

Utilisation de NetBeans pour les applications J2ME

Sommaire

1.	Introduction	2
2.	Présentation de l'EDI NetBeans	2
3.	Création d'une application en utilisant l'éditeur graphique	3
3.1.	Création d'une application MIDP : mode « Lazy Initialized »	3
3.2.	Adapter le code source.	5
3.3.	Compilation et exécution du projet	6
3.4.	Désactivation du mode « Lazy Initialized»	9
4.	Création d'une application MIDP en utilisant l'éditeur de code source.	12
4.1.	Création du projet	12
4.2.	Création du MIDlet	12
5.	Affichage de l'aide : javadocs	15
6.	Utilisation des tutoriaux.....	15
7.	Débogage	15

1. Introduction

Ce tutorial vous montrera les deux manières de créer un projet Java ME Midlet appelé `pjtMidletBonjour` qui affichera le texte « Bonjour d'Armentières » sur le simulateur.

2. Présentation de l'EDI NetBeans

Pré requis : Installation de NetBeans IDE 5.5 et de NetBeans Mobility Pack 5.5.

Projects :

- Choix du projet actif
- Visualisation des éléments Java du projet

Styles d'affichage :

- Source : codage Java
- Screen Design : conception d'un seul écran de l'IHM en utilisant la palette de composants
- Flow Design : conception de l'enchaînement des écrans de l'IHM en utilisant la palette de composants

Palette :

Composants graphiques disponibles

Inspector :

Visualisation de l'organisation des composants graphiques

Output :

Affichage des messages de construction du projet et des sorties console

Properties :

Visualisation et modification des propriétés du composant graphique

- Le choix du projet actif (pour l'exécution, le débogage, ...) s'effectue avec un clic droit sur le projet > Set Main Project. Le nom du projet s'affiche alors en caractère gras.

- Le Flow Design ne doit servir qu'à implémenter les différents écrans avec les flèches visualisant les passages de l'un à l'autre (grâce aux commandes associées).
- Le Screen Design permet de définir plus précisément les propriétés des composants déposés dans la / les fenêtre(s).
- L'affichage Source permet d'introduire votre propre code source en sachant que celui en surbrillance bleue n'est pas modifiable.

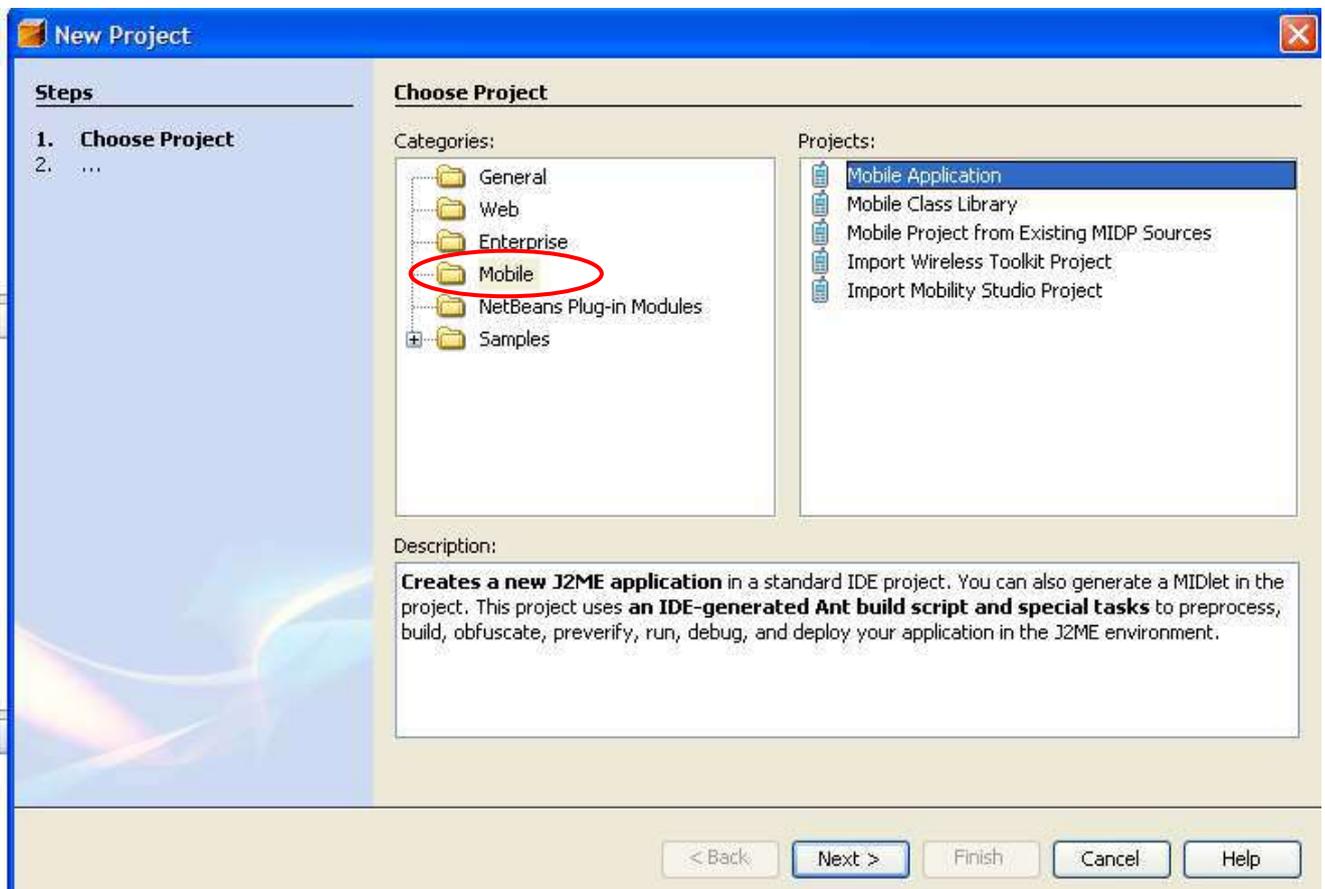
3. Création d'une application en utilisant l'éditeur graphique

NetBeans IDE vous propose un expert qui vous permet de créer rapidement un projet MIDP. Quand vous créez le projet, vous devez choisir de développer une application en utilisant l'éditeur graphique (Visual Mobil Designer) ou l'éditeur de code source (Source Code Editor).

