



BTS-SIO

Conception et développement d'applications

Atelier JavaFx

ATELIER N° 2 - CORRECTION



Objectifs

Le développement informatique concerne tout ce qui touche à l'étude, à la conception, à la construction, au développement, à la mise au point, à la maintenance et à l'amélioration des logiciels et autres applications et sites web. C'est le développeur informatique qui en a la charge. Il a plusieurs rôles : analyser les besoins des clients/utilisateurs, s'occuper de l'écriture informatique, rédiger les notices...etc.

Il doit posséder de multiples compétences : connaissance du langage informatique, expertise des technologies de bases de données.

JavaFX est un Framework et une bibliothèque d'interface utilisateur issue du projet OpenJFX, qui permet aux développeurs Java de créer une interface graphique pour des applications de bureau, des applications internet riches et des applications smartphones et tablettes tactiles.

Cet atelier va nous permettre de réviser les bases de JavaFx.



Contrainte de cet atelier

Connaitre correctement les bases du langage Java et la POO.

Avoir déjà vu les bases du Framework **JavaFx**

Considérations techniques & logicielles

Eclipse IDE pour les phases de programmation.

JavaFx installé et configuré sous Eclipse



Bloc

Bloc de compétences n°2 : option B « Solutions logicielles et applications métiers » - Conception et développement d'applications



Titre

Atelier 02 JavaFX - sujet

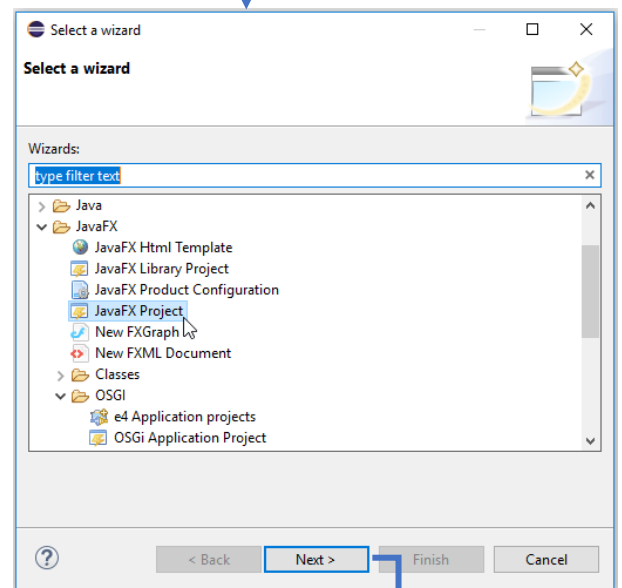
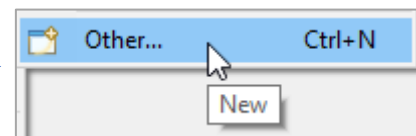
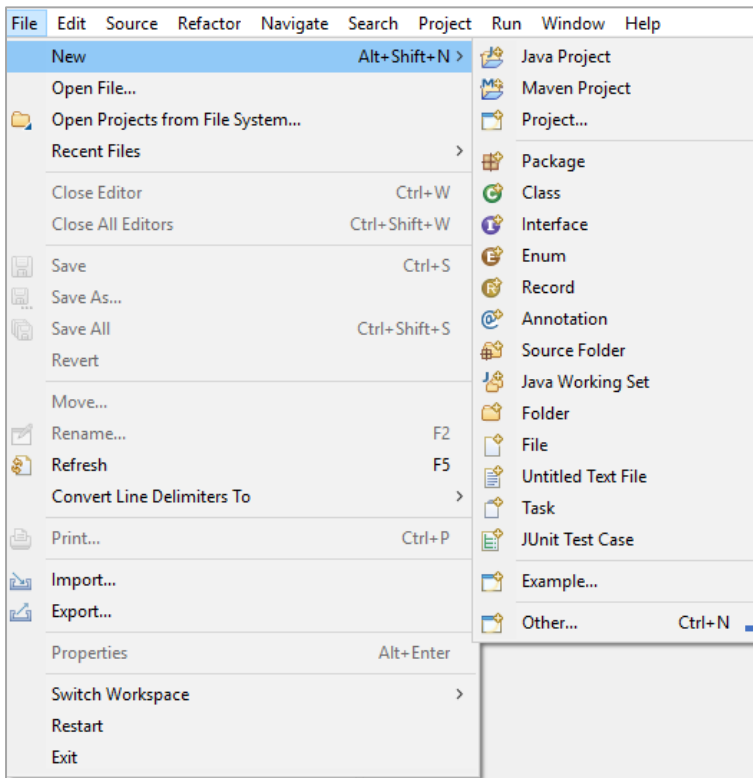
Table des matières

1. Projet.....	4
1.1. Création du projet.....	4
1.2. Création de la classe Rdv.....	5
1.3. Structure de la Première fenêtre – PriseRdv.fxml	5
1.3.1. Contrôle du contrôleur de la page	5
1.3.2. Création et définition des différents contrôles	6
1.4. Structure de la Première fenêtre – PriseRdvController.java	11
1.4.1. Gestion de la déclaration des contrôles fxml dans le contrôleur	11
1.4.1.1. Contrôles des fx:id dans le fichier fxml :	11
1.4.1.2. Déclaration de ces mêmes contrôles dans le fichier java :	11
1.4.1.3. Déclaration des variables globales dans le fichier java :	12
1.4.1.4. Descriptif de la méthode initialize()	13
1.4.1.5. Descriptif de la méthode evtOnMouseClickedBtnValider ()	14
1.5. Structure de la Seconde fenêtre – ConfirmationRdv.fxml	16
1.5.1. Contrôle du contrôleur de la page	16
1.5.2. Création et définition des différents contrôles	16
1.6. Structure de la Première fenêtre – PriseRdvController.java	17
1.6.1. Gestion de la déclaration des contrôles fxml dans le contrôleur	17
1.6.1.1. Contrôles des fx:id dans le fichier fxml :	17
1.6.1.2. Déclaration de ces mêmes contrôles dans le fichier java :	17
1.6.1.3. Descriptif de la méthode setLblTitre(String titre).....	18
1.6.1.4. Descriptif de la méthode setRdv(Rdv rdv)	18
1.6.1.5. Descriptif de la méthode getTxfReponse()	19
1.6.1.6. Descriptif de la méthode evtOnMouseClickedBtnValider ()	19
1.6.1.7. Descriptif de la méthode evtOnMouseClickedBtnRepondre ()	19
1.6.1.7. Descriptif de la méthode initialize ()	19
1.8. Modification du contrôleur – PriseRdvController.java afin de tenir compte des évolutions sur la seconde fenêtre	20

