ÉTUDES, DÉVELOPPEMENT & INTÉGRATION

FLEX 4.5 POUR MOBILES

Développer des applications multiplateformes pour iOS, Android et Tablet OS

Christophe Keromen

Préface Michaël Chaize

DUNOD

Table des matières

Préface III

Avant-propos XVII

Première partie – Développer avec Flex, AIR et Flash Builder

Chapitre 1 – La plateforme Flash 3

1.1 De l'animation vectorielle aux applications Internet riches 3

1.1.1 Qu'est-ce qu'une application Flash ? 4

1.1.2 Un déploiement en constante progression 5

1.2 Composants de la plateforme Flash 5

1.2.1 Écosystème Flash 8

1.2.2 Open Screen Project 9

1.3 Développer avec Flex 9

1.3.1 Principaux composants 9

1.3.2 MXML-ActionScript-ByteCode 10

1.3.3 Flash Builder 4.5.x 11

1.3.4 Générer une application AIR 11

1.3.5 Communiquer avec les serveurs 12

Chapitre 2 – Premiers pas avec Flash Builder 13

2.1 Préalable 13

2.1.1 Version US 13

2.1.2 Contexte de travail 13

2.1.3 Environnement de travail 14

2.2 Premier exemple 14

2.2.1 L'application Westerns 14

2.2.2 Ouverture de Flash Builder 15

2.2.3 Le projet créé 18

2.2.4 Perspective Développement 21

2.3 Compléments 23

2.3.1 Choix de créations 23

2.3.2 Onglet « Mobile Settings » 25

2.3.3 Nettoyer le projet 32

Chapitre 3 – Première liste en MXML 33

3.1 Aspect final désiré 33

3.2 Ajout de composants d'interface 34

3.2.1 Titre de la vue 34

3.2.2 Contrôles 34

3.2.3 Les contraintes de positionnement dans Flex 38

3.2.4 Contrôle List 38

3.3 Liaison des contrôles avec des données 43

3.3.1 Données pour la zone de texte 43

3.3.2 Test des modifications 45

3.3.3 Données pour la liste 47

3.4 Refactoring 51

3.4.1 Sur les noms 51

3.4.2 Positionnement d'objets 53

Deuxième partie – Développer pour les mobiles

Chapitre 4 – Déploiement sur les mobiles 57

4.1 Cibles et modes de déploiement 57

4.1.1 Les plateformes cibles 57

4.1.2 Modes de déploiement 58

4.2 Applications natives 60

4.2.1 Places de marché 60

4.2.2 Avantage des applications natives 60

4.2.3 Compléter AIR SDK 61

Chapitre 5 – Le Bouton : ses clics et ses taps 63

5.1 Nouveau projet avec un bouton 63

5.1.1 Projet + bouton 63

5.1.2 Gestion du clic 63

5.1.3 Debug 64

5.1.4 Vérification dans la console 65

5.1.5 Clic = Tap 65

5.2 Refactoring du gestionnaire d'événements 65

5.2.1 Gestionnaire d'événements généré 65

5.2.2 Remaniement pour remplacer par une fonction 66

5.2.3 Externalisation du code dans un fichier .as 67

5.2.4 Finalisation 68

5.3 La même expérience de développement 69

5.3.1 Multi-touch et Gesture 69

5.3.2 Une pression est gérée comme un clic 70

5.3.3 Test des gestuelles 71

5.3.4 Classes spécialisées 71

Chapitre 6 – Bénéfices de Flex pour mobiles 73

6.1 Adapté pour mobile 73

6.1.1 Au niveau du SDK 73

6.1.2 Au niveau de Flash Builder 74

6.1.3 Support de l'interface tactile 74

6.1.4 *Charting* ... 75
6.1.5 *Défilement de listes* ... 76
6.2 Architecture applicative ... 77
6.2.1 *Bref retour aux années quatre-vingt* ... 77
6.2.2 *L'interface liste/détail* ... 78
6.2.3 *Pile de navigation* ... 78
6.2.4 *Le retour du liste/détail* ... 78
6.2.5 *Un rendu de liste adapté* ... 80
6.3 Un monde presque parfait ... 80
6.3.1 *Write once ?* ... 80
6.3.2 *Différences à prendre en compte* ... 81
6.4 Contrôles de type texte ... 81
6.4.1 *Composants supportés* ... 81
6.4.2 *Styles supportés* ... 82
6.4.3 *Saisie prédictive et clavier virtuel* ... 82
6.5 Pour les développeurs Flex ... 83
6.5.1 *Questions existentielles* ... 83
6.5.2 *Maîtriser les nouveautés* ... 83
6.5.3 *Éviter les composants MX* ... 84
6.5.4 *Absence de certaines fonctionnalités de AIR Desktop* ... 84
6.5.5 *Réutilisation de la couche service* ... 85
6.5.6 *Utiliser ActionScript et FXG pour les skins* ... 85
6.5.7 *Absence du support des process natifs* ... 85

