3

Développer avec le kit Flex SDK

Un peu à la manière de Java et son SDK (*Software Development Kit*), le kit de développement Flex SDK offre un ensemble de fonctionnalités permettant de réaliser gratuitement des applications à l'aide du framework Flex. Ce kit comprend des compilateurs, un débogueur ainsi que les bibliothèques de composants du framework permettant de créer des applications.

Depuis la version 3 de Flex, la volonté d'Adobe est de rendre le SDK disponible sous licence MPL (*Mozilla Public License*). Selon l'éditeur, le passage à l'Open Source permettra une augmentation de l'utilisation de son produit. En effet, le SDK pourra être intégré au sein de divers IDE (*Integrated Development Environment*) et les entreprises à philosophie Open Source pourront opter pour l'utilisation de Flex. Enfin, comme chacun le sait, les produits libres sont souvent supportés par une communauté active, ce qui permet de les faire évoluer et d'offrir un support très réactif. Ceci assure ainsi à Flex une place prédominante par rapport à ses concurrents, notamment le géant Microsoft et son produit SilverLight, dont le choix stratégique viserait à imiter Adobe.

Licence MPL

Contrairement à la licence GPL (*General Public License*), qui oblige à fournir les sources de l'ensemble d'un programme, la MPL permet de ne pas divulguer certaines parties du programme. Ainsi, l'entreprise éditrice peut ne communiquer que certaines parties du code de son logiciel et garder secrètes les parties qu'elle juge sensibles.

Prérequis

Avant d'aller plus loin, il convient de vérifier la présence des éléments indispensables au bon fonctionnement du SDK.

Du point de vue matériel, vous devez disposer au minimum de 512 Mo de RAM et de 200 Mo d'espace disque.

Pour l'aspect logiciel, il est impératif que Java soit installé et correctement configuré sur votre machine, bien que le Runtime de Java (JRE, *Java Runtime Environment*) suffise pour exécuter Flex SDK.

> **JRE et JDK *(Java Developpement Kit)***
>
> Il peut être intéressant d'installer le Java Developpement Kit (JDK) si vous souhaitez par la suite développer vos propres applications Java. Sur ce point, nous verrons au chapitre 14 que Flex est capable d'interagir avec les applications développées dans ce langage. De plus, vous en aurez besoin pour mener à bien les exemples mentionnés dans ce chapitre. Étant donné que le JRE est contenu dans le JDK, pourquoi s'en priver ?

Dans cet ouvrage, nous utiliserons la version 6 du JDK (Java SE6), qui peut être téléchargée sur le site web de Sun Microsystems à l'adresse suivante : `http://java.sun.com/` (rubrique Downloads>Java SE). Une fois l'archive téléchargée, procédez à son installation en suivant les indications de l'assistant.

Vous pouvez éventuellement configurer la variable d'environnement `PATH` de votre système en y ajoutant le chemin du répertoire `bin` de Java. Ceci vous permettra d'appeler les programmes Java (`java.exe`, `javac.exe`, etc.) à partir de n'importe quel répertoire de votre système.

> **Important**
>
> Les instructions d'installation de logiciel ou de configuration d'environnement présentées dans cet ouvrage sont essentiellement orientées vers le système d'exploitation Windows, car il s'agit de l'environnement le plus utilisé.

Voici la marche à suivre pour modifier la variable `PATH` :

1. Cliquez sur Démarrer et sélectionnez Panneau de configuration>Système.
2. Dans la fenêtre qui apparaît alors à l'écran, cliquez sur l'onglet Avancé puis sur le bouton Variables d'environnement.
3. Dans la zone Variables système, sélectionnez la variable `PATH` et cliquez sur le bouton Modifier.
4. À la suite des valeurs déjà présentes, ajoutez le chemin du répertoire `bin` de Java (par défaut, `C:\Program Files\Java\jdk1.6.0_<version>\bin`) précédé d'un point-virgule.
5. Validez la saisie et fermer l'ensemble des fenêtres précédemment ouvertes.

L'installation du JDK de Java est à présent terminée. Pour vérifier la bonne configuration de votre système, ouvrez l'invite de commandes MS-DOS et saisissez l'instruction suivante :

```
java -version
```

Le résultat obtenu doit être semblable à celui de la figure 3-1.

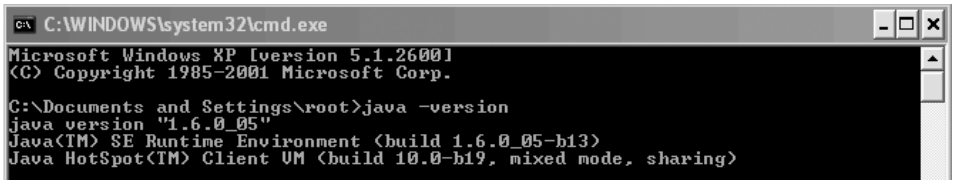

Figure 3-1

Vérification de la bonne configuration de Java

Vous aurez également besoin du Flash Player 9 d'Adobe, grâce auquel vous pourrez exécuter les applications Flex. Son installation s'effectue en ligne à partir du site web d'Adobe : `http://www.adobe.com/shockwave/download/download.cgi?P1_Prod_Version=ShockwaveFlash`.