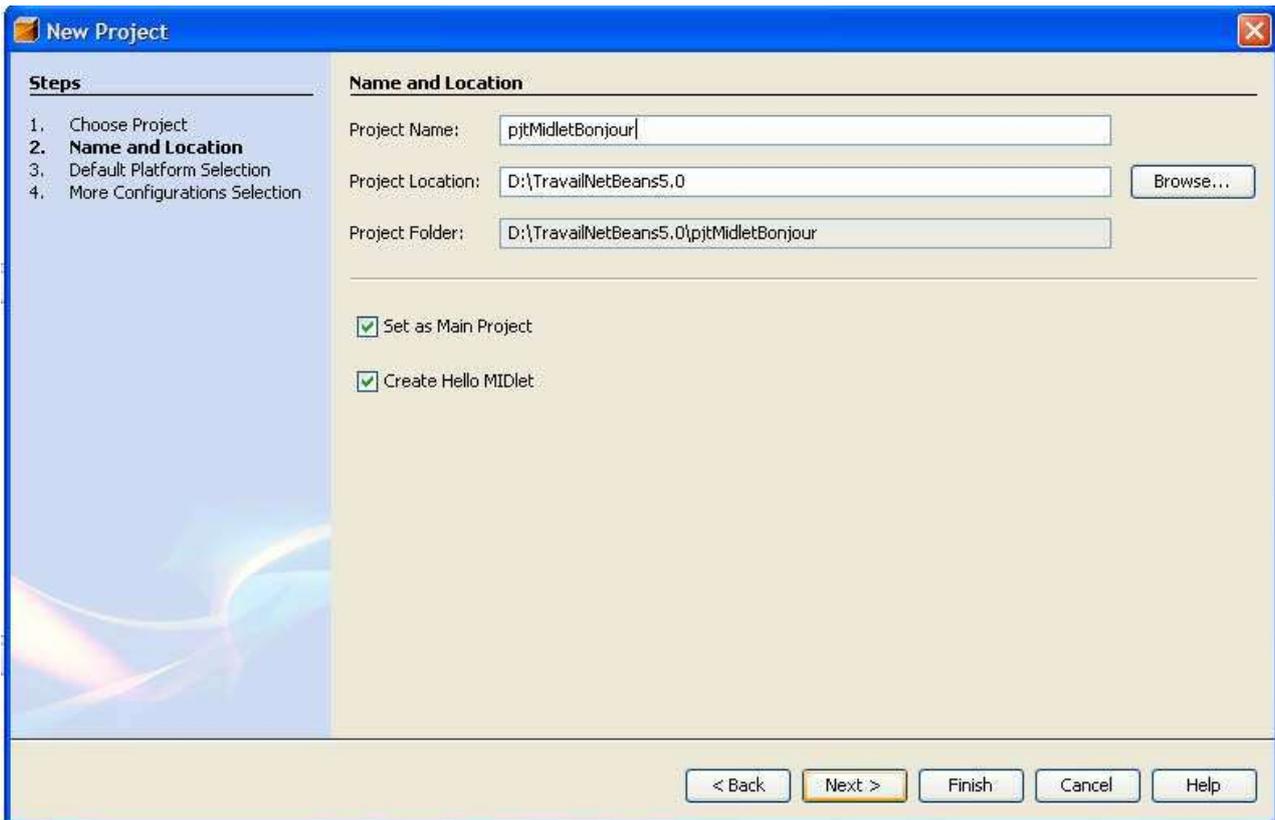
L'éditeur graphique vous permet de dessiner les flots et les écrans utilisés par votre application. Le code source de votre application est créé automatiquement.

3.1. Création d'une application MIDP : mode « Lazy Initialized »

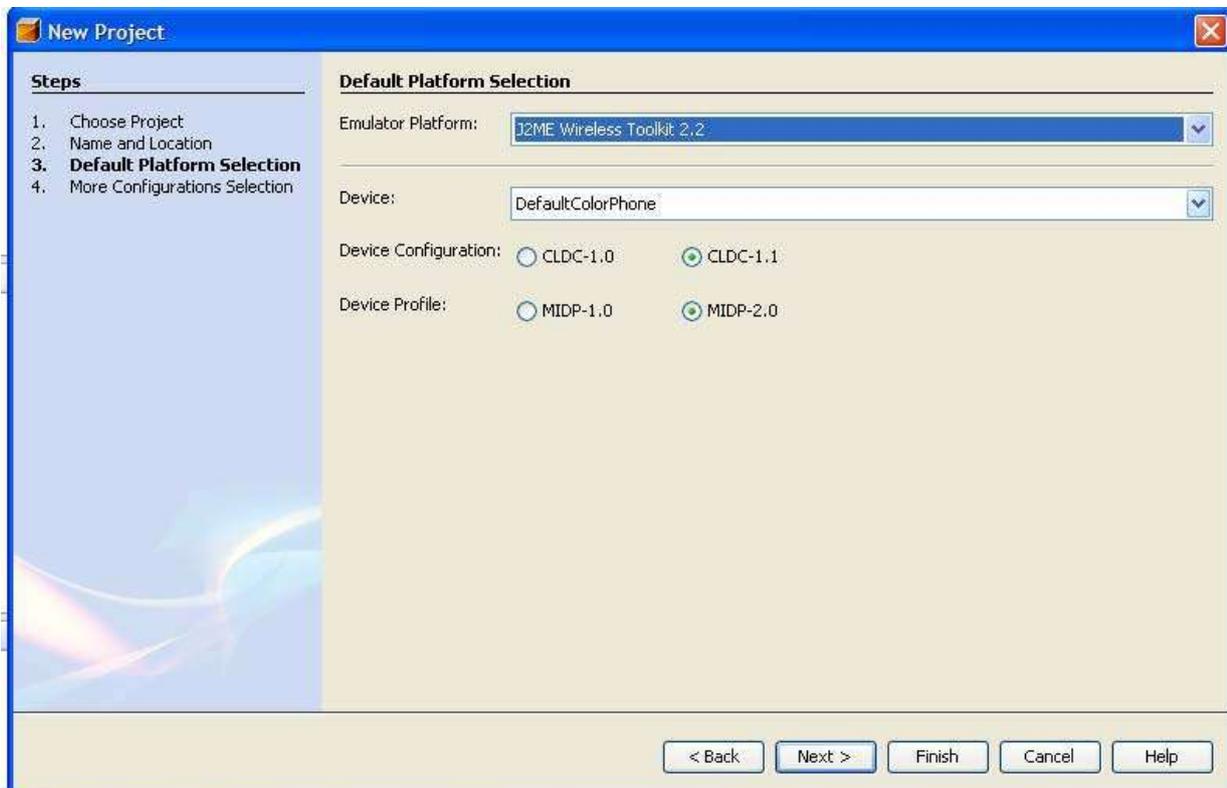
Choisissez File > New Project (Ctrl-Shift-N).