Chapitre 7 – Rendus de listes ... 87
7.1 Liste par défaut ... 87
7.1.1 *Les listes dans Flex* ... 87
7.1.2 *Éléments scrollables* ... 87
7.1.3 *Bounce and Pull* ... 88
7.1.4 *Rendu de base des listes* ... 88
7.2 LabelItemRenderer ... 89
7.2.1 *Un rendu simple basé sur un champ texte* ... 89
7.2.2 *Remaniement pour utiliser une requête HTTP* ... 90
7.2.3 *Appel d'une fonction de rendu* ... 92

7.3 IconItemRenderer 95
7.3.1 *Rappel de la composition du rendu* 95
7.3.2 *Utilisation de messageField* 95
7.3.3 *Ajout d'une icône et d'un décorateur* 96
7.3.4 *Fichier Chevron.fxg* 98
7.3.5 *Autres propriétés de IconItemRenderer* 98
7.3.6 *Amélioration de la liste des Westerns* 99
7.3.7 *Feuille de style CSS* 100
7.3.8 *Utiliser l'assistant en mode Design* 101
7.3.9 *Aller plus loin* 101

Troisième partie – Structurer une application mobile

Chapitre 8 – Architecture Navigateur - Vue 105
8.1 Une architecture pour smartphones 105
8.1.1 *Les trois modèles d'applications* 105
8.1.2 *Architecture de contenu* 106
8.2 ViewNavigator 106
8.2.1 *Définition* 106
8.2.2 *Propriétés* 107
8.2.3 *Méthodes de navigation* 108
8.2.4 *Conservation des vues* 109
8.2.5 *Boutons systèmes* 113
8.3 View 113
8.3.1 *Définition* 113
8.3.2 *Propriétés* 114
8.3.3 *Événements écoutables sur la vue* 114
8.3.4 *État, contexte et retour de données* 116
8.4 Exemples de Passage de données 118
8.4.1 *Contexte* 118
8.4.2 *Objet retour* 119
8.5 ActionBar 121
8.5.1 *Structure* 121
8.5.2 *Propriétés* 121

8.5.3 *Application à Westerns* 122
8.5.4 *Apparence* 125

Chapitre 9 – Westerns en détail 127

9.1 Détail d'un Western 127
9.1.1 *Préparation* 127
9.1.2 *Création de la vue détail* 128
9.1.3 *Appel depuis la liste* 130

9.2 Mode Debug 130
9.2.1 *Point d'arrêt* 130
9.2.2 *Perspective "Flash Debug"* 131
9.2.3 *Affichage de la propriété data* 132

9.3 Personnalisation de l'ActionBar 133
9.3.1 *Rappels* 133
9.3.2 *Zone de recherche* 134

9.4 Améliorations supplémentaires 135
9.4.1 *Suppression du bouton Accueil* 135
9.4.2 *Optimisation du chargement des données* 135

9.5 Retour de la vue "Poster" 136
9.5.1 *Déclarations dans "WesternsPoster.mxml"* 136
9.5.2 *Navigation depuis "WesternsDetails.mxml"* 137
9.5.3 *Retour vers le détail* 137

9.6 Nouvelle vue 138
9.6.1 *API IMDB* 138
9.6.2 *Navigation depuis "WesternsDetails"* 138
9.6.3 *Value Object IMDbvo* 139
9.6.4 *Vue "WesternsIMDbDetails"* 141
9.6.5 *Autres informations* 141
9.6.6 *Requête HTTPService* 143
9.6.7 *Gestion de la requête HTTP* 145

Chapitre 10 – Modèles de projets mobiles 147

10.1 Différents modèles de projets mobiles 147
10.1.1 *ViewNavigatorApplication* 147

10.1.2 Trois modèles disponibles 148
10.1.3 Variante avec onglets 150
10.1.4 Fonctionnement du modèle 152
10.2 Manipulation d'une TabbedApplication 153
10.2.1 Nouveau projet 154
10.2.2 Exploration du projet 155
10.2.3 Exécution de l'application 155
10.2.4 Code source 156
10.2.5 Utilisation d'icônes 157
10.2.6 Assistants 159
10.2.7 Personnalisation des ActionBars 159
10.2.8 Onglets et programmation 161

Quatrième partie – Finaliser une application mobile

Chapitre 11 – Utiliser les capacités du mobile 165
11.1 API AIR 165
11.1.1 APIS disponibles 165
11.1.2 Limites et évolutions 166
11.1.3 Adobe AIR Launchpad 167
11.1.4 Se reporter aux exemples 168
11.2 Géolocalisation 170
11.2.1 État inactif 171
11.2.2 Exploitation de l'API 172
11.2.3 Sur un appareil Android 173
11.2.4 Testez 174
11.2.5 Pendant la fermeture, l'activité continue 175
11.2.6 Précision 176
11.3 Gestion du clavier virtuel 177
11.3.1 Caractéristiques 177
11.3.2 API de redimensionnement 177
11.3.3 Exemple VirtualKeyboard 178
11.3.4 Gestion manuelle de l'affichage du clavier 181
11.3.5 Astuces 181