Vous disposez à présent de tous les outils nécessaires pour utiliser le kit Flex SDK correctement. Passons maintenant à son installation.

Mise en garde quant au choix du Flash Player

Comme nous l'avons vu au chapitre précédent, la version actuelle du Flash Player est la version 10. Cependant, il se peut que vous possédiez encore la version 9. Mettre à jour le Flash Player alors que vous avez déjà installé l'environnement de développement peut engendrer des disfonctionnements du Flex Builder, notamment pour la fonctionnalité de débogage. Afin de réaliser l'ensemble des cas pratiques de cet ouvrage dans les meilleurs conditions, nous vous conseillons de choisir l'une ou l'autre version et de ne pas procéder encore à la mise à jour. Si pour une raison ou une autre, votre Flash Player est mis à jour pour la version 10, reportez-vous au chapitre 25, où nous expliquerons comment adapter votre projet à la version 10. Enfin, nous attirons votre attention sur le fait que la version 10 du Flash Player étant récente, nous n'avons pas pu tester dessus l'ensemble des fonctionnalités de Flex 3. Par prudence, nous vous conseillons d'installer la version 9 (`http://www.adobe.com/support/flashplayer/downloads.html`), dans le cas où vous auriez déjà effectué la mise à jour de votre système.

Installation du kit Flex SDK

Pour commencer, téléchargez le kit Flex SDK sur le site d'Adobe à l'adresse suivante : `http://www.adobe.com/products/flex/flexdownloads/`.

Une fois le téléchargement terminé, décompressez l'archive obtenue dans un répertoire de votre choix. Nous avons ici choisi le répertoire `C:\flex_sdk` que nous nommerons

root_flexsdk par convention. Le répertoire root_flexsdk contient désormais les dossiers présentés et décrits dans le tableau 3-1.

Tableau 3-1 Les différents dossiers du répertoire root_flexsdk

Nom du dossier	Description
ant	Contient un ensemble de tâches Ant permettant d'automatiser certaines opérations lors des phases de compilation et de déploiement.
asdoc	Contient l'ensemble de la documentation de l'outil ASDoc permettant de créer la documentation de votre application sous forme de pages HTML.
bin	Contient un ensemble d'exécutables, tels que mxmlc.exe, compc.exe (voir sections suivantes) ou encore asdoc.exe.
frameworks	Contient l'ensemble des classes, des fichiers de configuration et du code source du framework.
lib	Contient l'ensemble des bibliothèques Java du framework.
runtimes	Contient les exécutables permettant d'installer l'environnement Adobe AIR et le module de débogage de Flash Player 10.
samples	Contient des exemples d'applications.
templates	Contient des templates de pages HTML contenant l'application Flex. Ceux-ci contiennent notamment la fonctionnalité de détection du Flash Player. Ainsi, si vous utilisez un template proposant cette fonctionnalité et que l'utilisateur ne possède pas le Flash Player, son téléchargement sera automatiquement proposé.

L'installation est à présent terminée et le Flex SDK est opérationnel. Voyons à présent le cœur du SDK, dont les compilateurs sont les principaux éléments.

Les compilateurs

Comme nous l'avons vu au chapitre précédent, une application Flex est composée de fichiers MXML et ActionScript. Ces fichiers sont ensuite compilés, donnant naissance à un fichier SWF. Flex distingue deux types de compilations qui dépendent chacune d'un compilateur spécifique : la compilation d'application et la compilation de composants.

mxmlc : le compilateur d'application

Le rôle principal du compilateur mxmlc est de créer un fichier de type SWF (*Shockwave Flash*) à partir des fichiers ActionScript et MXML d'une application Flex (figure 3-2).

Ce compilateur s'exécute en ligne de commande à l'aide de l'exécutable mxmlc.exe situé dans le dossier bin du répertoire root_flexsdk.

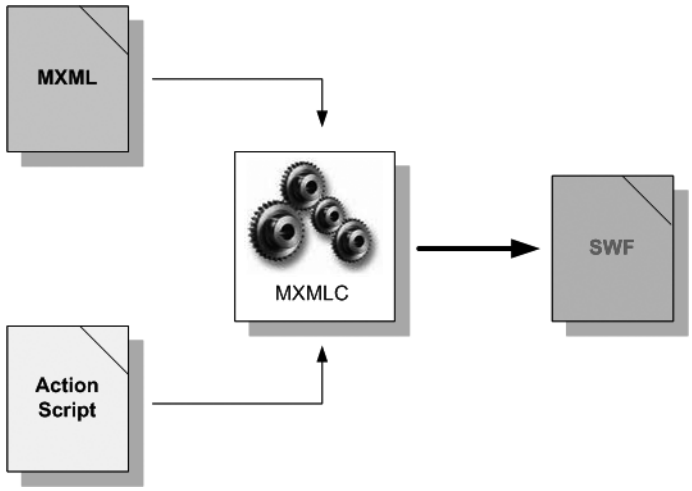

Figure 3-2
Mécanisme de compilation mxmlc

La syntaxe générale d'exécution de la compilation est la suivante :

```
mxmlc options fichier
```

où `mxmlc` fait appel au compilateur `mxmlc.exe`, `options` spécifie les options de compilation (date de création, nom du fichier SWF généré…) et `fichier` indique le fichier MXML à compiler.