Dans la liste box, choisissez Mobile et Mobile Application, puis Next

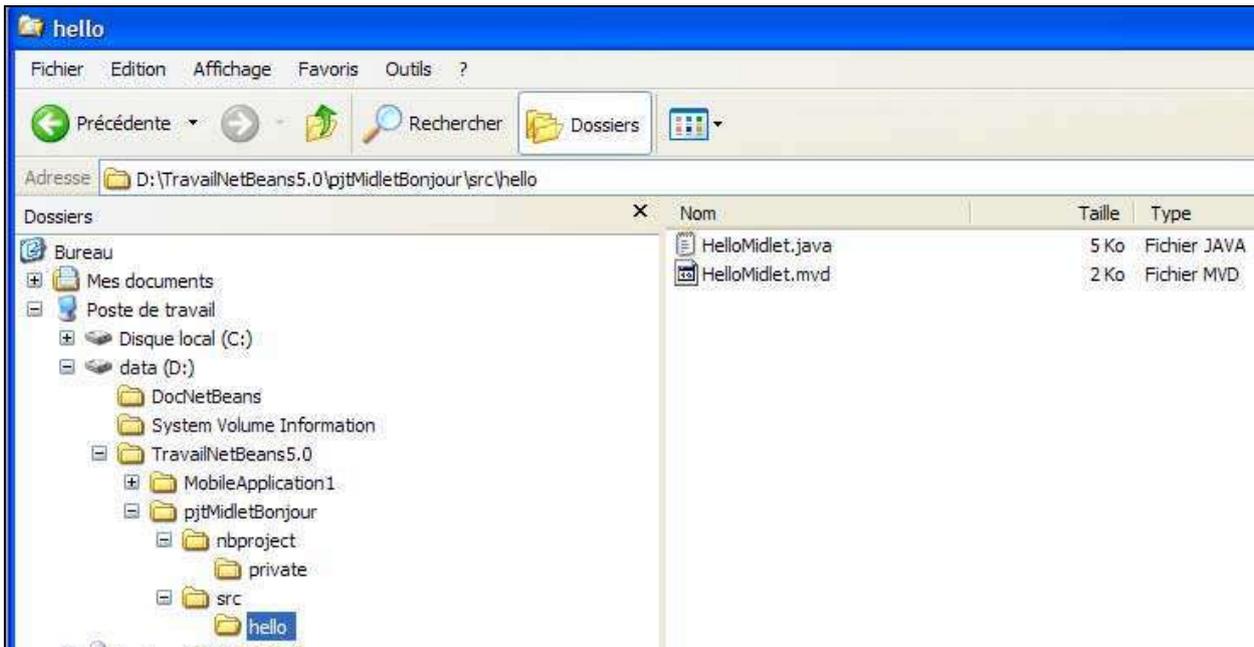


Entrez le nom du projet pjMidletBonjour et choisissez votre répertoire de sauvegarde. Validez les deux coches « Set as Main Project » et « Create Hello Midlet »

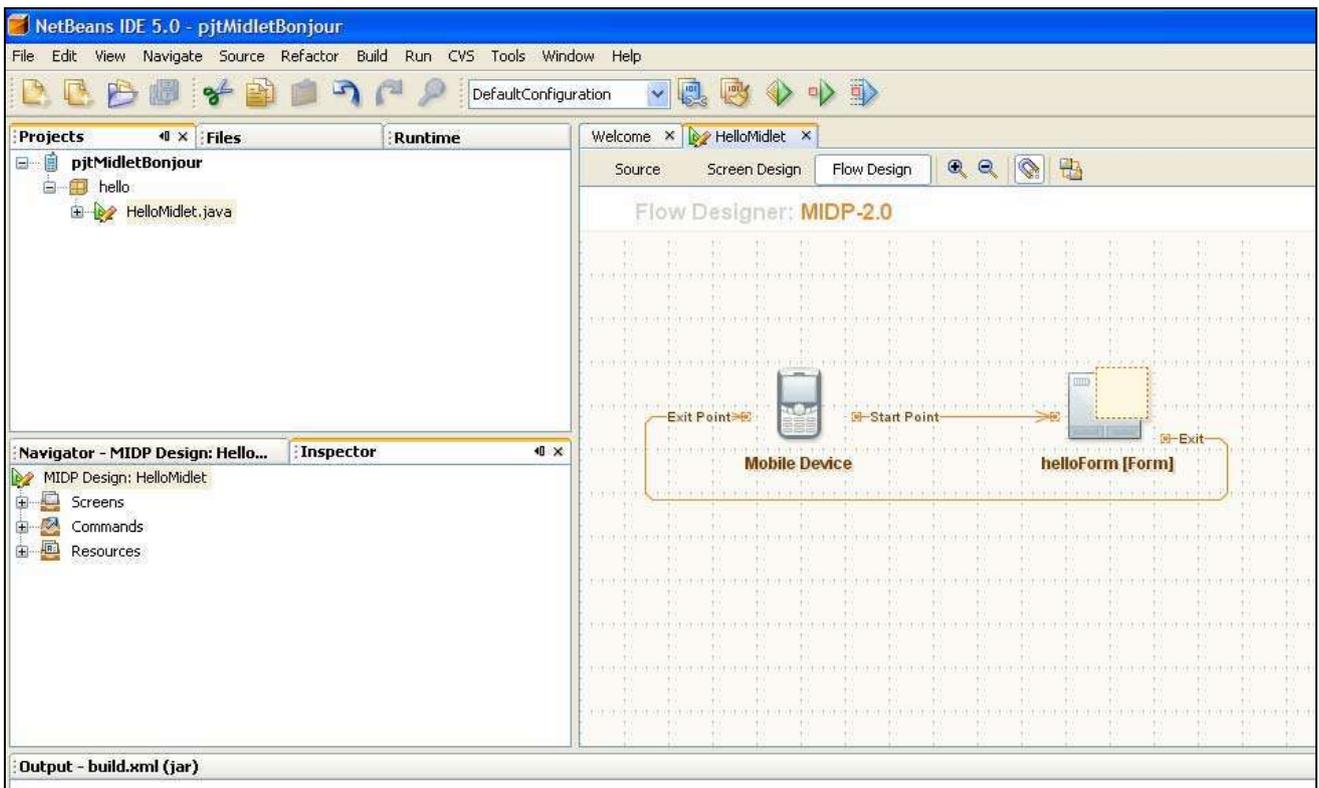


Gardez les sélections d'émulateur par défaut ainsi que la Configuration CLDC 1.1 et le Profile MIDP 2.0. Choisissez Finish pour clore la création du projet.