11.3.6 Attention au multiplateforme 182

11.4 APIS natives Sur *Westerns* 182

11.4.1 API StageWebView 182

11.4.2 Intégration avec les applications natives 186

Chapitre 12 – Interface tactile - Stockage local 191

12.1 Multi-touch 191

12.1.1 Contexte d'interaction tactile 192

12.1.2 Gestuelles 193

12.1.3 Gestion du multi-touch 194

12.2 Application à Westerns 195

12.2.1 Quatre gestuelles prises en charge 195

12.2.2 Refactoring de l'application 197

12.2.3 Navigation depuis "WesternsDetails" 204

12.2.4 Le courage de remanier 207

12.3 Stockage Local 207

12.3.1 Différents mécanismes de persistance 207

12.3.2 Données relationnelles 208

12.3.3 Application à Westerns 209

12.3.4 Compléments 214

Chapitre 13 – Persistance des données 215

13.1 Interrompu à tout moment ! 215

13.1.1 Succession de courtes sessions 215

13.1.2 Persistance entre deux sessions 215

13.1.3 PersistenceManager 217

13.1.4 Aller plus loin 219

13.2 Exemples 219

13.2.1 Persistance de DestructionPolicy 219

13.2.2 Persistance à la demande 223

13.2.3 Persistance de Westerns 225

Chapitre 14 – Gestion de l'orientation de l'écran 229

14.1 Deux écrans en un 229

14.1.1 Fixer l'orientation 230

14.1.2 S'adapter au changement 230

14.2 États et orientation 230

14.2.1 Flex et les états 230

14.2.2 Exemple sur un bouton 231

14.2.3 Correspondance orientation-état 232

14.2.4 Redimensionnement d'un bouton 232

14.2.5 Mode Design 233

14.3 Détection d'événements 233

14.3.1 API de gestion de l'orientation 233

14.3.2 Application au projet DestructionPolicy 235

14.4 Application à *Westerns* 236

14.4.1 Adaptation de la liste des films 237

14.4.2 Adaptation des autres vues 240

Chapitre 15 – Finalisations de l'application 243

15.1 Écran de démarrage (*Splash Screen*) 243

15.1.1 Limitations 243

15.1.2 Propriétés 244

15.1.3 Application à Westerns 246

15.2 Affichage d'un indicateur d'activité 246

15.2.1 Types d'indicateurs d'activité 246

15.2.2 BusyIndicator 246

15.2.3 Application à Westerns 247

15.3 Gestion du menu Android 249

15.3.1 Conteneur ViewMenu 249

15.3.2 Application à Westerns 251

15.4 Transitions 254

15.4.1 Les quatre transitions de Flex 255

15.4.2 Ajout d'effets (easing class) 257

15.4.3 Autres conteneurs 258

15.4.4 Application à Westerns 258
15.4.5 Ajout d'effets (easing class) 261
15.5 Forcer l'arrêt de l'application 261
15.5.1 Économiser la batterie 262
15.5.2 Complétion du projet TR_Geo 262
15.5.3 Remarques 262
15.6 Conserver certaines vues 263
15.6.1 Mémoire ou processeur 263
15.6.2 Résultats de requêtes 263
15.6.3 Propriété destructionPolicy 263
15.6.4 Dans Westerns 264
15.6.5 Synthèse 264

Cinquième partie – S'adapter au multi-écrans

Chapitre 16 – De bonnes résolutions 267
16.1 Multiples résolutions 267
16.1.1 État des lieux 267
16.1.2 Classification des résolutions 268
16.1.3 La résolution dans "Device Configurations" 269
16.1.4 Classes de résolutions 270
16.1.5 Répartition des résolutions pour Android 270
16.2 Gérer la résolution dans Flex 271
16.2.1 Déclarer la résolution par défaut 271
16.2.2 Adaptation automatique 273
16.2.3 La syntaxe @media pour les CSS 273
16.2.4 Une image par densité 275
16.2.5 Flash Builder 276
16.2.6 Récapitulatif des fonctions Flex liées à la résolution 276
16.3 Mise en œuvre sur Westerns 278
16.3.1 Mode « Design » 278
16.3.2 Test des configurations 278
16.3.3 ADL 279
16.3.4 Exemple de code 279

16.3.5 Bitmap Multi-resolution 280
16.3.6 CSS et @media 282
16.4 Compléments 284
16.4.1 Habillages pour Mobiles 284
16.4.2 Passer outre le calcul automatique par Flex 285