Le fichier MXML principal de l'application est spécifié en omettant les fichiers ActionScript associés à celle-ci. Cependant, les fichiers ActionScript liés à l'application ne sont pas oubliés lors de la compilation grâce aux liens spécifiés dans le fichier MXML principal.

Il est également important de noter que ce compilateur ne génère pas la page HTML chargée de contenir le SWF créé. Elle devra être créée manuellement.

compc : le compilateur de composants

Flex offre la possibilité de créer des composants afin de les réutiliser dans plusieurs applications différentes. Un composant peut être assimilé à une application Flex dans la mesure où il est également composé d'un fichier MXML et de fichiers ActionScript.

La compilation de ces composants est réalisée à l'aide du compilateur compc dont la syntaxe est décrite par l'exemple suivant :

```
compc -o composant.swc -root fichierMXML.mxml
```

où l'option `–o` permet de spécifier le nom du composant compilé et l'option `–root` indique le nom du fichier MXML utilisé pour la compilation du composant. Cette simple ligne de code permet de générer un composant (`composant.swc`) à partir d'un fichier MXML.

Le fonctionnement du compilateur compc est identique à celui du compilateur mxmlc à la seule différence qu'il génère un fichier de type SWC (*Shockwave Component*) et non SWF (figure 3-3).

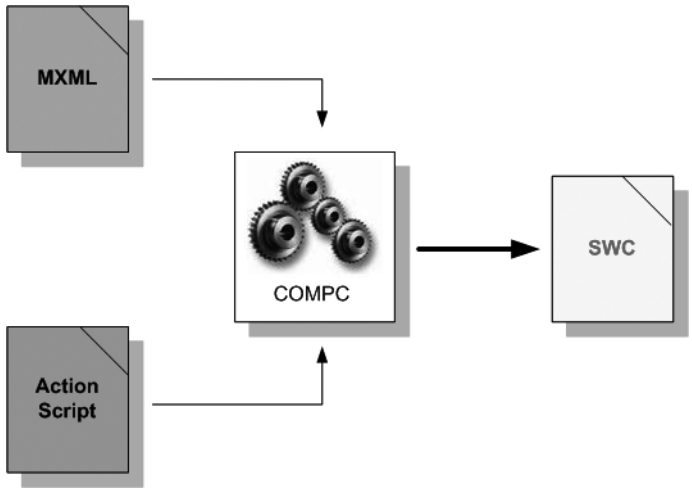

Figure 3-3
Mécanisme de compilation compc

Si nous récapitulons, voici ce que nous avons appris jusqu'à présent :

- Comment configurer un environnement pour exécuter le kit de développement Flex SDK et les applications Flex.
- Comment installer le kit Flex SDK.
- Quelques connaissances de base sur les différents compilateurs.

Cela peut paraître léger, mais nous allons voir qu'avec ces trois éléments seulement, nous sommes en mesure de créer notre première application Flex.

Première application Flex

Nous vous proposons ici de développer le fameux programme cher à tout informaticien et informaticienne, à savoir l'incontournable Hello World ! dont le but est d'afficher un message de bienvenue à l'écran.

Conception graphique

Nombreux sont les projets prenant naissance à partir d'un dessin griffonné sur un coin de table. Notre projet n'a pas échappé à cette règle comme l'illustre la figure 3-4.

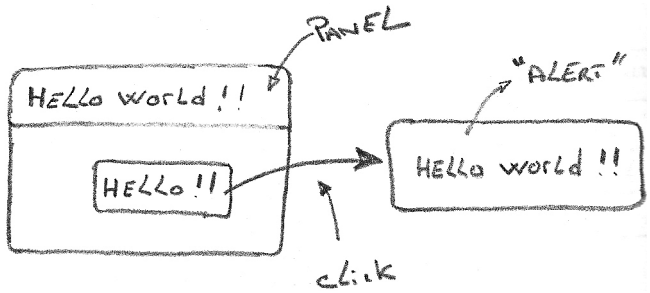

Figure 3-4
Conception graphique

Le principe de cette application est simple : un composant, nommé Panel contient un bouton qui, lorsqu'on clique dessus, affiche le message « Hello World ! ».

Création du projet

Au cours du chapitre précédent, nous avons abordé la notion d'arborescence des projets Flex et nous avons vu que tout projet doit posséder un répertoire racine contenant un sous-répertoire `src` dans lequel seront enregistrées les sources de l'application.

Le kit Flex SDK propose différents exemples d'applications situés dans le dossier `samples`. C'est dans ce dossier que nous allons placer la racine de notre projet, que l'on nommera `HelloWorld`. Dans la mesure où il s'agit du fichier racine de l'application, ce répertoire contiendra un sous-répertoire nommé `src` (figure 3-5).

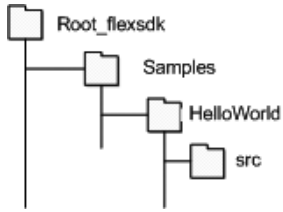

Figure 3-5
Arborescence du projet

Écriture du fichier MXML

Tout projet Flex comporte un fichier portant l'extension `.mxml` servant de base à l'application et permettant, entre autres choses, la description des interfaces graphiques.

Mais il ne suffit pas de créer des composants, il faut également définir leurs caractéristiques (nom, hauteur, largeur...) et leur positionnement. Les tableaux 3-2 et 3-3 présentent les différentes caractéristiques des composants `Panel` et `Button` que nous allons créer.