Vérifiez les répertoires et les fichiers créés :



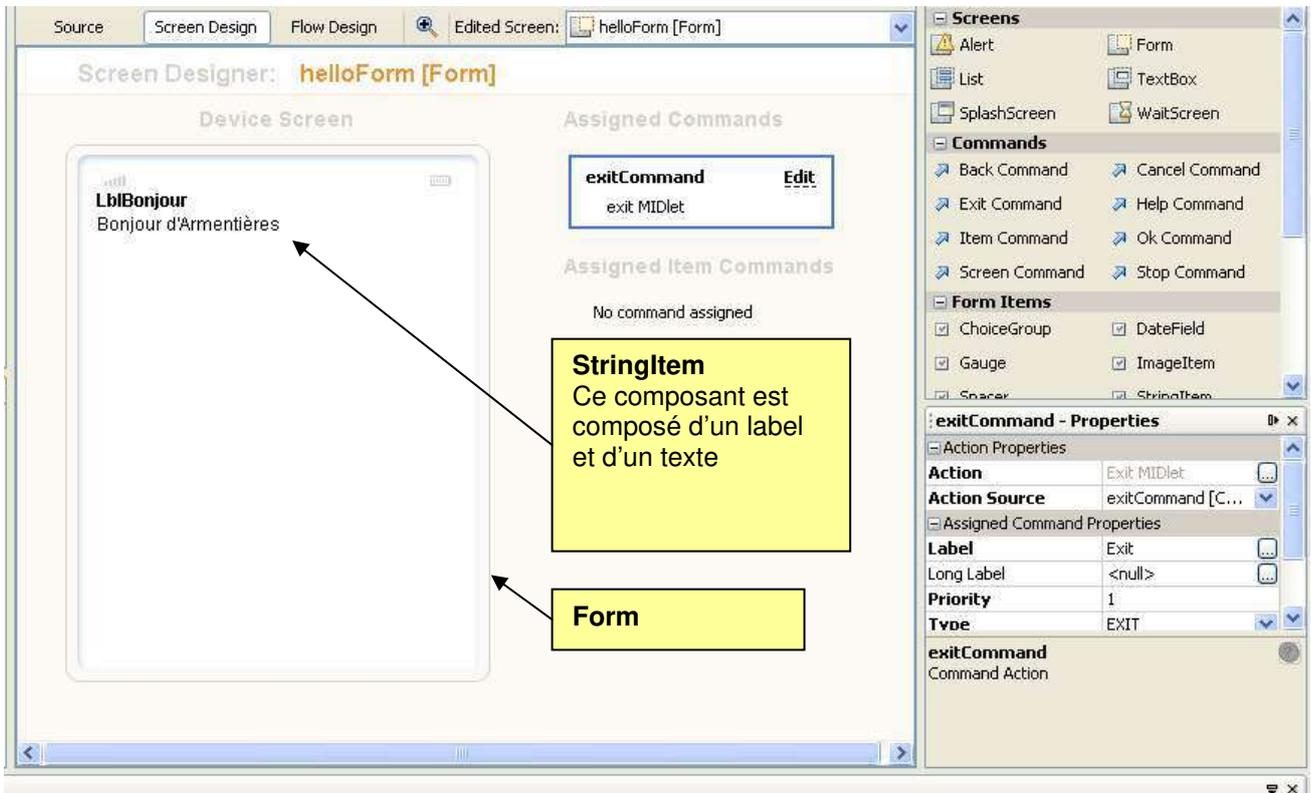
Voici une capture d'écran de NetBeans après la création de votre projet



3.2. Adapter le code source.

Nous allons éditer le code source créé pour le MIDlet.
Ouvrez le Screen Design
Affichez l'écran helloForm[Form]

Cliquez sur le composant StringItem (ce composant permet d'afficher du texte dans une Form). Modifiez sa propriété Text (Hello world !) par « Bonjour d'Armentières ». Modifiez également sa propriété Label (Hello) par « LblBonjour ».

