Programmation Flex 4

Applications Internet riches avec Flash ActionScript 3, Spark, MXML et Flash Builder

Aurélien Vannieuwenhuyze

EYROLLES

Programmation Flex 4

Applications Internet riches avec Flash ActionScript 3, Spark, MXML et Flash Builder

Flex 4, framework de référence pour le développement de clients et d'applications bureautiques riches Flash

Open source, le SDK de Flex offre un véritable environnement en phase avec les bonnes pratiques de génie logiciel (MVC…) et de gestion de projet (travail collaboratif…). Il propose des bibliothèques de composants graphiques (Spark…) et des fonctions pour dialoguer avec le serveur et s'interfacer avec des bases de données via des serveurs PHP/JEE.

Une bible avec de nombreux cas pratiques et retours d'expérience

Programmation Flex 4 explique aux développeurs web, qu'ils soient ou non rompus à Flash et ActionScript, comment utiliser le framework Flex pour concevoir et créer des applications web dites riches (RIA), à l'instar des applications Ajax ou Silverlight. Tout en rappelant les langages sur lesquels s'adosse cette technologie (ActionScript, MXML), l'ouvrage passe en revue l'intégralité des techniques de développement Flex : maîtrise de l'environnement de travail, création d'interfaces interactives et évoluées avec les vues, transitions et thèmes, gestion des données et échanges avec le serveur via RPC, mais aussi gestion et création des composants, débogage et optimisation, tests unitaires avec Flex Unit. Il les met ensuite en situation avec deux études de cas détaillant la création d'un catalogue interactif (où la gestion des produits met en œuvre Adobe Air 2), puis l'incorporation de l'API de géolocalisation de Google.

Enrichi de retours d'expérience d'experts, cet ouvrage fait la part belle aux aspects graphiques, ainsi qu'au design et au développement par composant. Il insiste sur l'interopérabilité avec la suite Adobe via Flash Catalyst CS5, ainsi que sur la manipulation d'objets 3D grâce à la bibliothèque PaperVision 3D. Il présente également l'intégration de composants et d'API externes (Google Maps…) ainsi que la compatibilité avec d'autres frameworks tels .NET et le très en vogue Spring.

Au sommaire

Comprendre Flex • Pourquoi choisir Flex 4 ? • À la découverte du framework Flex 4 • Développer avec le kit Flex SDK • Un premier outil de développement : FlashDevelop 3.1.1 • Développer des applications à l'aide de Flash Builder 4 • Spark, Halo, FXG, passerelles entre les composants et le design d'interface • Réaliser le design des interfaces • Le dynamisme applicatif avec les états, les transitions et les effets • La 3D avec PaperVision 3D • Créer un lecteur RSS avec Flash Catalyst CS5 • Créer des composants personnalisés • Gérer les données • Accéder aux services de la couche métier • Flex et PHP avec zend_AMF • Blaze DS ouverture open source vers le monde Java et la communication d'applications • Flex et .Net • Créer des applications modulaires • Interagir avec le conteneur web • Internationalisation des applications • Déboguer une application • Analyser les performances et optimiser l'application • Les tests unitaires avec Flex Unit • Adobe Air 2 • **Cas pratiques** • Création d'un projet e-commerce : un catalogue interactif • Module d'administration avec Adobe Air : exploitation d'une webcam et manipulation d'images • Mise en place du catalogue interactif • Gestion du panier d'achat et déploiement du e-catalogue • Google Maps dans une application Flex • **Annexes** • Les composants de contrôle • Flex, BlazeDS 4, Tomcat 7, Eclipse Hélios et Spring 3.0

À qui s'adresse cet ouvrage ?

– À toute la communauté des concepteurs et développeurs web qui désirent développer des applications Internet riches.
– Aux webmestres et développeurs connaissant Flash et ActionScript, qui recherchent un framework pour industrialiser la création d'interfaces graphiques pour leurs applications.
– Aux développeurs amenés à créer des applications autonomes exécutables avec ou sans navigateur web.