Tableau 3-2 Les différentes caractéristiques du composant Panel

Caractéristique	Description
Rôle	Conteneur principal de l'application
Nom	`pn_principal`
Titre (`title`)	`Hello World !`
Largeur (`width`)	270 pixels
Hauteur (`height`)	150 pixels
Position sur l'axe des X	10 pixels
Position sur l'axe des Y	10 pixels
Disposition (`layout`)	`absolute` (permet le positionnement des composants à l'aide de leurs coordonnées x et y).

Tableau 3-3 Les différentes caractéristiques du composant Button

Caractéristique	Description
Rôle	Bouton déclenchant l'affichage du message.
Nom	`btn_message`
Label	`Hello !!`
Largeur (`width`)	120 pixels
`horizontalCenter`	0 (centre le bouton à l'intérieur du panel sur l'axe horizontal de celui-ci).
`verticalCenter`	0 (centre le bouton à l'intérieur du panel sur l'axe vertical de celui-ci).

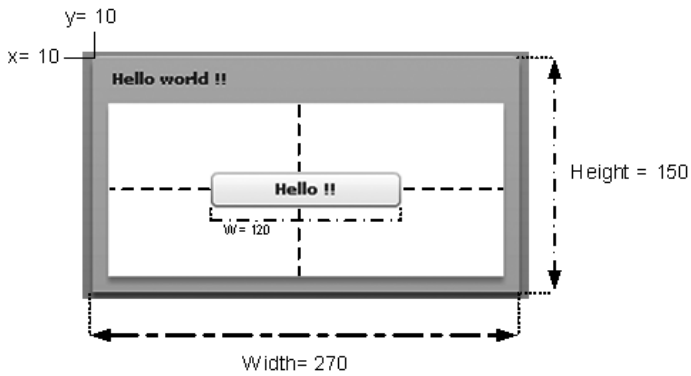

Figure 3-6
Mise en place des composants

Nous allons à présent nous placer dans le répertoire `src` et créer un nouveau fichier nommé `HelloWorld.mxml`, dans lequel nous saisirons le code suivant à l'aide du Bloc-notes de Windows :

```
<?xml version="1.0" encoding="utf-8"?>

<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">

<!-- Ajout du panneau -->
<mx:Panel id="panneau" x="10" y="10" width="270" height="150" layout="absolute"
➥title="Hello World !!">
<!-- Ajout du bouton -->
<mx:Button id="bouton_action" label="Hello !!" width="120" horizontalCenter="0"
➥verticalCenter="0"/>
</mx:Panel>

</mx:Application>
```

Les caractéristiques décrites aux tableaux 3-2 et 3-3 sont bien spécifiées pour chacun des composants. Notre interface graphique est à présent terminée. Passons maintenant à la partie ActionScript qui permettra d'afficher le message « Hello World !! ».

Écriture du fichier ActionScript

Tout d'abord, créez un nouveau fichier nommé `HelloActionScript.as` dans le répertoire `src` de l'application qui va contenir la procédure permettant d'afficher le message. Saisissez ensuite le code suivant dans ce fichier :

```
import mx.controls.Alert;

public function afficherMessage():void
{
Alert.show("Hello World !!");
}
```

Nous importons tout d'abord la bibliothèque nécessaire à l'utilisation de la classe `Alert`. Vient ensuite la création de la procédure `afficherMessage()` de type `void` étant donné qu'aucune valeur de retour n'est attendue. Cette procédure fait appel à la méthode `show` de la classe `Alert` dont l'analyse syntaxique du code suffit à sa compréhension.

Liaison graphique et action

Nous possédons à présent un fichier MXML servant à la description de l'interface graphique et un fichier ActionScript permettant d'afficher le message « Hello World !! ».

L'étape suivante va consister à créer la liaison entre ces deux fichiers et à définir l'action qui fera appel à la fonction `afficherMessage()`. Cette liaison s'effectue dans le fichier MXML à l'aide de la balise `<mx:Script>` pour laquelle il suffit de spécifier la source du script à utiliser, c'est-à-dire le chemin relatif du fichier ActionScript :

```
<mx:Script source="HelloActionScript.as"></mx:Script>
```

Lors de la phase de conception graphique, nous avons décidé que le message doit s'afficher lorsque l'utilisateur clique sur l'unique bouton de l'interface. Cela se fera via la définition de l'action sur `click` du bouton faisant appel à la fonction `afficherMessage()`.

```
<mx:Button id="bouton_action" label="Hello !!" width="120" horizontalCenter="0"
➥verticalCenter="0"
click="afficherMessage()"/>
```

Pour plus de clarté, voici le script complet du fichier `HelloWorld.mxml` :

```
<?xml version="1.0" encoding="utf-8"?>

<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">

<mx:Script source="HelloActionScript.as"></mx:Script>

<mx:Panel id="panneau" x="10" y="10" width="270" height="150" layout="absolute"
➥title="Hello World !!">
<mx:Button id="bouton_action" label="Hello !!" width="120" horizontalCenter="0"
➥verticalCenter="0" click="afficherMessage()"/>
</mx:Panel>

</mx:Application>
```

Compilation

Il nous faut à présent compiler notre application, ce qui aura pour effet de créer un fichier SWF pouvant être inséré dans une page HTML et diffusé sur Internet.