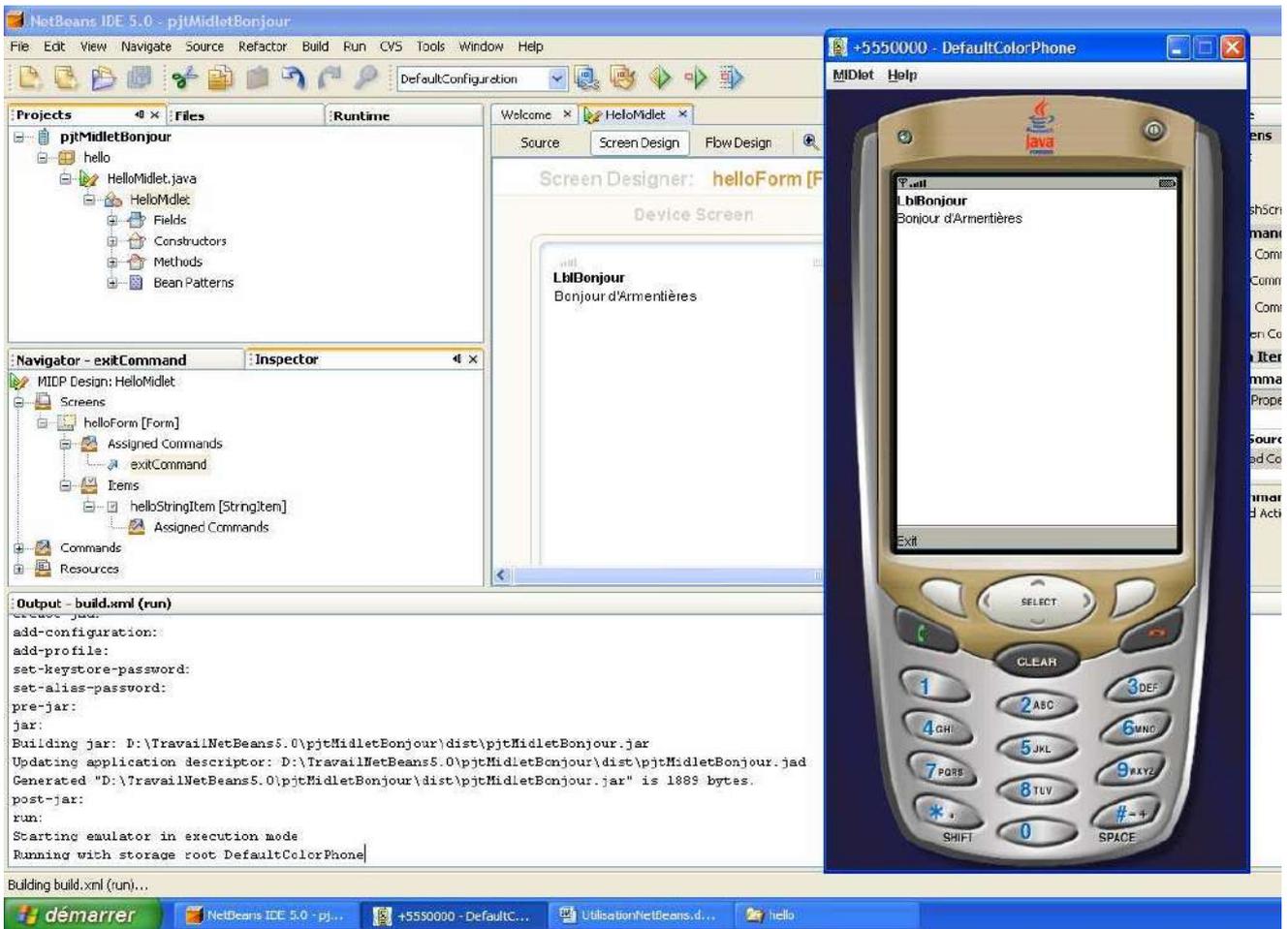


3.3. Compilation et exécution du projet

Choisissez Run > Run Main Project (F6) ou cliquez sur l'icône :

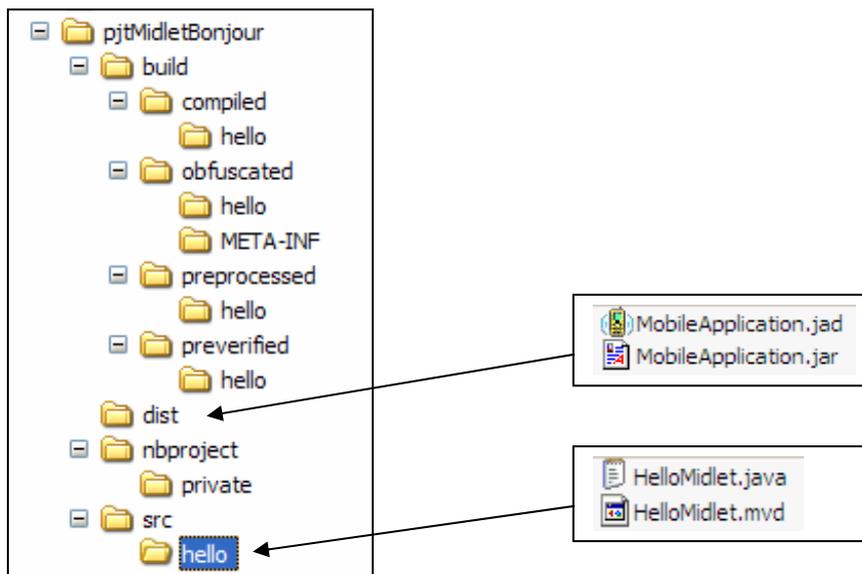


Agrandissez la fenêtre Output pour afficher correctement les messages de construction de votre projet. Un émulateur est lancé pour afficher le résultat de votre application. Le simulateur est « DefaultColorPhone ». Dans la fenêtre de l'émulateur, vous pouvez exécuter la commande Launch et Exit de la même manière que sur votre portable.



Appuyez sur Exit pour fermer le MIDlet. Puis cliquez sur le bouton de fermeture de la fenêtre du simulateur.

Voici l'arborescence des fichiers créés



Voici le code source généré par NetBeans :

```
/*
 * HelloMidlet.java
 *
 * Created on 18 octobre 2006, 15:30
 */

package hello;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author frederic
 */
public class HelloMidlet extends MIDlet implements CommandListener {

    /** Creates a new instance of HelloMidlet */
    public HelloMidlet() {

        private Form helloForm;
        private StringItem helloStringItem;
        private Command exitCommand;

        /** This method initializes UI of the application.
         */
        private void initialize() {
            // Insert pre-init code here
            getDisplay().setCurrent(get_helloForm());
            // Insert post-init code here
        }

        /** Called by the system to indicate that a command has been invoked on a particular displayable.
         * @param command the Command that ws invoked
         * @param displayable the Displayable on which the command was invoked
         */
        public void commandAction(Command command, Displayable displayable) {
            // Insert global pre-action code here
            if (displayable == helloForm) {
                if (command == exitCommand) {
                    // Insert pre-action code here
                    exitMIDlet();
                    // Insert post-action code here
                }
            }
            // Insert global post-action code here
        }

        /**
         * This method should return an instance of the display.
         */
        public Display getDisplay() {
            return Display.getDisplay(this);
        }

        /**
         * This method should exit the midlet.
         */
        public void exitMIDlet() {
            getDisplay().setCurrent(null);
            destroyApp(true);
            notifyDestroyed();
        }

        /** This method returns instance for helloForm component and should be called instead of
         accessing helloForm field directly.
         * @return Instance for helloForm component
         */
    }
}
```

2

6

```

public Form get_helloForm() {
    if (helloForm == null) {
        // Insert pre-init code here
        helloForm = new Form(null, new Item[] {get_helloStringItem()});
        helloForm.addCommand(get_exitCommand());
        helloForm.setCommandListener(this);
        // Insert post-init code here
    }
    return helloForm;
}

/** This method returns instance for helloStringItem component and should be called instead of
accessing helloStringItem field directly.
 * @return Instance for helloStringItem component
 */
public StringItem get_helloStringItem() {
    if (helloStringItem == null) {
        // Insert pre-init code here
        helloStringItem = new StringItem("LblBonjour\n", "Bonjour d'\Armenti\u00E8res");

        // Insert post-init code here
    }
    return helloStringItem;
}

/** This method returns instance for exitCommand component and should be called instead of
accessing exitCommand field directly.
 * @return Instance for exitCommand component
 */
public Command get_exitCommand() {
    if (exitCommand == null) {
        // Insert pre-init code here
        exitCommand = new Command("Exit", Command.EXIT, 1);
        // Insert post-init code here
    }
    return exitCommand;
}

public void startApp() {
    initialize();
}

public void pauseApp() {
}

public void destroyApp(boolean unconditional) {
}

```

3.4. Désactivation du mode « Lazy Initialized »

Par défaut, la génération de code de NetBeans utilise le mode « lazy ». Nous allons le désactiver pour identifier les différences de codage.