Sur le site www.editions-eyrolles.com
– Téléchargez le code source des exemples et cas pratiques du livre

A. Vannieuwenhuyze

Architecte Flex chez un célèbre éditeur, membre de l'association Outcomerea, spécialisée dans le domaine de la recherche médicale, Aurélien Vannieuwenhuyze s'est d'abord investi dans la conception et la réalisation d'applications JEE.
Pour créer des applications Internet riches, il a étudié les frameworks alliant richesse et rapidité de réalisation. Son choix s'est porté sur Flex, avec lequel il a réalisé des applicatifs à vocation médicale, des refontes d'applications et des extranets pour de grandes entreprises.

F. Nicollet

Ingénieur développeur en géomatique, Fabien Nicollet a créé en 2008 flex-tutorial.fr contenant des centaines de tutoriaux Flex/ActionScript.

Conception : Nord Compo

www.editions-eyrolles.com
Groupe Eyrolles | Diffusion Geodif

Programmation
FLEX 4

P. Chatelier. – **Objective-C pour le développeur avancé.**
N°12751, 2010, 224 pages.

E. Sarrion. – **XHTML/CSS et JavaScript pour le Web mobile.**
N°12775, 2010, 274 pages.

J. Stark. – **Applications iPhone avec HTML, CSS et JavaScript.**
N°12745, 2010, 190 pages.

D. Guignard, J Chable, E. Robles, N. Sorel. – **Programmation Android.**
N°12587, 2010, 486 pages.

P. Borghino, O. Dasini, A. Gadal. – **Audit et optimisation MySQL 5.**
N°12634, 2010, 266 pages.

J.-M. Defrance. – **Premières applications web avec Ajax, jQuery et PHP.**
N°12672, 2010, 474 pages.

T. Sarlandie. – **Programmation iPhone OS 3.**
N°12477, 2009, 250 pages.

D. Tardiveau. – **ActionScript 3.**
N°12552, 2009, 426 pages.

G. Leblanc. – **Silverlight 2.**
N°12375, 2008, 330 pages.

G. Ponçon et J. Pauli. – **Zend Framework.**
N°12392, 2008, 460 pages.

E. Daspet et C. Pierre de Geyer. – **PHP 5 avancé.**
N°12369, 5e édition, 2008, 844 pages.

C. Porteneuve. – **Bien développer pour le Web 2.0.**
N°12391, 2e édition, 2008, 600 pages.

R. Goetter. – **CSS 2 : pratique du design web.**
N°12461, 3e édition, 2007, 318 pages.

E. Sloïm. – **Sites web.** *Les bonnes pratiques*.
N°12802, 2010, 18 pages.

S. Bordage, D. Thévenon, L. Dupaquier, F. Brousse. – **Conduite de projets Web.**
N°12665, 5e édition, 2010, 432 pages.

N. Chu. – **Réussir un projet de site Web.**
N°12742, 6e édition, 2010, 246 pages.

O. Andrieu. – **Réussir son référencement web.**
N°12646, 2009, 462 pages.

D. Séguy, P. Gamache. – **Sécurité PHP 5 et MySQL.**
N°12554, 2009, 268 pages.

R. Rimelé. – **Mémento MySQL.**
N°12720, 2010, 14 pages.

M. Nebra. – **Réussir son site web avec XHTML et CSS.**
N°12485, 2e édition, 2010, 318 pages.

Programmation FLEX 4

Applications Internet riches avec ActionScript 3, Spark et Flash Builder

Aurélien Vannieuwenhuyze

Avec la contribution de Fabien Nicollet

EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

Remerciements à Romain Pouclet pour sa relecture sur la première édition et à David Feugey pour ses précieux conseils

Table des matières

Avant-propos 1

De la théorie à la pratique 1

Structure de l'ouvrage 2

À qui s'adresse ce livre ? 3

Remerciements 4

CHAPITRE 1

Pourquoi choisir Flex 4 ? 5

Les enjeux du développement des applications web 5

Flex en questions 7

Est-ce une technologie récente ? 7

Qu'est-ce que Flex ? 8

Quel est son fonctionnement ? 9

Sur quelles technologies repose-t-il ? 9

Choisir la méthode open source ou propriétaire ? 10

Flex : quelles différences avec Ajax ? 11

Flex et ses concurrents 13

Flex et Silverlight 13

Silverlight en quelques mots 14

Un début de comparaison 15

Les limites de Flex 16

Pourquoi choisir Flex ? 17

Qu'apporte Flex 4 ? 18

Faut-il migrer les applications Flex 3 vers Flex 4 ? 20

CHAPITRE 2

À la découverte du framework Flex 4 23

Rôle du front end 23
Architecture n-tiers 23
Positionnement d'une application Flex 24