1. Projet

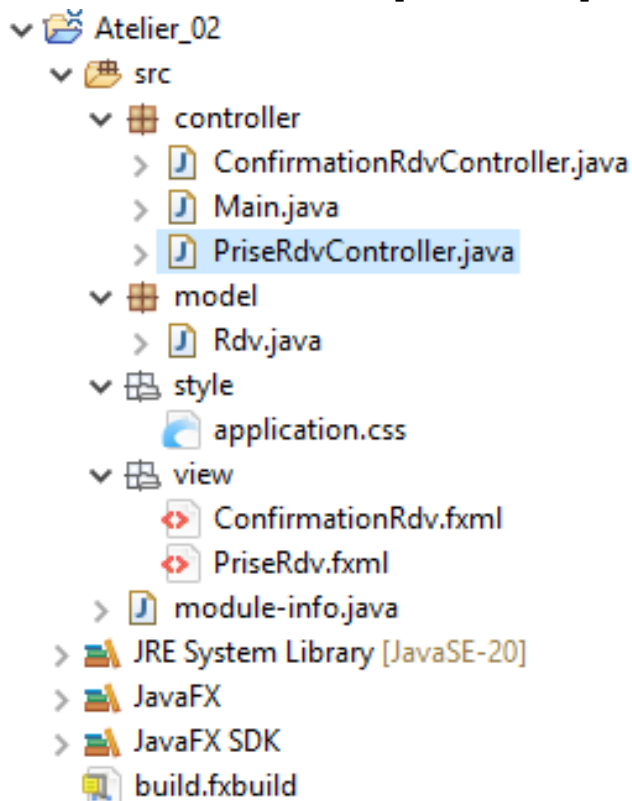
1.1. Création du projet

- Créez un nouveau projet JavaFx.



Next >

Donnez-lui comme nom [**Atelier-02**]



Modifier votre projet pour qu'il ait la structure suivante :

1.2. Création de la classe Rdv

1. Créez la classe [**Rdv**] à partir des attributs se trouvant dans le sujet,
2. Créez le constructeur avec tous les attributs,
3. Renommez les paramètres du constructeur en les faisant commencer par **newRdv** au lieu de **rdv**,
4. Rajoutez les **Getters** et les **Setters**
5. Rajoutez la méthode **toString()**.

1.3. Structure de la Première fenêtre – PriseRdv.fxml

1.3.1. Contrôle du contrôleur de la page

- La première page de l'application est la page [**PriseRdv.fxml**],
- Contrôlez dans le fichier FXML ou dans SceneBuilder qu'elle a comme contrôleur la classe [**PriseRdvController.java**] :



```
<AnchorPane prefHeight="600.0" prefWidth="400.0"
xmlns="http://javafx.com/javafx/20.0.1"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="controller.PriseRdvController">
```

1.3.2. Création et définition des différents contrôles

La fenêtre a la structure suivante :

Remarques

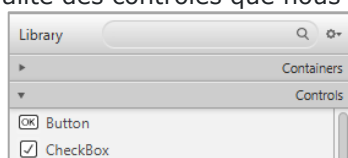
1. Etiquette Titre de la fenêtre
2. Choix « **Monsieur** »
3. Choix « **Madame** »
4. Etiquette « **Nom :** »
5. Zone de saisie du nom
6. Etiquette « **Prénom :** »
7. Zone de saisie du prénom
8. Etiquette « **n° de SS** »
9. Zone de saisie du n° de SS
10. Etiquette « **Sujet :** »
11. Zone de saisie du sujet
12. Case à cocher « **Réponse demandée** »
13. Etiquette « **Texte de la réponse** »
14. Etiquette « **Date :** »
15. Calendrier
16. Etiquette « **De :** »
17. Choix heure de début
18. Choix minute de début
19. Etiquette « **A :** »
20. Choix heure de fin
21. Choix Minute de fin
22. Etiquette « **Commentaires** »
23. Zone de saisie du commentaire
24. Zone d'affichage des erreurs de saisie
25. Bouton « **Valider** »

The screenshot shows a reservation form titled "Prise de Rdv". It includes fields for name (Nom, Prénom), SS number (n° de SS), subject (Sujet), and a checkbox for "Réponse demandée". There are also date and time pickers (De, A) and a large text area for "Commentaires". A "Valider" button is at the bottom right. Numbered callouts 1 through 25 identify specific UI elements as listed in the remarks.

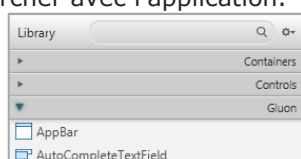
Explications :



L'intégralité des contrôles que nous allons utiliser se trouve dans le menu [**Controls**] de SceneBuilder



En aucun cas dans le menu [**Gluon**] dans lequel il y a des contrôles similaires mais qui risquent de ne pas marcher avec l'application.