Remarque : Désactiver le mode « Lazy Initialized » est souvent utile pour pouvoir insérer des modifications dans le codage.

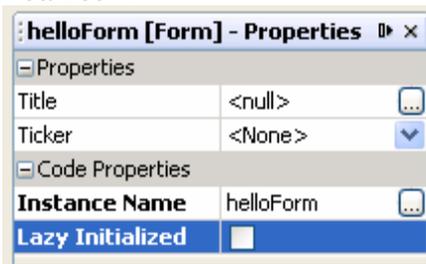
Créez un nouveau projet identique au précédent : Choisissez File > New Project (Ctrl-Shift-N).

- Dans la liste box, choisissez Mobile et Mobile Application, puis Next
- Entrez le nom du projet pjMidletBonjourNonLazy et choisissez votre répertoire de sauvegarde.
- Validez les deux coches « Set as Main Project » et « Create Hello Midlet »
- Gardez les sélections d'émulateur par défaut ainsi que la Configuration CLDC 1.1 et le Profile MIDP 2.0.
- Choisissez Finish pour clore la création du projet.

Nous allons modifier la génération de code source pour la Form du MIDlet.

- Ouvrez le Screen Design

- Cliquez sur l'écran helloForm[Form] et, dans la fenêtre des propriétés, enlevez la coche « Lazy Initialized ».



- Modifiez la propriété Text du StringItem en « Bonjour d'Armentières »
- Modifiez la propriété Label du StringItem en « LblBonjour »
- Enlevez la coche « Lazy Initialized » du StringItem



Vérifiez le code source généré et comparez le avec l'exemple précédent en mode « Lazy Initialized »

```

/*
 * HelloMidlet.java
 *
 * Created on 22 novembre 2006, 13:44
 */

package hello;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author frederic
 */
public class HelloMidlet extends MIDlet implements CommandListener {

    /** Creates a new instance of HelloMidlet */
    public HelloMidlet() {
    }

    private Form FormHello;
    private StringItem helloStringItem;
    private Command exitCommand;

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        helloStringItem = new StringItem("LblHello", "Bonjour d'Armentieres");
        FormHello = new Form("Titre de la forme", new Item[] {helloStringItem});
        FormHello.addCommand(get_exitCommand());
        FormHello.setCommandListener(this);
        getDisplay().setCurrent(FormHello);
        // Insert post-init code here
    }
}

```

2

```

    /** Called by the system to indicate that a command has been invoked on a particular
    displayable.
    * @param command the Command that ws invoked
    * @param displayable the Displayable on which the command was invoked
    */
    public void commandAction(Command command, Displayable displayable) {
        // Insert global pre-action code here
        if (displayable == FormHello) {
            if (command == exitCommand) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
        // Insert global post-action code here
    }
}

/**
 * This method should return an instance of the display.
 */
public Display getDisplay() {
    return Display.getDisplay(this);
}

/**
 * This method should exit the midlet.
 */
public void exitMIDlet() {
    getDisplay().setCurrent(null);
    destroyApp(true);
    notifyDestroyed();
}

/** This method returns instance for exitCommand component and should be called instead
of accessing exitCommand field directly.
 * @return Instance for exitCommand component
 */
public Command get_exitCommand() {
    if (exitCommand == null) {
        // Insert pre-init code here
        exitCommand = new Command("Exit", Command.EXIT, 1);
        // Insert post-init code here
    }
    return exitCommand;
}

public void startApp() {
    initialize();
}

public void pauseApp() {
}

public void destroyApp(boolean unconditional) {
}
}

```

Compilez et exécutez le programme

- Choisissez Run > Run Main Project (F6).

Agrandissez la fenêtre Output pour afficher correctement les messages de construction de votre projet. Un émulateur est lancé pour afficher le résultat de votre application. Le simulateur est « DefaultColorPhone ». Dans la fenêtre de l'émulateur, vous pouvez exécuter la commande Launch et Exit de la même manière que sur votre portable.

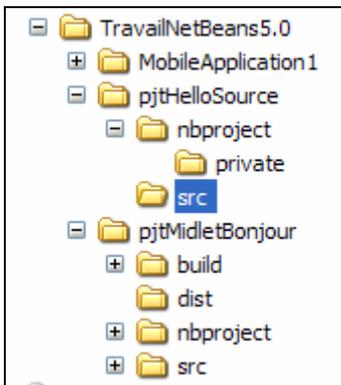
4. Création d'une application MIDP en utilisant l'éditeur de code source.

Cette procédure vous permet de programmer une application suivant vos propres exigences ou lorsque l'interface graphique ne vous permet pas d'aborder des problèmes particuliers (Canvas par exemple).

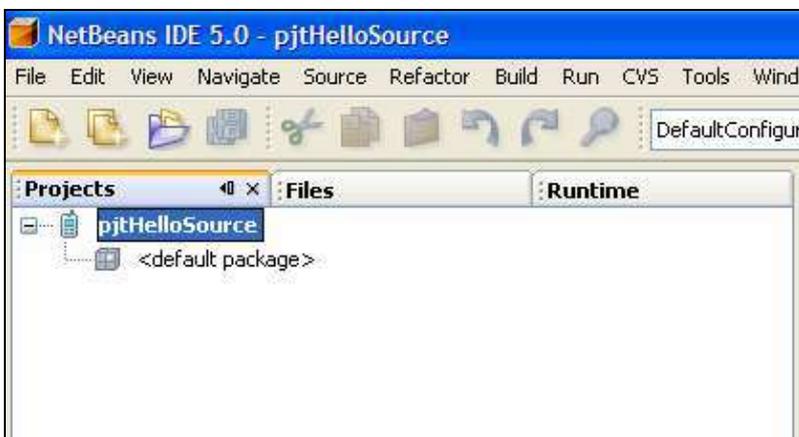
4.1. Création du projet

- Choisissez File > New Project (Ctrl-Shift-N)
- Choisissez la catégorie « Mobile » puis « Mobile Application ».
- Entrez le nom du projet `pjtHelloSource` et précisant son répertoire.
- Validez la coche « Set as Main Project » mais enlevez celle de « Create Hello MIDlet »
- Validez les paramètres par défaut du Wireless Toolkit
- Cliquez sur Finish et votre projet est créé.