Chapitre 17 – Stratégies multi-écrans 287
17.1 Les problèmes à résoudre 287
17.1.1 Des écrans vraiment très différents 287
17.1.2 Expériences utilisateur 289
17.1.3 Différentes techniques 291
17.2 Tenir compte du système d'exploitation 292
17.2.1 Code dépendant de l'OS 292
17.2.2 Synthèse des différences sur l'ActionBar 293
17.2.3 Adaptation dynamique 294
17.2.4 Appliquer les règles @media 295
17.2.5 Autres éléments de différentiation 297
17.3 Réutilisation du code 300
17.3.1 Application grand public 300
17.3.2 Application métier en mode itératif 302
17.4 Déclinaison de Westerns pour tablette 303
17.4.1 Déclinaisons multiplateforme 303
17.4.2 Modèle fréquent pour tablette 303
17.4.3 Westerns sur tablette 304
17.4.4 Architecture logicielle 309

Index 311

2

Premiers pas avec Flash Builder

Objectif

Dans ce chapitre, vous créerez votre premier projet mobile afin de vous familiariser avec l'outil de développement Flash Builder qui nous servira tout au long de cet ouvrage.

2.1 PRÉALABLE

2.1.1 Version US

Quoiqu'une version localisée pour le français existe, les développeurs expérimentés préfèrent généralement employer la version internationale (anglaise). Elle permet de s'y retrouver beaucoup plus facilement lors des échanges et du suivi des différents tutoriels que l'on trouve sur Internet et qui sont majoritairement en anglais. Autant prendre de bonnes habitudes tout de suite, **les copies d'écran et les instructions relatives à Flash Builder se référeront donc à la version internationale**.

2.1.2 Contexte de travail

Dans cet ouvrage, vous apprendrez à développer des applications Flex autonomes pour Google Android, Apple iOS et BlackBerry TabletOS. Nous utiliserons pour cela l'IDE Flash Builder avec le Flex SDK 4.5.x. Si ces notions ne sont pas claires, je vous invite à vous reporter au chapitre précédent 01 *La plateforme Flash*.

2.1.3 Environnement de travail

Reportez-vous à l'annexe de ce chapitre pour la marche à suivre afin d'installer votre environnement de travail. Si vous n'avez pas encore téléchargé l'archive des ressources complémentaires depuis le site, c'est le moment de le faire[1] !

Pour suivre les exercices de ce chapitre et des suivants, vous devez avoir installé au préalable Flash Builder version 4.5.x version internationale sur votre machine.

Note pour les utilisateurs d'un Mac : Flash Builder 4.5.x n'est pas supporté sur Mac OS X 10.5.x. Vous devez installer au minimum une version 10.6.

2.2 PREMIER EXEMPLE

2.2.1 L'application Westerns

Commençons ensemble le développement de **l'application Westerns** qui servira de fil rouge aux mises en pratique de cet ouvrage d'apprentissage. Cette application a pour objectif d'afficher une liste de Westerns célèbres, de consulter certaines informations stockées localement, d'interroger le site **IMDb** (http://www.imdb.fr/) pour obtenir des informations complémentaires.

La figure 2.1 présente l'aperçu d'une liste de la version finale après application d'une feuille de styles personnalisés.

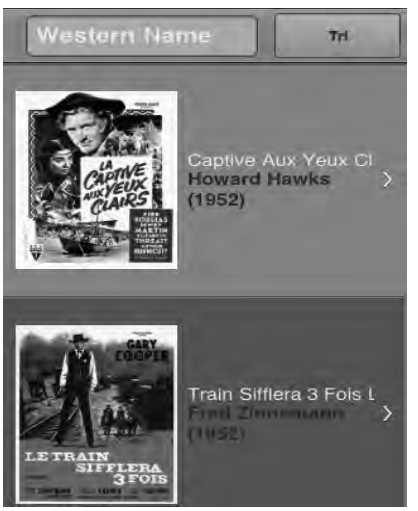

Figure 2.1 — Liste des Westerns

1. Cf. le paragraphe *Compléments en ligne* dans l'avant-propos.

Cette application se cantonne à des fonctionnalités simples, mais qui nous permettront de mettre en pratique les principales connaissances nécessaires à la réalisation d'applications mobiles avec Flex 4.5.x.

L'objectif dans ce premier exemple est d'acquérir les fondamentaux d'utilisation de Flash Builder. Je reviendrai en détail au fur et à mesure des chapitres suivants sur les différents concepts abordés lors de ces manipulations[1].

2.2.2 Ouverture de Flash Builder

Vocabulaire Eclipse

Quelques définitions de termes employés pour désigner les composants d'Eclipse et par héritage ceux de Flash Builder sont présentés dans le tableau 2.1.

Tableau 2.1 — Définitions des termes utilisés dans Eclipse

workbench	L'environnement de travail proposé à l'écran.
perspective	Groupe et disposition de vues et d'éditeurs dans le *workbench*. Flash Builder propose par défaut deux perspectives : **Flash** pour le développement de code et d'interface et **Flash Debug** pour le débogage d'applications. Dans la version Premium vous pouvez accéder en sus à la perspective de **Flash Profiling**.
view	Une vue propose un éditeur ou un outil (« Package Explorer » par exemple).
éditeur	Un éditeur permet d'éditer une famille de fichiers, par exemple un éditeur pour les fichiers MXML, un autre pour les fichiers ActionScript, un autre pour les feuilles de style CSS...