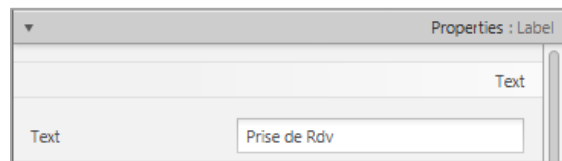


Garder toujours en tête que votre code peut être relu, modifié, etc... par un autre programmeur. Vos variables, méthodes, etc. se doivent d'avoir un nom explicite et qui respecte les normes de programmation.

1. Etiquette Titre de la fenêtre :

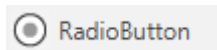
Le contrôle qui permet ce traitement est le contrôle [**Label**]

Cette information est fixe et ne sera pas modifiée par l'application, on ne modifie donc que la propriété [**Text**]

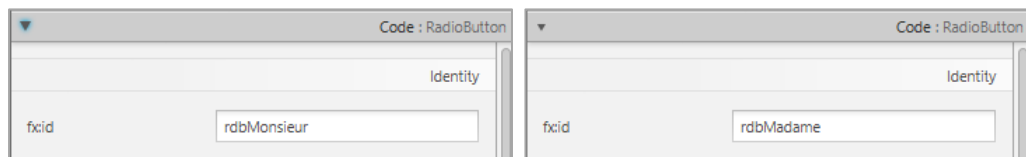


2. Choix « **Monsieur** » et 3. Choix « **Madame** »

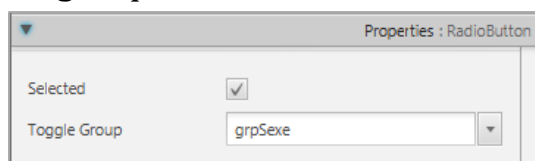
Le contrôle qui permet ce traitement est le contrôle [**radioButton**]



Ces contrôles seront gérés par l'application, il faut donc leur donner un **fx:id**. En respectant la logique de nommage des contrôles JavaFX :



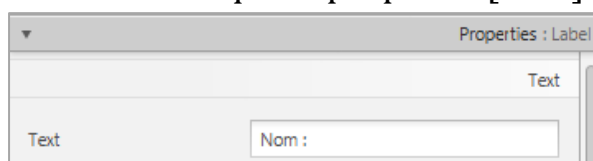
Petite particularité de ce contrôle, il faut que les deux appartiennent au même [**ToggleGroup**] pour pouvoir proposer un seul choix actif. Il faut donc rajouter un groupe commun sur les deux contrôles.



4. Etiquette « **Nom :** »

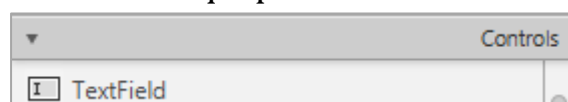
Le contrôle qui permet ce traitement est le contrôle [**Label**]

Cette information est fixe et ne sera pas modifiée par l'application, on ne modifie donc que la propriété [**Text**]

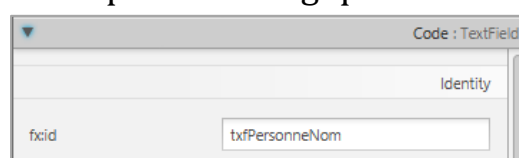


5. Zone de saisie du nom.

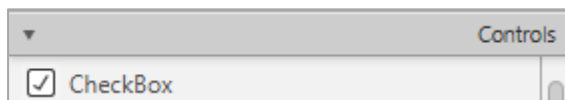
Le contrôle qui permet la saisie est le contrôle [**TextField**]



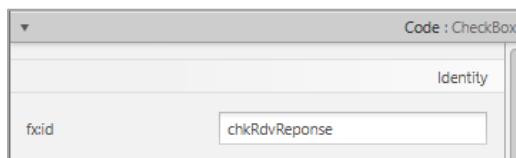
Ce contrôle sera géré par l'application, il faut donc lui donner un **fx:id**. En respectant la logique de nommage des contrôles JavaFX :



6. La Etiquette bel « **Prénom :** »
même logique que pour l'étiquette «**Nom :** » (4.)
7. Zone de saisie du prénom
même logique que pour la zone de saisie du nom (5.)
avec comme **fx:id** spécifique : **txfPersonPrenom**
8. Etiquette « **n° de SS** »
même logique que pour l'étiquette «**Nom :** » (4.)
9. Zone de saisie du n° de SS
même logique que pour la zone de saisie du nom (5.)
avec comme **fx:id** spécifique : **txfPersonNumSs**
10. Etiquette « **Sujet :** »
même logique que pour l'étiquette «**Nom :** » (4.)
11. Zone de saisie du sujet
même logique que pour la zone de saisie du nom (5.)
avec comme **fx:id** spécifique : **txfRdvSujet**
12. Case à cocher « **Réponse demandée** »
Le contrôle qui propose un choix (sélectionné/non-sélectionné) est le contrôle [**TextField**]

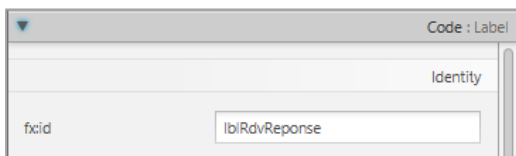


Ce contrôle sera géré par l'application, il faut donc lui donner un **fx:id**.
En respectant la logique de nommage des contrôles JavaFX :

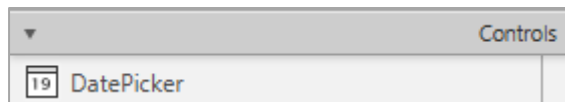


Par défaut ce contrôle est décoché. Le texte affiché est à saisir dans la propriété [**Text**]