Voici l'arborescence créée :



Voici une capture d'écran du projet dans NetBeans



4.2. Création du MIDlet

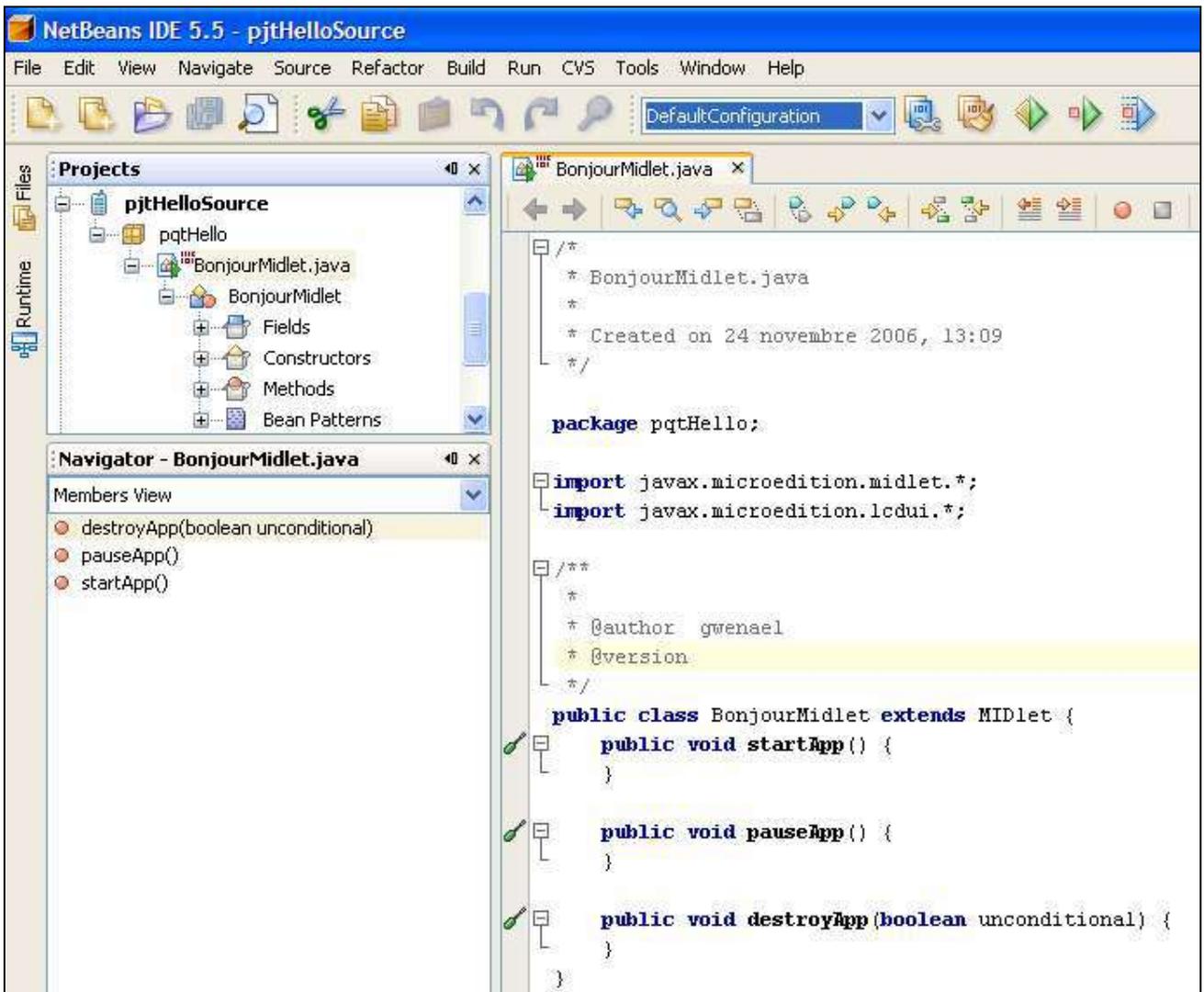
Créez un nouveau paquetage :

- Clic droit sur le nœud `pjtHelloSource` > New > Java Package
- Donnez-lui pour nom : `pqtHello`

Création du MIDlet :

- Clic droit sur le paquetage « `pqtHello` » > New > File/Folder
- Dans la catégorie MIDP, choisissez MIDlet, puis Next
- Entrez le nom du MIDlet : `BonjourMIDlet` et vous avez terminé : la classe `BonjourMIDlet` est créée

Double cliquez sur `BonjourMIDlet.java` et vous affichez le code source dans l'éditeur.



Modifiez le code en insérant les caractères gras du listing suivant.

```

/*
 * BonjourMidlet.java
 *
 * Created on 24 novembre 2006, 13:09
 */

package pqHello;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author gwenael
 * @version
 */
public class BonjourMidlet extends MIDlet implements CommandListener {
    private Display objDisplay;
    private Form helloForm;
    private StringItem helloStringItem;
    private Command exitCommand;
}