Ouverture Flash Builder

Lancez Flash Builder. L'écran d'accueil s'affiche comme sur la figure 2.2. Adobe a effectué un gros effort pour que cet écran d'accueil constitue un carrefour d'orientation vers les différentes ressources d'apprentissage.

Création du projet

Vous allez créer le premier projet.

Un projet rassemble les différents constituants d'une application : ressources (images, fichiers externes, feuilles de styles...), définitions d'écrans en MXML, code source en ActionScript. Ces éléments sont stockés sur le disque sous la forme de fichiers physiques rangés dans des dossiers.

Flash Builder utilise un espace de travail (*workspace*) correspondant à un dossier physique sur le disque où il stocke des informations sur les projets. Le dossier

1. Pour une description complète des fonctionnalités de Flash Builder, se référer à la documentation qui fait plus de 300 pages !

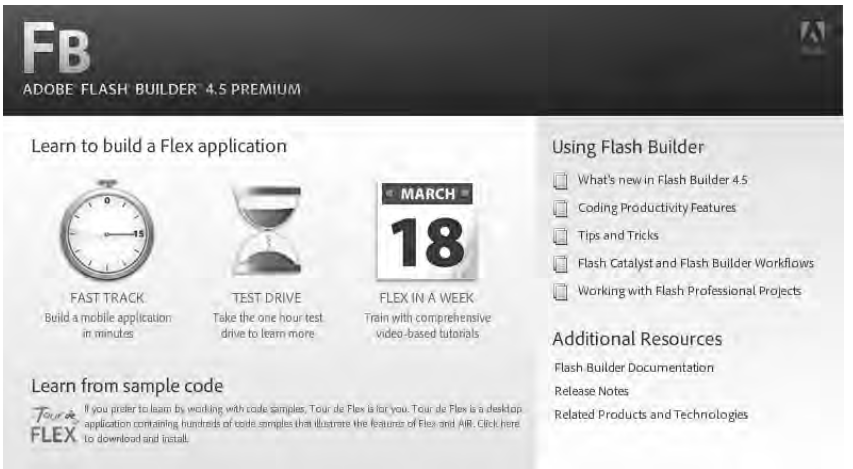

Figure 2.2 — Écran d'accueil Flash Builder

physique d'un projet (contenant les différents constituants) n'a pas besoin d'être stocké physiquement sur le disque dur dans le dossier de l'espace de travail.

Pour ce premier exemple, la création du dossier du projet s'effectuera dans le *workspace* par défaut. Sachez qu'il est possible à tout moment de créer un nouveau dossier de travail ou de basculer vers un *workspace* existant comme l'illustre la figure 2.3.

Attention, basculer vers un espace de travail différent relance en fait Flash Builder.

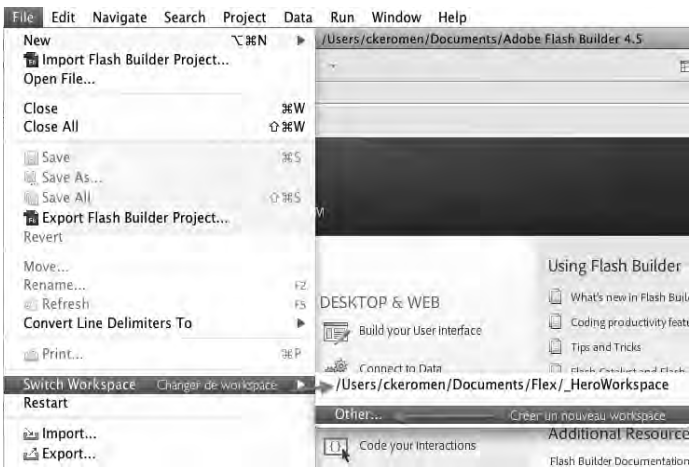

Figure 2.3 — Changement de *workspace*

Assistant de création du projet

Sélectionnez **File > New > Flex Mobile Project** pour créer un projet d'application mobile comme dans la figure 2.4.

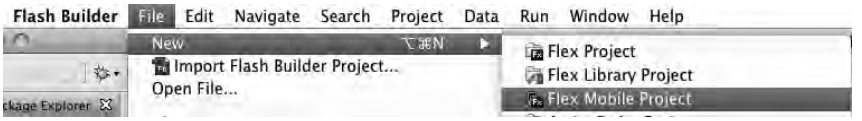

Figure 2.4 — New Flex Mobile Project

L'assistant de l'IDE vous propose alors un assistant de création de projet dont la première page se nomme **Project location** (figure 2.5).