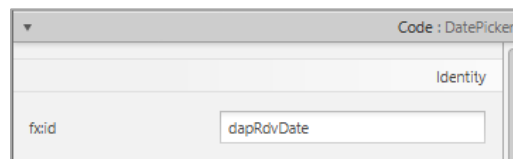
13. Etiquette «**Texte de la réponse**»
Petite particularité pour ce contrôle [**Label**], il va être géré par l'application (modification du texte, visibilité). il faut donc lui donner un **fx:id**.
En respectant la logique de nommage des contrôles JavaFX :



14. Etiquette « **Date :** »
même logique que pour l'étiquette «**Nom :** » (4.)
15. Calendrier
Le contrôle qui propose un choix de date à partir d'un calendrier est le contrôle [**DatePicker**]



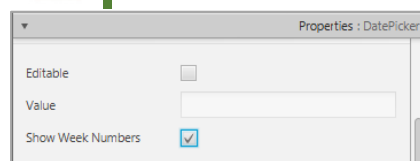
Ce contrôle sera géré par l'application, il faut donc lui donner un **fx:id**.
En respectant la logique de nommage des contrôles JavaFX :



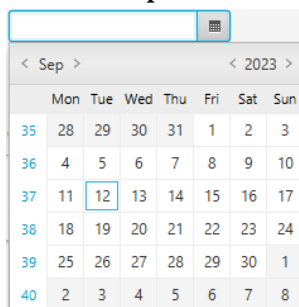
Un contrôle [**DatePicker**] propose à l'utilisateur de sélectionner une date ou de la saisir. Nous n'allons pas lui laisser cette seconde possibilité. Nous allons rendre le contrôle **non-editable**.



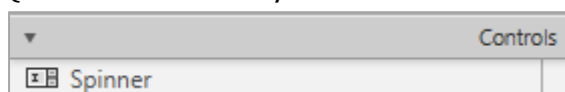
La logique est toujours la même, si l'utilisateur choisit simplement au lieu de la saisir, nous n'aurons pas à contrôler la validité de cette date.



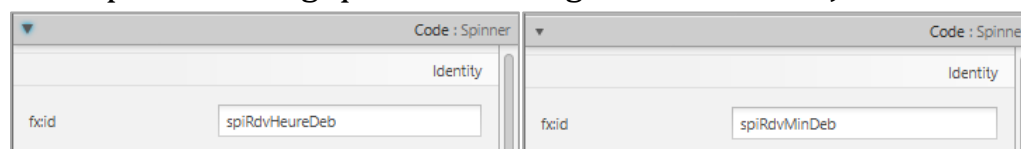
Nous en profitons également pour faire afficher les n° de semaine.



16. Etiquette « **De :** »
même logique que pour l'étiquette « **Nom :** » (4.)
17. Choix heure de début et 18. Choix minute de début
Le contrôle qui propose un choix de valeurs avec des boutons de déplacement (incrémentement / décrémentation dans notre cas) est le contrôle [**Spinner**]



Ce contrôle sera géré par l'application, il faut donc lui donner un **fx:id**.
En respectant la logique de nommage des contrôles JavaFX :



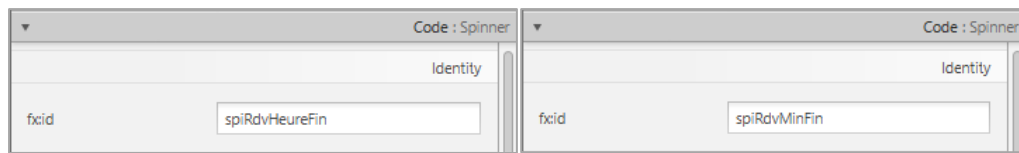
Les modalités de remplissages d'un contrôle [Spinner] sont expliquées dans la documentation technique du contrôle.

19. Etiquette « **A :** »
même logique que pour l'étiquette « **Nom :** » (4.)

20. Choix heure de fin et 21. Choix Minute de fin

La logique est exactement la même que pour le point (17.)

Avec les **fx:id** suivants :

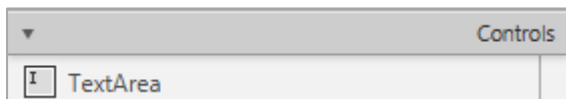


22. Etiquette «Commentaires»

même logique que pour l'étiquette «**Nom** : » (4.)

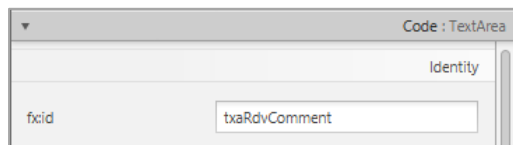
23. Zone de saisie du commentaire

Ici la saisie peut se faire sur plusieurs lignes, le contrôle qui permet de faire cela est le contrôle [**TextArea**].



Ce contrôle sera géré par l'application, il faut donc lui donner un **fx:id**.

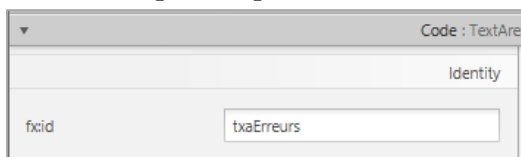
En respectant la logique de nommage des contrôles JavaFX :



24. Zone d'affichage des erreurs de saisie

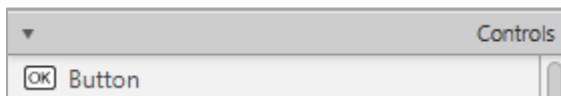
Cette zone doit permettre d'afficher les messages d'erreur lors du contrôle de la fenêtre, l'affichage se fera donc sur plusieurs lignes.

La logique est exactement la même que pour le point (23.) avec, évidemment, un **fx:id** spécifique :



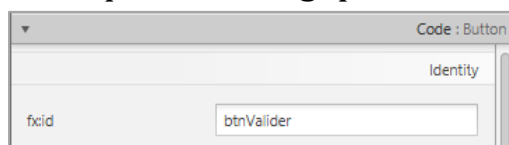
25. Bouton « Valider »

Comme son nom l'indique, il s'agit d'un bouton (contrôle [**Button**]).