L'application et le modèle-vue-contrôleur (MVC) 25

Le langage MXML 27

Le langage ActionScript 28
Les variables 28
Les boucles et les conditions 29

Le mécanisme client/serveur 39

En résumé 42

CHAPITRE 3

Développer avec le kit Flex SDK 45

Prérequis 46

Installation du kit Flex SDK 47

Les compilateurs 48
mxmlc : le compilateur d'application 48
compc : le compilateur de composants 49

Première application Flex 50
Conception graphique 50
Création du projet 50
Écriture du fichier MXML 51
Écriture du fichier ActionScript 52
Liaison graphique et action 53
Compilation 54
Intégration dans une page web 55
Déploiement 56

Automatisation de la compilation à l'aide de Ant 56
Ant et Flex 4 57
Installation de Ant 57
Création d'un script de compilation 57

En résumé 62

CHAPITRE 4

Un premier outil de développement : FlashDevelop 3.1.1 63

Comprendre FlashDevelop 63

Installer FlashDevelop 64

Étapes d'installation 64

Installer les modules de débogage complémentaires 66

L'incontournable Hello World 66

Créer le projet 66

Utiliser le SDK de Flex 4 68

Un peu de code 69

Compiler (build) le projet 70

Déboguer le projet 71

En résumé 72

CHAPITRE 5

Développer des applications à l'aide de Flash Builder 4 73

Présentation de Flash Builder 73

Les différentes versions de Flash Builder 74

Les différents modes de distribution de Flash Builder 74

Installer Flash Builder 4 75

Le mode Autonome 75

Le mode plug-in 79

Les perspectives de Flash Builder 81

La perspective de développement 82

La perspective de design 83

La perspective de débogage 85

La perspective de profiling 87

Créer un projet 88

Arborescence d'un projet Flash Builder 90

Exécution et tests 94

Exporter/importer un projet 94

Exportation 94

Importation 95

En résumé 96

CHAPITRE 6

Spark, Halo, FXG, passerelles entre les composants et le design de l'interface 97

Spark en définition 97

Que sont devenus les anciens composants ? 98

Les apports significatifs de Spark 98

La bibliothèque graphique FXG 100

L'habillage (skinning) des composants Flex 4 102

Créer un habillage ou skin personnalisé 103

Définition 103

Créer une première skin 103

Exploiter l'habillage dans une l'application 111

En résumé 112

CHAPITRE 7

Réaliser le design des interfaces 115

Dimensionner et positionner les composants 115

Spécifier les dimensions des composants 115

Positionner les composants 117

Les styles 125

La balise Style 126

Le fichier CSS externe 127

CSS et ActionScript 127

Les thèmes 128

En résumé 130

CHAPITRE 8

Le dynamisme applicatif avec les états, les transitions et les effets 131

Les états ou view states : étude de cas 131

Création de l'état de base 132

Création de l'état enfant 134

Mise en place de l'interactivité 137

Les transitions 137
Combiner des effets 139
Mise en pratique 140

Améliorer l'interactivité avec des effets et la personnalisation d'éléments d'interface 143
Mise en pratique de quelques effets 143
La classe Animation 148
Modifier le curseur de la souris 151
Un loader d'application personnalisé 152

En résumé 155

CHAPITRE 9

La 3D avec PaperVision 3D 157

Réaliser de la 3D avec Flash Builder 157
Créer l'interface 157
Le code de l'application 159
Intégrer un objet 3 D dans une application Flex 161
Nos outils 161
Créer le projet et importer les bibliothèques PaperVision3D 162
Créer l'interface graphique 164
Le code ActionScript 165
En résumé 168

CHAPITRE 10

Créer un lecteur RSS avec Flash Catalyst 169

Installation 169
Premier projet avec Flash Catalyst 170
Le design de l'interface graphique avec Photoshop 170
Importer des éléments graphiques dans Flash Catalyst CS5 171
Composition de l'interface 172
Un peu d'animation 176
Importer l'interface dans un projet Flex 180