Figure 2.5 — Project Location

Vous pouvez définir le nom du projet, écrivez **MyMobileProject**.

Laissez l'emplacement de création par défaut dans le *workspace* courant (case à cocher **Use default location**).

> Notez que ce projet utilisera le SDK par défaut (Flex 4.5.x). Il est possible de choisir la version du SDK au moment de la création du projet : Flash Builder 4.5.x autorise la création de projets utilisant par exemple le SDK 3.

Cliquez sur **Next >** pour accéder à l'écran **Mobile Settings** (figure 2.6).

Cochez simplement l'option **Full screen** sur cet écran afin de masquer la barre d'état de l'OS et d'occuper tout l'écran lors de l'exécution[1].

1. Nous détaillons cet écran dans le paragraphe *Onglets Mobile Settings*.

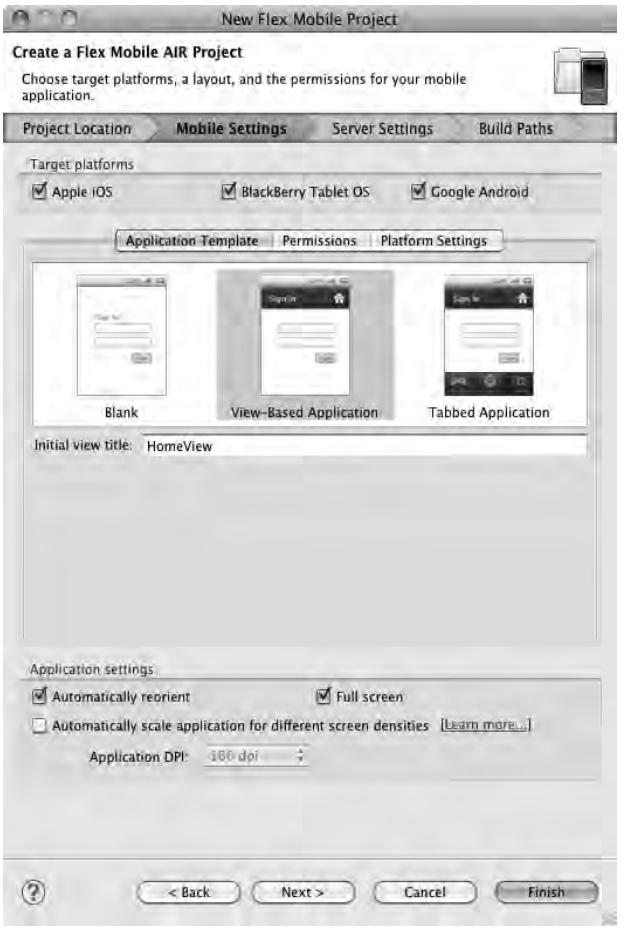

Figure 2.6 — Écran mobile Settings

Cliquez directement sur **Finish** sans passer par les autres étapes de configuration.

> Notez que vous pouvez sur chaque panneau de l'assistant revenir à l'étape précédente en cliquant sur le bouton **< Back**. Si l'onglet du haut de la fenêtre prend une couleur bleue pour indiquer l'étape courante, il n'est pas possible de l'utiliser pour naviguer d'une étape à l'autre : il s'agit d'un simple indicateur visuel.

2.2.3 Le projet créé

Dossier du Projet

Flash Builder crée dans le dossier du *workspace* courant une arborescence de dossiers et de fichiers sur le disque (figure 2.7).

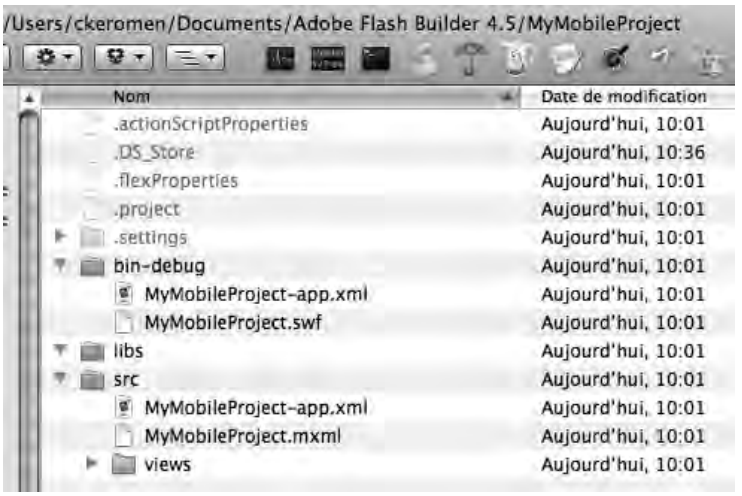

Figure 2.7 — Arborescence du contenu du projet sur le disque

Vous constatez sur la figure 2.7 que le dossier sur le système de fichiers contient des éléments visibles et d'autres invisibles (fichiers en gris et dont le nom est précédé d'un point). Ces derniers correspondent à des métadonnées gérées par Flash Builder.