Ce contrôle sera géré par l'application, il faut donc lui donner un **fx:id**.

En respectant la logique de nommage des contrôles JavaFX :



Ce contrôle va nous permettre de déclencher un évènement (*validation de la fenêtre entre autres*). Nous allons donc devoir rajouter un évènement. Cet évènement se déclenchera lors d'un clic souris sur le bouton.

En respectant la logique de nommage des événements en JavaFX :




Sauvegardez votre fenêtre et contrôlez bien que le fichier FXML a bien été mis à jour dans l'éditeur.

1.4. Structure de la Première fenêtre – PriseRdvController.java

1.4.1. Gestion de la déclaration des contrôles fxml dans le contrôleur

Maintenant que le fichier FXML est correctement créé, il faut déclarer les contrôles dont nous aurons besoin dans le contrôleur. C'est assez simple :

1.4.1.1. Contrôles des fx:id dans le fichier fxml :



```
<RadioButton fx:id="rdbMonsieur" layoutX="36.0" layoutY="83.0" mnemonicParsing="false" selected="true"
text="Monsieur">
<ToggleGroup fx:id="grpSexe" />
<RadioButton fx:id="rdbMadame" layoutX="124.0" layoutY="83.0" mnemonicParsing="false" text="Madame"
toggleGroup="$grpSexe" />
<TextField fx:id="txfPersonneNom" GridPane.columnIndex="1" />
<TextField fx:id="txfPersonPrenom" GridPane.columnIndex="1" GridPane.rowIndex="1" />
<TextField fx:id="txfPersonNumSs" GridPane.columnIndex="1" GridPane.rowIndex="2" />
<TextField fx:id="txfRdvSujet" layoutX="36.0" layoutY="239.0" prefHeight="25.0" prefWidth="339.0" />
<CheckBox fx:id="chkRdvReponse" layoutX="35.0" layoutY="273.0" mnemonicParsing="false" text="Réponse
demandée" />
<DatePicker fx:id="dapRdvDate" editable="false" layoutX="35.0" layoutY="318.0" showWeekNumbers="true" />
<Spinner fx:id="spiRdvHeureDeb" layoutX="67.0" layoutY="350.0" prefWidth="75.0" />
<Spinner fx:id="spiRdvMinDeb" layoutX="159.0" layoutY="350.0" prefWidth="75.0" />
<Spinner fx:id="spiRdvHeureFin" layoutX="67.0" layoutY="379.0" prefWidth="75.0" />
<Spinner fx:id="spiRdvMinFin" layoutX="159.0" layoutY="379.0" prefWidth="75.0" />
<TextArea fx:id="txaRdvComment" layoutX="20.0" layoutY="432.0" prefHeight="68.0" prefWidth="366.0" />
<Button fx:id="btnValider" layoutX="166.0" layoutY="561.0" mnemonicParsing="false"
onMouseClicked="#evtOnMouseClickedBtnValider" text="Valider" />
<Label fx:id="lblRdvReponse" layoutX="183.0" layoutY="273.0" prefHeight="17.0" prefWidth="191.0" text="Texte
de la réponse" />
<TextArea fx:id="txaErreurs" editable="false" layoutX="21.0" layoutY="500.0" prefHeight="59.0"
prefWidth="366.0" />
```

1.4.1.2. Déclaration de ces mêmes contrôles dans le fichier java :



```
/** Contrôles de la fenêtre */
@FXML private TextField txfPersonneNom;
@FXML private TextField txfPersonPrenom;
@FXML private TextField txfPersonNumSs;
@FXML private TextField txfRdvSujet;
@FXML private RadioButton rdbMonsieur;
@FXML private RadioButton rdbMadame;
@FXML private CheckBox chkRdvReponse;
@FXML private DatePicker dapRdvDate;
@FXML private Spinner<Integer> spiRdvHeureDeb;
@FXML private Spinner<Integer> spiRdvHeureFin;
@FXML private Spinner<Integer> spiRdvMinDeb;
@FXML private Spinner<Integer> spiRdvMinFin;
@FXML private TextArea txaRdvComment;
@FXML private TextArea txaErreurs;
@FXML private Button btnValider;
@FXML private Label lblRdvReponse;
```


Dans ce contrôleur, nous n'aurons besoin que de deux méthodes :

1. La méthode **initialize()** pour initialiser les différents contrôles,
2. La méthode **evtOnMouseClickedBtnValider()** pour gérer le clic sur la souris.

1.4.1.3. Déclaration des variables globales dans le fichier java :

Nous allons avoir besoin de variables globales dans notre classe java, essentiellement pour gérer les contrôles [**Spinner**].

Nous allons également rajouter une constante pour gérer les retours à la ligne lors de l'affichage des messages d'erreur.




```
/** Constantes */
private static final String CR = "\n";
/** ----- Attributs de la classe ----- */
/** 1. Listes contenant les informations des Spinners */
private ObservableList<Integer> listeHeure = FXCollections.observableArrayList();
private ObservableList<Integer> listeMinute = FXCollections.observableArrayList();
/** 2. Value des différents Spinners */
private SpinnerValueFactory<Integer> valueFactoryHeureDebut = null;
private SpinnerValueFactory<Integer> valueFactoryHeureFin = null;
private SpinnerValueFactory<Integer> valueFactoryMinuteDebut = null;
private SpinnerValueFactory<Integer> valueFactoryMinuteFin = null;
```

1.4.1.4. Descriptif de la méthode initialize()

Cette méthode est appelée « *en première position* » lors de l'appel de notre fenêtre par le FXMLoader.