Appeler le flux RSS 182

Les limites de la procédure 186

En résumé 187

CHAPITRE 11

Créer des composants personnalisés 189

Penser composant 189
L'indispensable définition 189
Vers une application modulaire 190

Créer un composant 190
MXML ou ActionScript ? 190
Le compilateur compc 191
Description du composant 191
Le processus de création pas à pas 193

Réemployer le composant 214

Extraire des composants dans de vastes projets 216
Les limites de l'interface 217
Les propriétés de réutilisabilité 217

En résumé 219

CHAPITRE 12

Gérer les données 221

Lier des données 221
DataBinding bi-directionnel 223
La balise <fx:Binding> 223
Le DataBinding en ActionScript 224

Stocker des données grâce au modèle 229
Implémenter le modèle à l'aide de la balise <fx:Model> 229
Les modèles et ActionScript 233

Valider des données 238
Rendre la saisie obligatoire 239
Vérifier le format 240
Personnaliser les messages d'erreur 240
Les événements de validation 241
Gérer la validation en ActionScript 241
Mettre la validation en forme 244

Formater des données 244

En résumé 248

CHAPITRE 13

Accéder aux services de la couche métier 249

Employer les requêtes http avec HTTPService ou REST-Style webservice 249
Le composant WebService 255
Le service web en détail 256
Création de l'interface graphique 260
Appel du service web 261
Analyse du fonctionnement de l'application 262

Flash Builder et l'accès aux services web 264

Accéder aux classes distinctes grâce à RemoteObject 269
Comprendre RemoteObject 269
Implémenter RemoteObject 270

En résumé 271

CHAPITRE 14

Flex et PHP via zend_AMF 273

Accéder aux services PHP 273

Cas pratique : gestion de clients 273
Préparation 274
Créer le projet 281
Génération automatique d'interfaces graphiques 296

En résumé 301

CHAPITRE 15

BlazeDS ouverture open source vers le monde Java et la communication d'applications 303

Le projet BlazeDS 303
Les différentes versions du serveur BlazeDS 304
Installation 304
Le serveur Tomcat 306
Test d'installation 308

Interaction avec les méthodes J2EE 308
Processus d'exécution 308
Exemple d'application 310

Assistance à la connexion 316
Échanger des données en temps réel 318
Création du projet 319
Configuration du serveur 320
<mx:Producer> et <mx:Consumer> : les composants émetteurs et récepteurs 322
Les procédures de communication 323
Test de l'application 324
En résumé 325

CHAPITRE 16
Flex et .NET 327
Prérequis 327
Accéder au service .NET 327
Créer un service .NET 328
Écrire l'application Flex 328
Appeler le service .NET 330
Communication entre une application Flex et Silverlight 332
Un canal commun 333
Étude de cas 333
Le code de l'application Flex 334
Code de l'application Silverlight 335
En résumé 341

CHAPITRE 17
Créer des applications modulaires 343
Qu'est ce qu'un module ? 343
Création d'un module 344
Intégrer un module dans une application 345
ModuleLoader 346
Utiliser le ModuleLoader en ActionScript 346
Charger le module en ActionScript avec ModuleManager 351
Communication de l'application vers le module 354
Communication du module vers l'application 356

Communiquer avec l'application via les interfaces ActionScript 358
Diminuer le couplage des modules, via le mode évènementiel de Flash Player 360

Résumé 361

CHAPITRE 18

Interagir avec le conteneur web 363

La notion de deep linking 363
Principe de fonctionnement 363
Mise en pratique du deep linking 365
Déploiement du projet 369

Communiquer avec le conteneur web 369
De l'application Flex vers le conteneur web 370
Du conteneur web vers l'application Flex 373

La bibliothèque Flex-Ajax Bridge 377
Ajouter la bibliothèque à un projet 377
Exploiter la bibliothèque 378
Déploiement et test 380

En résumé 381

CHAPITRE 19

Internationalisation des applications 383

Les paramètres régionaux 383

Écrire une application multilingue 383
Spécifier les options de compilation 384
Créer l'interface graphique et les fichiers de traduction 385
Mettre en place de la traduction 386
Traduction dynamique 388