Pour cela, ouvrez, l'invite de commandes MSDOS et placez-vous dans le répertoire `root_flexsdk\bin` avec la commande `cd root_flexsdk\bin`. Étant donné qu'il s'agit d'une application, il convient d'utiliser le compilateur mxmlc. Voici donc la commande à saisir pour générer le fichier SWF de notre application :

```
mxmlc ../samples/HelloWorld/src/Helloworld.mxml
```

Placez-vous ensuite dans le répertoire `src` du projet, vous devriez à présent y trouver le fichier `HelloWorld.swf` issu de la compilation.

Que s'est-il passé ? Le compilateur mxmlc a analysé le fichier `.mxml` de l'application et a constaté qu'il possédait un lien vers un fichier ActionScript. Il a donc pris en compte ce fichier et a transformé le fichier MXML en un ensemble de classes ActionScript pour ensuite les intégrer dans un fichier de type SWF.

À ce stade, nous pouvons déjà tester notre application à l'aide du Flash Player. En cliquant sur le bouton, vous devriez en effet voir apparaître le message « Hello World !! » (figures 3-7 et 3-8).

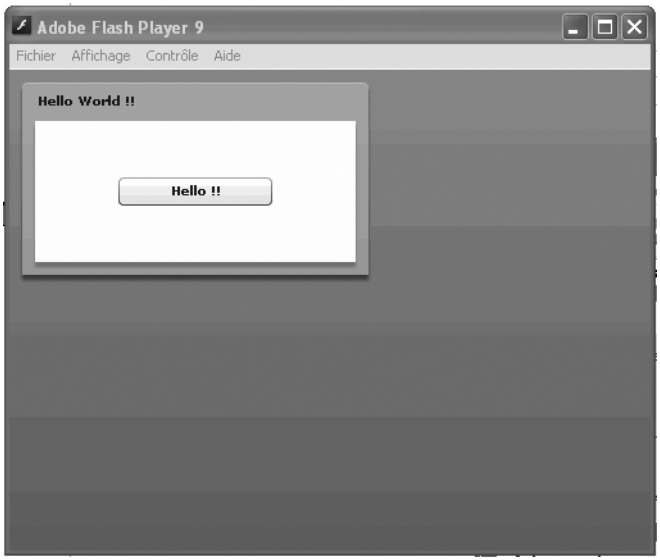

Figure 3-7

Exécution de l'application dans le Flash Player 9

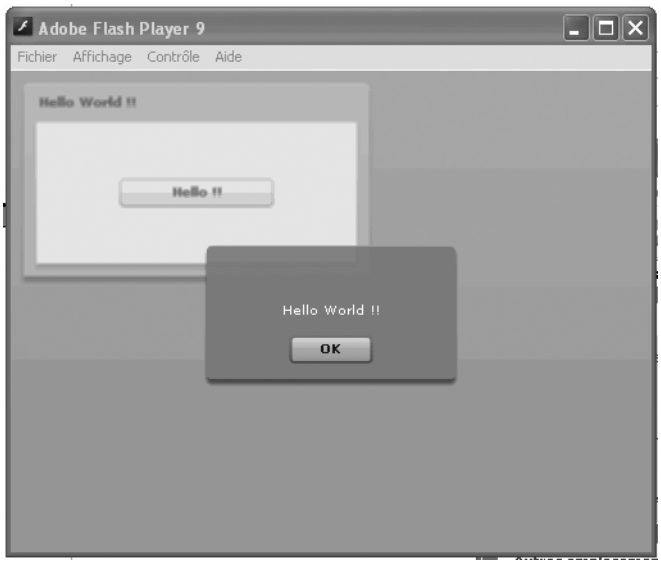

Figure 3-8

Affichage du message « Hello World !! »

Nous disposons maintenant d'un fichier SWF complet et autonome. Nous entendons par là que ce fichier n'a pas besoin des fichiers sources (MXML et ActionScript) pour son exécution car ceux-ci ont été compilés et intégrés au fichier SWF. L'étape suivante va consister à intégrer ce fichier dans une page web afin de diffuser notre application sur Internet.

Intégration dans une page web

Pour intégrer notre application dans une page web, que nous nommerons `index.html`, nous allons utiliser le langage HTML et ses balises `<object>` qui nous permettront de faire référence au fichier SWF.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/1998/
➥REC-html40-19980424/strict.dtd">

<html>
<head>
    <title>Hello World !</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>

<body>
    <p>
        <object type="application/x-shockwave-flash" data="HelloWorld.swf"
        ➥width="100%" height="100%">
        <param name="movie" value="HelloWorld.swf">
        </object>
    </p>
</body>