Généralement, elle contient toutes les phases d'initialisation des différents composants spécifiques (ComboBox, Spinner, DatePicker, etc...). Cette initialisation n'ayant pas besoin d'être refaite ensuite dans le code.



```
/** Méthode initialize */
@FXML private void initialize() {
    /** Initialisation du DatePicker à la date du jour */
    dapRdvDate.setValue(LocalDate.now());
    /** Initialisation de l'heure de début */
    int heureDebut = LocalTime.now().getHour();
    if(heureDebut<8) heureDebut=8;
    /** ----- Initialisation des Spinners ----- */
    /** Les heures doivent être comprises entre 8h et 18h */
    /** Les minutes doivent être comprises entre 0 et 55 par pas de 5 */
    /** 1. Initialisation des listes contenant les valeurs à afficher */
    for(int i = 8; i <= 18; i++) listeHeure.add(i);
    for(int i = 0; i < 60; i+=5) listeMinute.add(i);
    /** 2. Affectation des ObservableListe aux différents objets ValueFactory */
    valueFactoryHeureDebut = new SpinnerValueFactory.ListSpinnerValueFactory<Integer>(listeHeure);
    valueFactoryHeureFin = new SpinnerValueFactory.ListSpinnerValueFactory<Integer>(listeHeure);
    valueFactoryMinuteDebut = new SpinnerValueFactory.ListSpinnerValueFactory<Integer>(listeMinute);
    valueFactoryMinuteFin = new SpinnerValueFactory.ListSpinnerValueFactory<Integer>(listeMinute);
    /** 3. Affectation des ValueFactory aux contrôles Spinner */
    valueFactoryHeureDebut.setValue(heureDebut);
    valueFactoryHeureFin.setValue(heureDebut);
    spiRdvHeureDeb.setValueFactory(valueFactoryHeureDebut);
    spiRdvHeureFin.setValueFactory(valueFactoryHeureFin);
    spiRdvMinDeb.setValueFactory(valueFactoryMinuteDebut);
    spiRdvMinFin.setValueFactory(valueFactoryMinuteFin);
    /** La zone d'affichage des erreurs doit être non visible */
    txaErreurs.setVisible(false);
    /** Le texte du Label réponse est vidé - on peut aussi gérer la propriété Visible */
    lblRdvReponse.setText("");
}
```

1.4.1.5. Descriptif de la méthode evtOnMouseClickedBtnValider ()

Cette méthode est appelée lorsque l'utilisateur clique sur le bouton [valider]. Elle va se constituer de deux parties :

- La première partie contrôlant les valeurs obligatoires et affichant un message d'erreur si besoin

```
@FXML private void evtOnMouseClickedBtnValider() {
    /** Initialisation d'une variable de type String qui contiendra les messages d'erreur */
    String messageErreur = "";
    /** Contrôle du nom */
    if(txfPersonneNom.getText().isBlank()) messageErreur = "Le nom est obligatoire" + CR;
    /** Contrôle du prénom */
    if(txfPersonPrenom.getText().isBlank()) messageErreur += "Le prénom est obligatoire" + CR;
    /** Contrôle du sujet */
    if(txfRdvSujet.getText().isBlank()) messageErreur += "Le sujet est obligatoire" + CR;
    /** Initialisation d'une variable de type LocalDate contenant la date du jour */
    LocalDate aujourd'hui = LocalDate.now();
    /** Contrôle de la date sélectionnée, elle ne peut être inférieure à la date du jour */
    if(dapRdvDate.getValue().isBefore(aujourd'hui)) messageErreur += "La date ne peut être avant aujourd'hui" + CR;
    /** Initialisation d'une variable contenant l'heure en cours */
    int heure = (LocalTime.now().getHour()) + 1;
    /** Si le Rdv a lieu, la date de début doit être supérieure à l'heure actuelle + 1h */
    if(dapRdvDate.getValue().isEqual(aujourd'hui) && spiRdvHeureDeb.getValue() < heure) {
        messageErreur += "Le rdv étant aujourd'hui l'heure saisie doit être supérieure à l'heure en cours" + CR;
    }
    /** L'heure de fin doit être supérieure à l'heure de début */
    if(spiRdvHeureFin.getValue() < spiRdvHeureDeb.getValue()) {
        messageErreur += "L'heure de fin doit être supérieure ou égale à l'heure de début";
    }
    if((spiRdvHeureFin.getValue() == spiRdvHeureDeb.getValue()) &&
        (spiRdvMinFin.getValue() >= spiRdvMinDeb.getValue())) {
        messageErreur += "Le Rdv étant sur la même heure, les minutes de fin doivent être supérieures";
    }
    /** S'il y a au moins un message, on l'affiche */
    if(!messageErreur.isBlank()) {
        txaErreurs.setVisible(true);
        txaErreurs.setText(messageErreur);
    } else {
        ....
    }
}
```



- La seconde créant un objet de type Rdv.

```
    } else {
        int          rdvPersonneCivilite= (rdbMonsieur.isSelected())? 1 : 2);
        String       rdvPersonneNom    = txfPersonneNom.getText();
        String       rdvPersonnePrenom = txfPersonPrenom.getText();
        String       rdvPersonNumSs    = txfPersonNumSs.getText();
        String       rdvSujet          = txfRdvSujet.getText();
        boolean      rdvReponseDemandee = chkRdvReponse.isSelected();
        LocalDate    rdvDate           = dapRdvDate.getValue();
        int          rdvHeureDebut      = spiRdvHeureDeb.getValue();
        int          rdvMinuteDebut     = spiRdvMinDeb.getValue();
        int          rdvHeureFin        = spiRdvHeureFin.getValue();
        int          rdvMinuteFin       = spiRdvMinFin.getValue();
        String       rdvCommentaires   = txaRdvComment.getText();
        Rdv          rdv               = new Rdv(rdvPersonneCivilite, rdvPersonneNom, rdvPersonnePrenom, rdvPersonNumSs,
rdvSujet, rdvReponseDemandee, rdvDate, rdvHeureDebut, rdvMinuteDebut, rdvHeureFin, rdvMinuteFin, rdvCommentaires);
    }
```



1.5. Structure de la Seconde fenêtre – ConfirmationRdv.fxml

Cette fenêtre doit être entièrement créée.