> Par défaut, ces fichiers invisibles sont masqués à l'utilisateur. Il faut modifier les réglages systèmes pour les visualiser. Les manipulations nécessaires dépendent des systèmes d'exploitation et ne sont pas requises sur votre poste pour utiliser Flash Builder.

Vue logique

L'arborescence du projet est représentée de manière logique dans la vue **Package Explorer** en haut et à gauche de la perspective de développement. Cliquez sur la flèche à gauche de **MyMobileProject** pour déployer le projet.

Flash Builder a créé un sous-dossier **src** à l'intérieur du dossier du Projet. Le dossier **src** contient lui-même deux sous-dossiers :

- **(default package)** qui est un niveau logique (d'où les parenthèses) rajouté par Flash Builder. Vous pouvez constater que ce niveau hiérarchique n'existe pas sur votre disque dur. Vous y trouvez le fichier principal de l'application **MyMobileProject.mxml**. Si vous y prêtez attention, l'icône de ce fichier présente un point bleu qui désigne l'application par défaut du Projet. Cela signifie que ce fichier sera exécuté au lancement de l'application. C'est en quelque sorte le « main » (fichier principal exécuté au démarrage) de notre application.

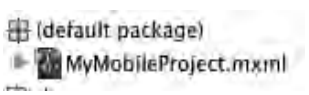

Figure 2.8 — Application par défaut

- **views**, dossier physique qui a vocation à contenir les différentes vues de l'application. L'assistant a créé une vue par défaut portant le nom du projet suivi de **HomeView** : **MyMobileProjectHomeView.mxml**. Considérez qu'il s'agit d'un écran présenté à l'utilisateur[1].

Arborescence finale du projet

À l'issue de ce livre, le projet de l'application aura évolué pour ressembler à l'arborescence présentée dans la figure 2.9.

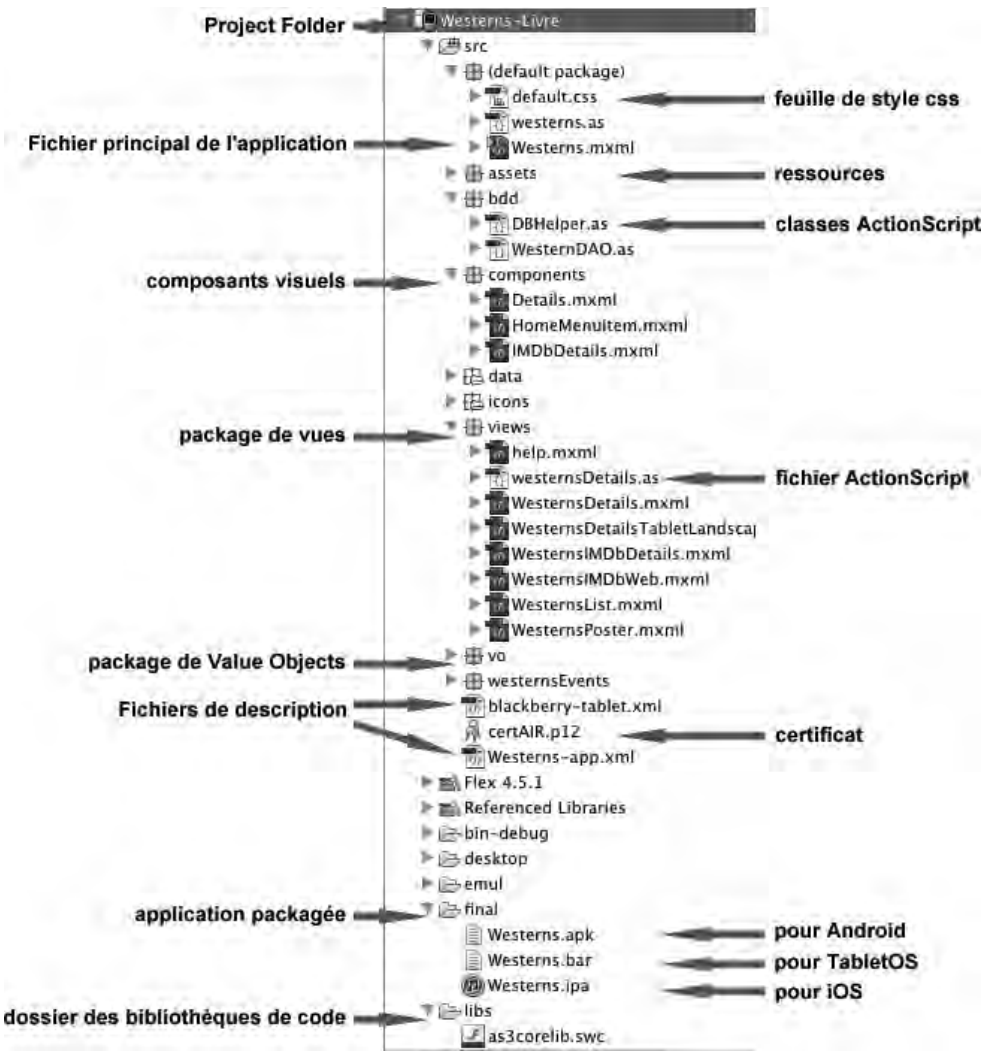

Figure 2.9 — Projet final

1. Nous reviendrons dans le chapitre 8 *Architecture Navigateur / Vue* sur cette notion de vue.

Le tableau 2.2 explicite les différents termes.

