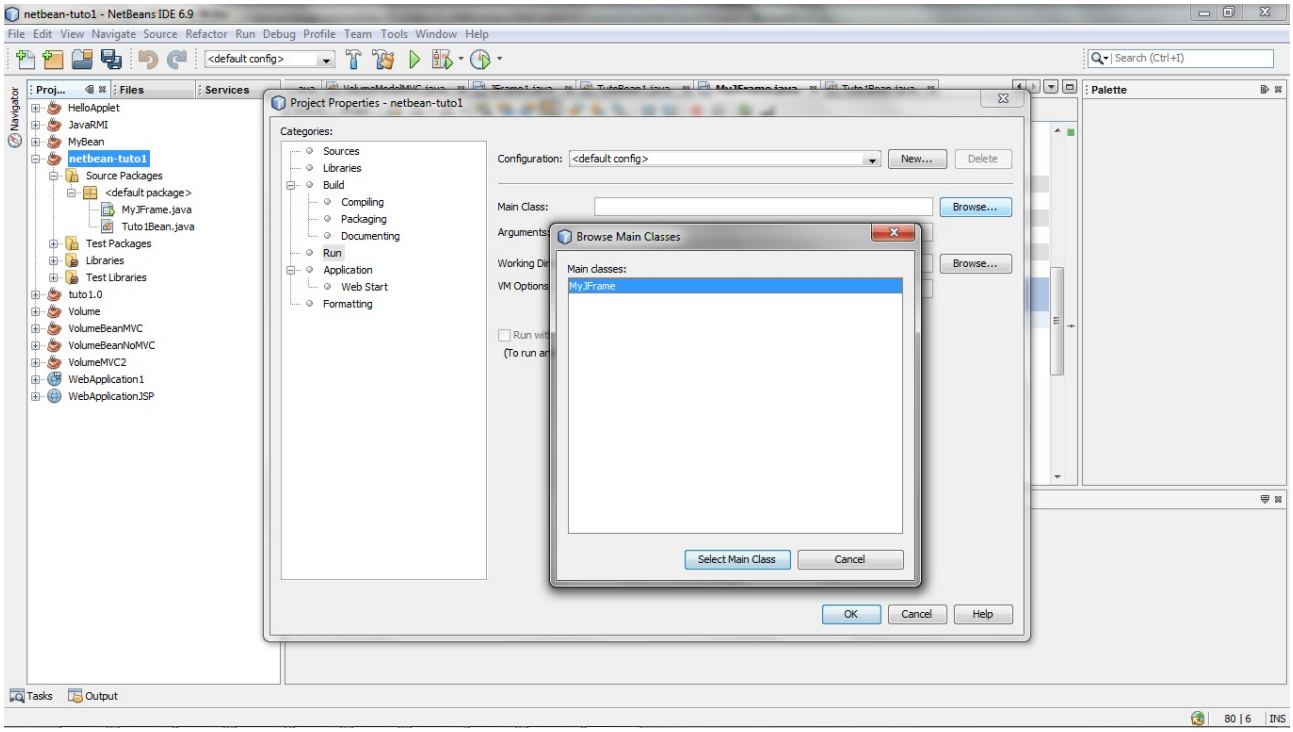
Sélectionner l'opération déclenchée coté recepteur
clicker sur MethodCall puis setText



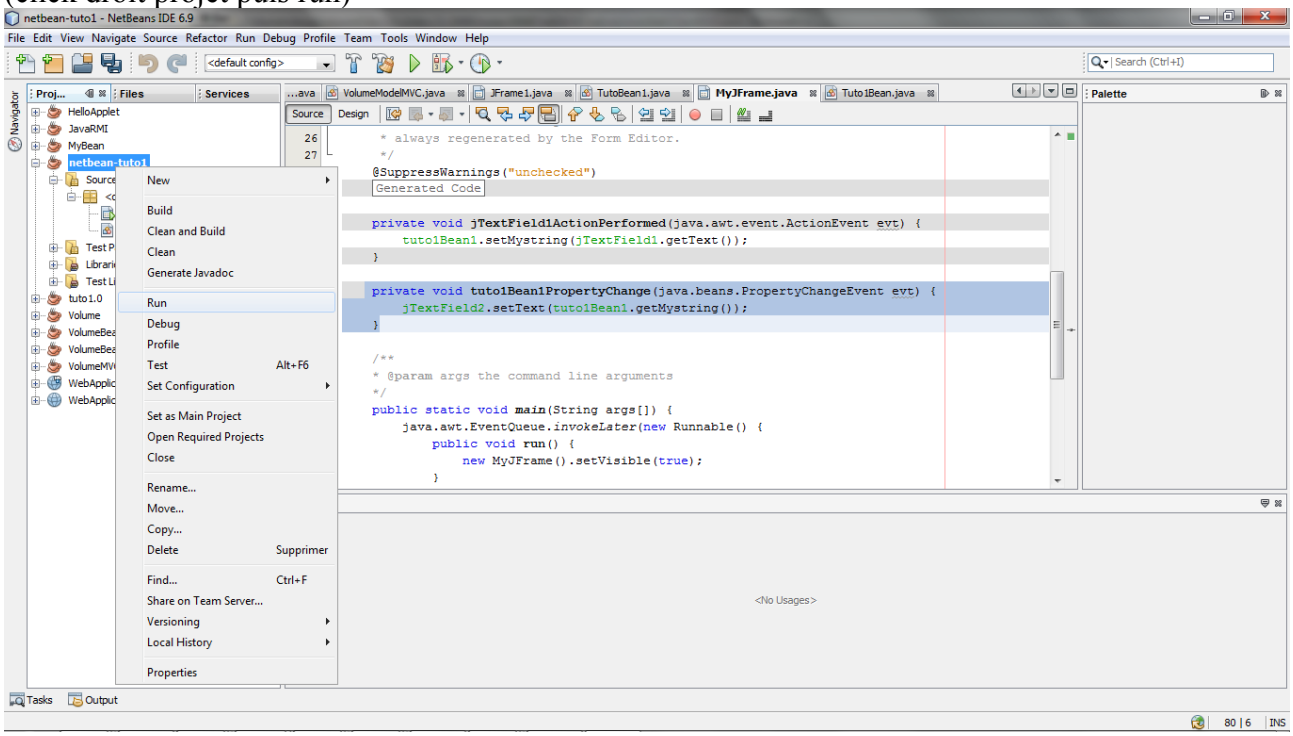
Sélectionner comment les données sont extraites de la source (tuto1Bean1)
(sélectionner Property puis dans le popup, selectionner tutoBean1 puis la propriété myString).



Etape 13 : définissez la classe du main
(click droit sur le projet puis property puis selectionner MyFrame dans le popup)



Etape 14 : Lancer le projet
(click droit projet puis run)



Travail à faire

Modification 1.
Lier jTextfield2 au bean et le bean au jTextfield1 de telle manière que lorsqu'on edite jTextField2, jTextfield1 est mis à jour.

Modification 2.
Ajouter un jTextField3, lier le textfield3 au bean et vice versa de telle manière que quelque soit le textfield qu'on édite, les deux autres sont mis en cohérence.