</html>
```

Cette insertion est des plus simples. Notez l'utilisation des propriétés `width` et `height` permettant de spécifier la taille de l'objet SWF par rapport à celle de la page HTML. Dans notre cas, l'application prendra la taille de la page HTML affichée dans le navigateur (figure 3-9).

Nous verrons dans la section « Création d'un script de compilation » qu'il est possible de définir plusieurs options pour l'inclusion du fichier SWF, notamment la détection de la présence du Flash Player sur le poste client, nécessaire à l'exécution de l'application (cette fonctionnalité n'est pas implémentée ici).

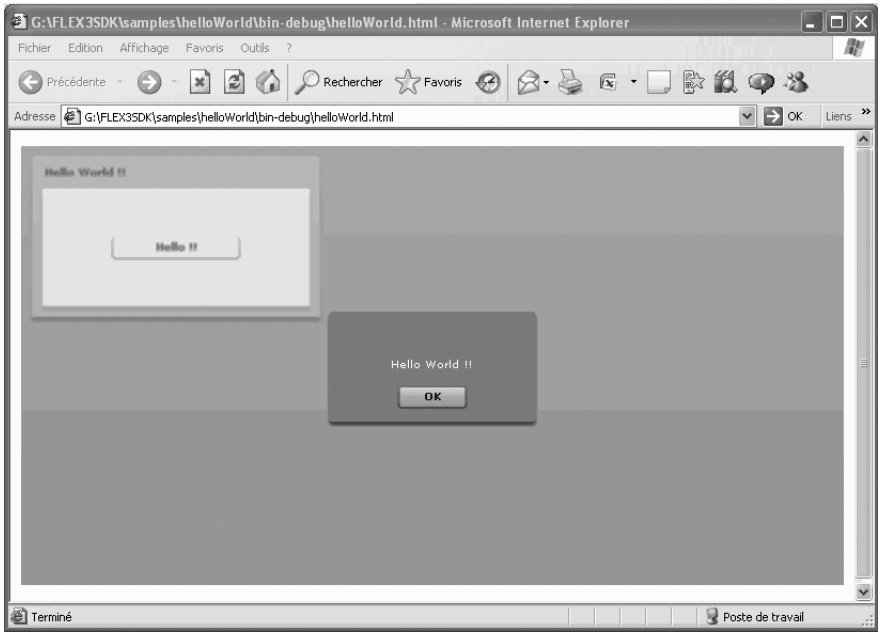

Figure 3-9
Application insérée dans une page HTML

Déploiement

Si vous possédez un serveur web, il vous suffit de placer les fichiers `HelloWorld.swf` et `index.html` dans l'un de ses répertoires, cela terminant par ailleurs la phase de déploiement.

Libre à vous ensuite de communiquer l'URL de votre application à l'ensemble de vos contacts. Dans la mesure où celle-ci ne permet pas de détecter la présence du Flash Player sur le poste client, n'oubliez pas de mentionner qu'il est nécessaire et doit être installé pour pouvoir exécuter votre application.

Automatisation de la compilation à l'aide de Ant

Nous verrons au chapitre suivant que l'utilisation de Flex Builder facilite grandement le déploiement d'une application Flex. Néanmoins, en fonction de votre budget, vous préférerez peut-être le kit Flex SDK. Vous devrez alors procéder à une compilation manuelle, mais cette tâche est particulièrement fastidieuse (notamment lors des phases de test).

Nous vous proposons donc de voir comment automatiser les tâches de compilation à l'aide de Ant, outil développé par la *Apache Software Foundation* et communément utilisé dans le déploiement des applications Java.

Ant

Projet de la *Apache Software Foundation*, Ant, signifie *Another Neat Tool*. Il est écrit en Java et a été développé par James Duncan Davidson en 2000. Ant permet d'automatiser la compilation et le déploiement des projets. Il repose sur l'exécution de scripts de « build » au format XML comportant un projet et un ensemble de tâches, organisées ou non, qui y sont liées.

Ant et Flex

Lors de l'analyse du répertoire principal de Flex, nous avons noté la présence d'un dossier nommé `ant`. Ce répertoire contient la bibliothèque nécessaire à `Ant` pour pouvoir compiler et déployer des projets Flex. Cette bibliothèque est nommée `flexTasks.jar` et se trouve dans le répertoire `lib` du dossier `ant`. Elle contient trois tâches Ant qui permettent d'utiliser :

- le compilateur mxmlc ;
- le compilateur compc ;
- la tâche `html-wrapper` permettant de générer la page HTML intégrant le fichier `.swf` créé lors de la compilation.

Voyons comment utiliser le couple Ant/flexTasks pour automatiser la compilation et le déploiement d'une application Flex.

Installation de Ant

Commencez par télécharger Ant à partir de l'adresse suivante : `http://ant.apache.org`. Décompressez ensuite l'archive obtenue dans le répertoire de votre choix et configurez votre système d'exploitation comme suit :

1. Créez une variable d'environnement nommée `ANT_HOME` pointant sur le répertoire `ant` (par exemple, `c:\ant`).
2. Modifiez la variable `PATH` en ajoutant le chemin du répertoire `bin` comme suit :

```
%ANT_HOME%\bin
```

3. Ouvrez ensuite l'invite de commandes MS-DOS et tapez la commande `ant`. Si l'environnement est correctement configuré, vous devriez voir apparaître ceci :

```
Buildfile: build.mxml does not exist !
BuildFailed
```

Passons maintenant aux choses sérieuses !

Création d'un script de compilation

Avant tout, récapitulons : nous disposons d'un fichier MXML nommé `HelloWorld.mxl` situé dans le répertoire `root_flexsdk\samples\HelloWorld\src`, ainsi que d'un fichier ActionScript

nommé HelloActionScript.as. Notre objectif est de compiler et d'intégrer automatiquement le fichier SWF issu de la compilation dans une page HTML. Voyons comment procéder.

Nous avons choisi de placer les scripts de compilation Ant dans le répertoire root_flexsdk\ant. Comme nous l'avons vu, un script de compilation est un fichier XML contenant une suite de tâches. La première étape va donc consister à créer ce fichier XML, que l'on nommera HelloWorld.xml. Nous devrons ensuite spécifier le projet Ant en lui affectant un nom.