Tableau 2.2 — Différents fichiers et dossiers de l'application

Regroupement	Description
Fichier principal de l'application	Obligatoire, il est créé lors de la création du projet. Contient le code source exécuté pour lancer l'application.
Package de bibliothèque de code (*Library*)	Permet de découper son code en paquets réutilisables entre plusieurs projets. Important pour une stratégie multi-écrans
Package de vues	Regroupe les vues (écrans) de l'application, c.-à-d. ce qui concerne l'interface utilisateur
Fichier de description (*descriptor file*)	Fichier XML contenant les paramètres d'exécution de l'application en fonction de chaque plateforme. C'est par exemple dans ce fichier que se décrivent les différentes permissions requises par l'application lors de l'installation sur un mobile Android. Vous trouverez plus d'informations sur cet aspect dans le fichier pdf disponible en annexe "Versions de débogage et de déploiement".
Application packagée	Archive suffixée .apk générée par Flash Builder pour le déploiement sur un périphérique Android. Pour iOS, il s'agira de .ipa et pour TabletOS de .bar.
Feuille de styles CSS	Contient les règles de style applicables aux différents éléments de l'application afin de modifier leur apparence visuelle[a].
Classes ActionScript	Ces fichiers contiennent du code ActionScript 3 permettant de définir la logique métier de votre application. Ces fichiers sont souvent des classes ActionScript. Elles peuvent par exemple représenter les structures de données manipulées par l'application (communément appelées VO pour *Value Object*).

a. Nous verrons dans le chapitre 16 *De bonnes résolutions* que ces règles peuvent être conditionnées par la plateforme et/ou la résolution.

2.2.4 Perspective Développement

Sélectionnez **MyMobileProject.mxml** dans les onglets des différents éditeurs ouverts ou double-cliquez sur son nom dans le **Package Explorer**. Ici l'éditeur de code source MXML est affiché pour éditer le fichier **MyMobileProject.mxml**.

Cliquez maintenant sur l'onglet **Design** pour basculer de l'édition de code vers l'édition graphique de l'interface. Vous devez obtenir un écran similaire à la figure 2.10.

Résumons dans le tableau 2.3 les principaux points intéressants dans cette perspective.

INFOPRO

TYPE D'OUVRAGE		
L'ESSENTIEL	**SE FORMER**	RETOURS D'EXPÉRIENCE

Christophe Keromen

Préface Michaël Chaize

MANAGEMENT DES SYSTÈMES D'INFORMATION

APPLICATIONS MÉTIERS

ÉTUDES, DÉVELOPPEMENT, INTÉGRATION

EXPLOITATION ET ADMINISTRATION

RÉSEAUX & TÉLÉCOMS

FLEX 4.5 POUR MOBILES

Développer des applications multiplateformes pour iOS, Android et Tablet OS

Ce ouvrage s'adresse non seulement aux développeurs de la communauté Flex mais aussi à tous les développeurs qui souhaitent créer des applications « mobiles » pour les entreprises, ou des applications pour les différents « *stores* ».

Aujourd'hui les applications métiers des entreprises sont de plus en plus portées sur smartphones et sur tablettes tactiles, mais la variété des environnements de développement complique le travail des équipes.

Cet ouvrage vous guide dans l'apprentissage du *framework* Flex d'Adobe qui propose de bâtir des applis multiplateformes grâce à un modèle unifié de programmation, d'environnement de développement et de base de code.

Conçu comme un **manuel d'auto-formation** il vous permet une mise en pratique immédiate des notions présentées par le biais d'un **projet-exemple** qui sert de fil rouge aux différents chapitres.

Quand vous maîtriserez le contenu de cet ouvrage vous serez à même de déployer des applications mobiles sur Apple **iOS**, Google **Android** et BlackBerry **Tablet OS**.

CHRISTOPHE KEROMEN

est ingénieur en informatique. *Technical Evangelist* pour l'éditeur de logiciel 4D jusqu'en 2009, il est aujourd'hui consultant indépendant et formateur dans le domaine des applications mobiles. Il est le créateur d'un *Flex user's group* en Bretagne.

Retrouvez les **exemples de code** téléchargeables gratuitement sur **www.dunod.com** sur la page dédiée à l'ouvrage.

9 782100 567065

6945323

ISBN 978-2-10-056706-5