```

```

public BonjourMidlet() {
    objDisplay = Display.getDisplay(this);
    helloStringItem = new StringItem("Bienvenue", "Bonjour d'Armentières");
    exitCommand = new Command("Quitter", Command.EXIT, 1);
    helloForm = new Form(null, new Item[] {helloStringItem});
    helloForm.addCommand(exitCommand);
    helloForm.setCommandListener(this);
}

public void commandAction(Command command, Displayable displayable) {
    if (command == exitCommand) {
        objDisplay.setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }
}

public void startApp() {
    objDisplay.setCurrent(helloForm);
}

public void pauseApp() {
}

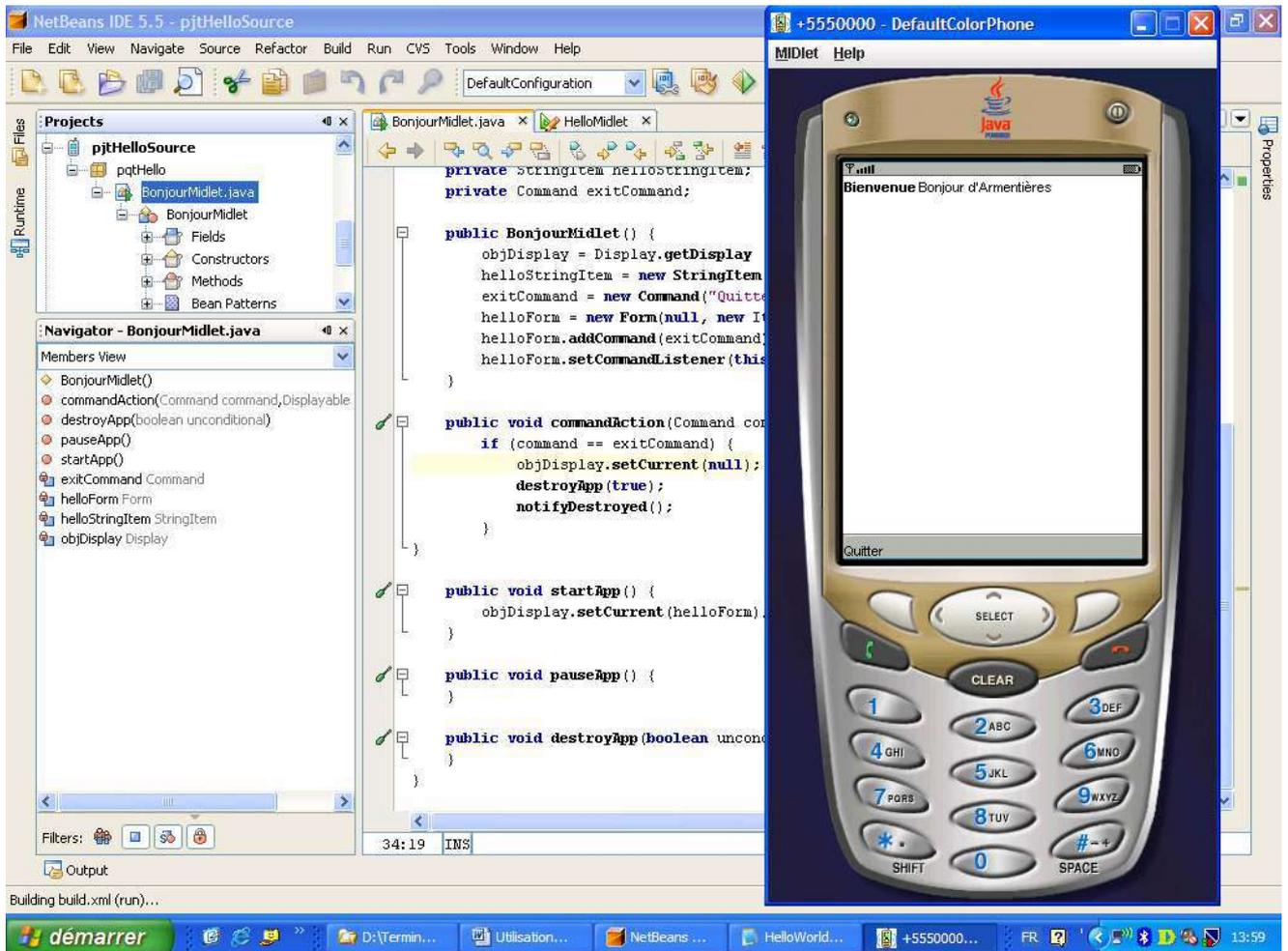
public void destroyApp(boolean unconditional) {
}
    
```

1

3

2

Lancez l'application, vous devez obtenir :



5. Affichage de l'aide : javadocs

A tout moment dans votre éditeur source, vous pouvez obtenir de l'aide en appuyant sur ALT F1, le curseur positionné sur le mot Java posant des problèmes, une page JavaDocs s'ouvrira en vous donnant les informations désirées.

6. Utilisation des tutoriaux

Dans le menu Help, choisissez Welcome Screen et vous aurez accès à différents Getting Started (liaison Internet nécessaire) On peut par ce menu ajouter de nouvelles fonctionnalité telles que le C++.
L'écran Welcome Screen vous propose également des exemples de projets (généraux, Web, Mobile, Entreprise, Web Service, ...)

Dans le menu Help, vous avez également accès aux JavaDoc References (Java EE 5 SDK et JUnit API)

7. Débogage

Quand vous démarrez une session de débogage dans l'environnement de NetBeans, l'IDE compile les fichiers que vous voulez déboguer, les exécute en mode debug, affiche la fenêtre Debugger Console.

Pour démarrer une session de debug, sélectionnez le fichier désiré et utilisez l'une des commandes suivantes :

- **Debug Main Project (F5)** : exécute les programme jusqu'au premier point d'arrêt rencontré.
- **Step Into (F7)** : idem
- **Run to Cursor (F4)** : exécute le programme jusqu'à la position du curseur.

La fenêtre de Debugger Console ouvre les fenêtres suivantes : Watches , Local Variables et Call Stack.

The screenshot shows the NetBeans IDE interface during a debug session. The main editor displays the source code of the `HelloMidlet` class. A breakpoint is set at line 32, which is highlighted in red. The `initialize()` method is currently executing, and the `commandAction()` method is also visible. The `Debugger Console` window at the bottom left shows the execution log, including the message "Breakpoint hit at line 32 in class hello.HelloMidlet by thread KVM_main." The `Watches` window at the bottom right shows the state of the `helloStringItem` object, with fields like `str` (value: "Bonjour d'Armentières?") and `font` (value: #147501). The `Local Variables` window at the bottom right shows the current state of local variables.

Liste des différentes fenêtres de débogage :

Nom	Raccourcis	Description
Local Variables	Alt-Shift-1	Liste les variables locales de la méthode courante
Watches	Alt-Shift-2	Liste les variables et les expressions que vous désirez visualiser
Call Stack	Alt-Shift-3	Liste la séquence des appels durant l'exécution du thread courant
Classes	Alt-Shift-4	Affiche la hiérarchie des classes chargées pendant le débogage.
Breakpoints	Alt-Shift-5	Liste les points d'arrêt du projet
Session	Alt-Shift-6	Liste les sessions de débogage en cours
Threads	Alt-Shift-7	Liste les threads de la session
Sources	Alt-Shift-8	Liste les répertoires sources de votre projet.

Autres commandes :

- **Step Over (F8)** : Exécute l'instruction en évitant les appels de méthode.
- **Step Into (F7)** : Exécute l'instruction en entrant dans les appels de méthode
- **Step Out (Alt-Shif-F7)** : Exécute une instruction. Si la ligne de code fait partie d'une méthode, les lignes de codes suivantes de la méthode s'exécutent et on retourne dans le code de la méthode appelante.
- **Continue (Ctrl-F5)** : Continue l'exécution de l'application.