```
<?xml version="1.0" encoding="utf-8"?>

<project name="Compilateur HelloWorld">

</project>
```

Une fois cela fait, nous devrons préciser le chemin du fichier flexTasks.jar. Si vous avez choisi de placer le fichier de compilation dans le répertoire ant du Flex SDK, le fichier attendu se situe dans le répertoire suivant root_flexsdk\ant\lib :

```
<taskdef resource="flexTasks.tasks" classpath="c:/root_flexsdk/ant/lib/flexTasks.jar" />
```

Vous devrez ensuite paramétrer les variables qui seront utilisées lors de la compilation :

- FLEX_HOME : répertoire d'installation du kit Flex SDK (root_flexsdk) ;
- APPLICATION_HOME : répertoire du fichier à compiler (root_flexsdk/samples/HelloWorld/src).

```
<property name="FLEX_HOME" value="c:/root_flexsdk"/>
<property name="APPLICATION_HOME" value="c:/root_flexsdk/samples/HelloWorld/src"/>
```

Vient ensuite la spécification de la tâche à réaliser. Une tâche (<target>) est définie par un nom qui sera invoqué lors de l'utilisation de Ant.

```
<target name="compilation">

</target>
```

Chaque tâche doit comporter un certain nombre d'instructions. La première que nous allons écrire est l'action de création du fichier SWF. Comme nous l'avons vu précédemment, pour compiler un fichier ActionScript ou MXML relatif à une application, nous devons utiliser le compilateur mxmlc :

```
<echo message=">> Génération du fichier SWF" />

<mxmlc file="${APPLICATION_HOME}/HelloWorld.mxml"></mxmlc>
```

Ce script a pour résultat d'afficher un message « Génération du fichier SWF » et de réaliser la compilation de l'application en prenant les fichiers MXML et ActionScript situés dans le répertoire source spécifié dans la variable ${APPLICATION_HOME}.

Avant d'aller plus loin, testons notre script. Vérifiez d'abord que son contenu est indique à cela :

```
<?xml version="1.0" encoding="utf-8"?>

<project name="My App Builder">
<taskdef resource="flexTasks.tasks" classpath="c:/root_flexsdk/ant/lib/flexTasks
➥.jar" />
    <property name="FLEX_HOME" value="c:/root_flexsdk"/>
    <property name="APPLICATION_HOME" value="c:/root_flexsdk/samples/HelloWorld/src"/>

    <target name="compilation">
        <echo message=">> Génération du fichier SWF" />

<mxmlc file="${APPLICATION_HOME}/HelloWorld.mxml"></mxmlc>
    </target>

</project>
```

Ensuite, ouvrez l'invite de commandes MS-DOS et placez-vous dans le répertoire contenant le fichier de build que nous venons de créer :

```
cd c:\root_flexsdk\ant
```

Pour exécuter le script, saisissez la commande suivante, faisant appel à la tâche de compilation :

```
ant -buildfile HelloWorld.xml compilation
```

Si tout s'est correctement déroulé lors de la compilation, le fichier SWF est créé dans le répertoire de l'application Flex, `root_flexsdk/samples/HelloWorld/src`.

Passons maintenant à l'instruction qui permet de générer le fichier HTML contenant le fichier SWF. Cette action est possible grâce à la tâche `html-wrapper` qui fournit plusieurs templates de génération. Le tableau 3-4 présente les différents arguments de la tâche `html-wrapper`.

Tableau 3-4 Les arguments de la tâche html-wrapper

Arguments	Description
`application`	Nom de l'application SWF.
`bgcolor`	Couleur d'arrière-plan de la page HTML (blanc par défaut).
`height`	Hauteur de l'application SWF (400 pixels par défaut).
`width`	Largeur de l'application SWF (400 pixels par défaut).
`output`	Chemin de génération de la page HTML.
`swf`	Nom du fichier SWF à inclure (ne pas spécifier l'extension `.swf` du fichier).

Tableau 3-4 Les arguments de la tâche html-wrapper *(suite)*

Arguments	Description
`template`	Trois types de templates sont disponibles : • `client-side-detection` : détecte la version du Flash Player installée sur le poste client et propose son téléchargement le cas échéant. • `express-installation` : détecte si Flash Player est présent et propose son installation en mode Express s'il n'est pas installé. • `no-player-detection` : ne vérifie pas la présence du Flash Player.
`history`	Permet d'utiliser la fonction de *DeepLinking*, c'est-à-dire la navigation à l'aide des boutons Précédent et Suivant du navigateur (`false` par défaut).
`title`	Titre de la page HTML (`Flex application` par défaut).
`version-major`	Indique la version majeure du Flash Player à utiliser (version 9 par défaut). Cet argument n'a de sens que si le type de template spécifié est `client-side-detection` ou `express-installation`.
`version-minor`	Indique la version mineure du Flash Player à utiliser (version 0 par défaut). Cet argument n'a de sens que si le type de template spécifié est `client-side-detection` ou `express-installation`.
`version-revision`	Indique la version de mise à jour du Flash Player à utiliser (version 0 par défaut). Cet argument n'a de sens que si le type de template spécifié est `client-side-detection` ou `express-installation`.

Ajoutons donc une nouvelle action de génération de fichier HTML :