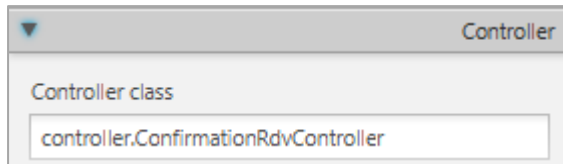
Rajoutez un fichier **ConfirmationRdv.fxml** dans le package [view]

Rajoutez un fichier **ConfirmationRdvController.java** dans le package [controller]

1.5.1. Contrôle du contrôleur de la page

Contrôlez dans le fichier FXML ou dans SceneBuilder qu'elle a comme contrôleur

- la classe [ConfirmationRdvController.java] :



```
<AnchorPane prefHeight="600.0" prefWidth="400.0"
xmlns:fx="http://javafx.com/fxml/1"
xmlns="http://javafx.com/javafx/20.0.1"
fx:controller="controller.ConfirmationRdvController">
```

1.5.2. Création et définition des différents contrôles

La fenêtre a la structure suivante :

A diagram of the 'Demande de Rdv' window. It shows a layout with various controls numbered 1 through 14. 1: Title bar. 2: Message label. 3: Person label. 4: Date label. 5: Subject label. 6: Comment label. 7: Comment text area. 8: Response label. 9: Response text area. 10: Subject label. 11: Subject text area. 12: Error message label. 13: 'Répondre' button. 14: 'Valider' button.

Remarques

1. Etiquette Titre de la fenêtre
2. Etiquette message « **Bonjour, vous** ... »
3. Etiquette de la personne
4. Etiquette date du rdv
5. Etiquette message « **sur le sujet...** »
6. Etiquette sujet
7. Etiquette « **Commentaire...** »
8. Zone de saisie du commentaire
9. Etiquette « **Réponse à apporter...** »
10. Zone de saisie de la réponse
11. Zone de saisie du sujet
12. Zone d'affichage des erreurs de saisie
13. Bouton « **Répondre** »
14. Bouton « **Valider** »

En gardant la même logique que la première fenêtre, les contrôles [Label] :1,2,5,7 sont de simples étiquettes, nous n'avons pas besoin de leur affecter un **fx:id**.

Les autres contrôles seront gérés par l'application, il faut donc leur donner un **fx:id**.

1.6. Structure de la Première fenêtre – PriseRdvController.java

1.6.1. Gestion de la déclaration des contrôles fxml dans le contrôleur

Maintenant que le fichier FXML est correctement créé, il faut déclarer les contrôles dont nous aurons besoin dans le contrôleur. C'est assez simple :

1.6.1.1. Contrôles des fx:id dans le fichier fxml :



```
<Label fx:id="lblTitre" layoutX="154.0" layoutY="14.0" text="Demande de Rdv" />
<Label fx:id="lblRdvPersonne" layoutX="33.0" layoutY="96.0" prefWidth="335.0" text="Label" />
<Label fx:id="lblRdvDateHeures" layoutX="46.0" layoutY="124.0" prefHeight="17.0" prefWidth="320.0"
text="Label" />
<Label fx:id="lblRdvSujet" layoutX="32.0" layoutY="177.0" prefHeight="17.0" prefWidth="335.0" text="Label"
/>
<TextArea fx:id="txaCommentaires" editable="false" layoutX="32.0" layoutY="219.0" prefHeight="87.0"
prefWidth="335.0" />
<Label fx:id="lblReponseTitre" layoutX="32.0" layoutY="318.0" prefHeight="17.0" prefWidth="335.0"
text="Réponse a apporter à la demande" />
<TextField fx:id="txfReponse" layoutX="32.0" layoutY="335.0" prefHeight="25.0" prefWidth="335.0" />
<Button fx:id="btnValider" layoutX="192.0" layoutY="506.0" mnemonicParsing="false"
onMouseClicked="#evtOnMouseClickedBtnValider" text="Valider" />
<Button fx:id="btnRepondre" layoutX="113.0" layoutY="506.0" mnemonicParsing="false"
onMouseClicked="#evtOnMouseClickedBtnRepondre" text="Répondre" />
<TextArea fx:id="txaErreurs" editable="false" layoutX="33.0" layoutY="366.0" prefHeight="108.0"
prefWidth="335.0" />
```

1.6.1.2. Déclaration de ces mêmes contrôles dans le fichier java :



```
/** Contrôles de la fenêtre */
@FXML private Label      lblRdvPersonne;
@FXML private Label      lblRdvDateHeures;
@FXML private Label      lblRdvSujet;
@FXML private Label      lblTitre;
@FXML private TextArea    txaCommentaires;
@FXML private TextArea    txaErreurs;
@FXML private Label      lblReponseTitre;
@FXML private TextField    txfReponse;
@FXML private Button      btnValider;
@FXML private Button      btnRepondre;
```

Dans ce contrôleur, nous aurons besoin des méthodes suivantes :

- La méthode **initialize()** pour initialiser les différents contrôles,
- La méthode **evtOnMouseClickedBtnValider()** pour gérer le clic de la souris sur le bouton [Valider],
- La méthode **evtOnMouseClickedBtnRepondre ()** pour gérer le clic de la souris sur le bouton [Valider],
- La méthode **setRdv(Rdv rdv)** qui va permettre de recevoir le Rdv saisi dans la première fenêtre,
- La méthode **setLblTitre(String titre)** qui va permettre de recevoir le titre depuis la première fenêtre,
- La méthode **getTxfReponse()** qui va permettre de renvoyer la réponse à la première fenêtre,