```
<target name="generationHTML" depends="compilation">

<echo message=">> Génération du fichier HTML" />

<html-wrapper
    swf = "HelloWorld"
    height = "100%"
    width = "100%"
    application = "HelloWorld"
    template = "express-installation"
    version-major = "9"
    version-minor = "0"
    version-revision = "0"
    output="${APPLICATION_HOME}"
    title = "Hello World !!"/>

</target>
```

Avant d'exécuter ce script, attardons-nous sur la notion de dépendance entre les différentes tâches dans un fichier de compilation de Ant. En analysant l'exemple de code précédent, nous voyons que le terme `depends` a été ajouté à la tâche `generationHTML`. Cela signifie qu'elle ne pourra être exécutée que si la tâche `compilation` l'a été au préalable.

Ajoutons également la propriété `default` au projet, qui permet de spécifier la première tâche à réaliser lors de la compilation.

```
<project name="My App Builder" default="generationHTML">
```

On définit la première tâche comme étant celle réalisant la génération du fichier HTML. Ne pouvant s'exécuter avant la tâche de compilation, Ant va d'abord compiler le projet puis terminer son travail par la génération de la page web. Lors de l'exécution du script Ant, il ne nous sera plus nécessaire de spécifier la tâche à exécuter étant donné que celle-ci est maintenant précisée. La nouvelle commande de compilation est donc :

```
ant -buildfile HelloWorld.xml
```

Afin que tout ceci soit plus clair, voici le script de compilation dans sa totalité :

```
<?xml version="1.0" encoding="utf-8"?>

<project name="My App Builder" default="generationHTML">
<taskdef resource="flexTasks.tasks" classpath="c:/root_flexsdk/ant/lib/flexTasks.jar" />

<property name="APPLICATION_HOME" value="c:/root_flexsdk/samples/HelloWorld/src"/>

    <target name="compilation">
        <echo message=">> Génération du fichier SWF" />
        <mxmlc file="${APPLICATION_HOME}/HelloWorld.mxml"></mxmlc>
    </target>

    <target name="generationHTML" depends="compilation">
        <echo message=">> Génération du fichier HTML" />
        <html-wrapper
            swf = "HelloWorld"
            height = "100%"
            width = "100%"
            application = "HelloWorld"
            template = "express-installation"
            version-major = "9"
            version-minor = "0"
            version-revision = "0"
            output="${APPLICATION_HOME}"
            title = "Hello World !!"/>
    </target>

</project>
```

Procédons à la compilation et rendons-nous ensuite dans le répertoire `src` de l'application. Nous notons alors la création des fichiers suivants :

- `AC_OETags.js` : il s'agit du fichier JavaScript vérifiant la version du Flash Player installée en fonction des paramètres `version-major`, `version-minor` et `version-revision` spécifiés dans la tâche `html-wrapper`.

- `playerProductInstall.swf` : application permettant l'installation du Flash Player requis si le client ne le possède pas.
- `index.html` : page HTML contenant le fichier SWF généré lors de la compilation.

Nous pourrions nous arrêter là, mais nous avons décidé de pousser l'automatisation jusqu'au bout en évitant la saisie de la commande `ant` dans la console MS-DOS. Pour ce faire, nous vous proposons de créer un script `bat` qui s'exécutera en double-cliquant dessus.

Nous vous invitons donc à créer un fichier nommé `compilation.bat` dans le répertoire `src` de l'application et contenant les paramètres suivants (à adapter selon votre configuration) :

```
@ECHO OFF
cls

echo ***************************************
echo .::::::      COMPILATION ANT      ::::::.
echo ***************************************

ant -buildfile c:\root_flexsdk\ant\HelloWorld.xml
```

Double-cliquez ensuite dessus pour l'exécuter. Grâce à ce script exécutant la tâche `ant` spécifiée auparavant, nous automatisons les tâches de déploiement et diminuons ainsi le nombre d'opérations à effectuer dans la console MS-DOS. Ceci réduit le temps requis pour le déploiement et les tests d'exécution de l'application.

En résumé

Ce chapitre vous a permis de créer votre première application Flex. Certes, elle est très simple, mais elle nous a néanmoins permis de découvrir les principaux aspects de la réalisation d'une application. Nous avons également vu les différentes phases de développement d'une application :

- la conception de l'interface graphique ;
- la création du fichier MXML dérivé de la conception ;
- le développement des traitements ActionScript ;
- la mise en relation entre les actions et l'interface graphique ;
- la compilation ;
- le déploiement.

De toutes ces phases, la plus compliquée est sans doute celle visant à concevoir l'interface graphique, car elle demande une grande part d'abstraction, notamment pour disposer les composants à l'aide de leurs coordonnées. Le rêve serait qu'un un outil WYSIWG regroupant toutes ces phases existe ! Eh bien ce rêve est devenu réalité grâce à Flex Builder, que nous vous invitons à découvrir grâce au chapitre suivant.