1.6.1.3. Descriptif de la méthode setLblTitre(String titre)

Cette méthode va permettre à la première fenêtre de passer en paramètre le titre à la seconde fenêtre (elle est équivalente à un Setter).



```
public void setLblTitre(String titre) {  
    lblTitre.setText(titre);  
}
```

1.6.1.4. Descriptif de la méthode setRdv(Rdv rdv)

Cette méthode va permettre à la première fenêtre de passer en paramètre un objet Rdv à la seconde fenêtre (elle est équivalente à un Setter). Elle va aussi afficher les différentes informations sur cette fenêtre.



```
public void setRdv(Rdv rdv) {  
    /** Variables **/  
    String civilite = "";  
    String personne = "";  
    this.rdv = rdv;  
    /** Texte sur la personne **/  
    Civilite = ((this.rdv.getRdvPersonneCivilite()==1)? "Monsieur" : "Madame");  
    Personne = civilite + " " + this.rdv.getRdvPersonnePrenom() + " " + this.rdv.getRdvPersonneNom();  
    lblRdvPersonne.setText(personne);  
    /** Texte sur la date du Rdv **/  
    String dateFormate = this.rdv.getRdvDate().format(DateTimeFormatter.ofLocalizedDate(FormatStyle.FULL));  
    String heureDebFormate = Integer.toString(this.rdv.getRdvHeureDebut()) + ":" +  
    Integer.toString(this.rdv.getRdvMinuteDebut());  
    String heureFinFormate = Integer.toString(this.rdv.getRdvHeureFin()) + ":" +  
    Integer.toString(this.rdv.getRdvMinuteFin());  
    lblRdvDateHeures.setText(dateFormate + " de " + heureDebFormate + " à " + heureFinFormate);  
    /** Texte sur le sujet **/  
    lblRdvSujet.setText(this.rdv.getRdvSujet());  
    /** Texte sur le commentaire **/  
    txCommentaires.setText(this.rdv.getRdvCommentaires());  
    /** Gestion de l'affichage des boutons et des zones réponses **/  
    lblReponseTitre.setVisible(this.rdv.isRdvReponseDemandee());  
    txfReponse.setVisible(this.rdv.isRdvReponseDemandee());  
    btnRepondre.setVisible(this.rdv.isRdvReponseDemandee());  
    btnValider.setVisible(!this.rdv.isRdvReponseDemandee());  
}
```

1.6.1.5. Descriptif de la méthode getTxfReponse()

Cette méthode va permettre à la renvoyer la réponse à la première fenêtre (elle est équivalente à un Getter).



```
public String getTxfReponse() {  
    return txfReponse.getText();  
}
```

1.6.1.6. Descriptif de la méthode evtOnMouseClickedBtnValider ()

Cette méthode va permettre de fermer la fenêtre dans le cas où le Rdv est validé.



```
@FXML private void evtOnMouseClickedBtnValider() {  
    Stage stage = (Stage) btnRepondre.getScene().getWindow();  
    stage.close();  
}
```

1.6.1.7. Descriptif de la méthode evtOnMouseClickedBtnRepondre ()

Cette méthode est appelée pour contrôler lors d'un clic sur le bouton [Répondre].si l'utilisateur a bien saisi une réponse. Si ce n'est pas le cas, la méthode affiche un message d'erreur sinon elle ferme la fenêtre.



```
@FXML private void evtOnMouseClickedBtnRepondre() {  
    String messageErreur = "";  
    if(txfReponse.getText().isBlank()) messageErreur = "Vous devez saisir une réponse ";  
    if(!messageErreur.isBlank()) {  
        txaErreurs.setVisible(true);  
        txaErreurs.setText(messageErreur);  
    } else {  
        evtOnMouseClickedBtnValider();  
    }  
}
```

1.6.1.7. Descriptif de la méthode initialize ()

Cette méthode est appelée « *en première position* » lors de l'appel de notre fenêtre par le FXMLLoader.

Généralement, elle contient toutes les phases d'initialisation des différents composants spécifiques (ComboBox, Spinner, DatePicker, etc...). Cette initialisation n'ayant pas besoin d'être refaite ensuite dans le code.



```
@FXML private void initialize() {  
    txaErreurs.setVisible(false);  
}
```

1.8. Modification du contrôleur – PriseRdvController.java afin de tenir compte des évolutions sur la seconde fenêtre

Maintenant que la seconde fenêtre est créée, il faut l'appeler dans le contrôleur de la première fenêtre. Nous allons modifier la méthode **evtOnMouseClickedBtnValider()** en rajoutant, après la création de l'objet Rdv, l'appel et le retour de la seconde fenêtre.

```
/** Appel de la fenêtre de confirmation */
try {
    FXMLLoader loader = new FXMLLoader();
    loader.setLocation(getClass().getResource("/view/ConfirmationRdv.fxml"));
    AnchorPane root = (AnchorPane) loader.load();
    Scene scene = new Scene(root);
    scene.getStylesheets().add(getClass().getResource("/style/application.css").toExternalForm());
    Stage primaryStage = new Stage();
    ConfirmationRdvController controller = loader.getController();
    primaryStage.setScene(scene);
    primaryStage.setTitle("Prise de Rdv");
    controller.setRdv(rdv);
    if(chkRdvReponse.isSelected()) {
        controller.setLblTitre("Demande de réponse à un Rdv");
    } else {
        controller.setLblTitre("Validation d'un Rdv");
    }
    primaryStage.showAndWait();
    if(chkRdvReponse.isSelected()) {
        lblRdvReponse.setText(controller.getTxfReponse());
    } else {
        btnValider.setVisible(false);
        txtaRdvComment.setText(txtaRdvComment.getText()+CR+"Le rendez a été validé");
    }
} catch (Exception e) {
    e.printStackTrace();
}
```