En résumé 391

CHAPITRE 20

Déboguer une application 393

Création de l'application d'étude 393

Identifier et corriger les erreurs grâce à la vue Erreurs 395

Le débogage avancé 396
Les points d'arrêt 396
Modifier la valeur des variables 398
Tracer un parcours 398
Sélectionner les variables à surveiller 399

En résumé 401

CHAPITRE 21

Analyser les performances et optimiser l'application 403

Présentation de l'analyseur 403

L'analyseur en action 404
Analyse des performances 405
Analyse de l'occupation mémoire. 407

Monitoring réseau 408

Méthodes d'analyse 410
Détecter les allocations mémoire inutiles. 410
Analyser le temps d'exécution des méthodes 411
Déterminer le nombre d'instances d'une classe 411
Visualiser les allocations d'objets. 412

Bonnes pratiques d'optimisation 412
Calculer les performances applicatives 412
Améliorer l'exécution sur les postes clients 415
Réduire la taille des fichiers SWF 416
Multiplier les fichiers SWF. 417
Améliorer les méthodes de codage. 417

Déployer une application Flex 418
L'incontournable liste de contrôle 418
Mettre l'application en ligne 419

En résumé 420

CHAPITRE 22

Les tests unitaires avec Flex Unit 423

Comprendre Flex Unit 4 423

Flex Unit en pratique 423
Créer la classe métier 424

Créer la classe de test 424
Créer les cas de tests. 425
Mise en pratique du cas de test. 427

En résumé 428

CHAPITRE 23

Adobe AIR 2 429

Le projet Adobe AIR. 429
Fonctionnement d'une application AIR 429
Nouveaux composants 430

Première application AIR 432
Création du projet 432
Déploiement de l'application 435

Gestion des données avec la bibliothèque SQLite 436
Utiliser la bibliothèque SQLite dans une application AIR. 437
SQLite par l'exemple 438

Les nouveautés de AIR 2 446

En résumé 447

CHAPITRE 24

Création d'un projet e-commerce : un catalogue interactif 449

Présentation du projet 449

Conception UML. 450

Architecture du projet 451

Création du projet. 451
Préparation du serveur 452

Faisons le point 466

CHAPITRE 25

Module d'administration avec Adobe AIR. 467

L'interface graphique 467
Écran de connexion 468
Écran de gestion des produits. 468

Partie 2 : ajout d'un produit 470
À vos claviers 471

Le listing des catégories 475
Configurations 475
Appel du service getCategories() 476

Affichage des produits 479
Créer la classe Produit ActionScript. 479
Listing des produits 481

Utiliser la webcam 483
Connexion à la webcam grâce à ActionScript 483
Capture d'une image. 484
Manipulation d'une image 484
À vos claviers 485
Enregistrer une image sur le disque dur 485

Enregistrer un nouveau produit 487

Que reste-t-il à faire ? 488

CHAPITRE 26

Mise en place du catalogue interactif 489

Un grand merci 489

Utiliser le composant catalogue 490
Télécharger le composant 490
Créer un nouveau projet web 490
Notre premier catalogue 492

Préparation des pages 493
Les images de fonds 494
Mise en place des pages 495

Ajout de produits 497
À vos claviers 499
La solution 499

Ajout d'interactivité 503
Détail de l'image 503
Plus de navigation. 506

En résumé 508

CHAPITRE 27

Gestion d'un panier d'achat et déploiement du e-catalogue ... 509

Le projet ... 509

Le glisser-déposer ... 510

Quelques exemples utiles ... 512

Un mini panier ... 512

Du panier à la commande ... 517

Déployer l'application ... 520

Créer la base de données ... 520

Créer d'une version de déploiement ... 520

Paramétrer l'application ... 520

En conclusion ... 521

CHAPITRE 28

Google Maps dans une application Flex ... 523

S'inscrire aux services de cartographie de Google ... 523

Coupler l'API dans un projet Flex ... 524

Afficher notre première carte ... 525

Ajouter des options de visualisation ... 527

Prêt pour le départ en vacances ? Calculer un itinéraire ... 530

En résumé ... 535

ANNEXE 1

Les composants de contrôle ... 537

Les boutons et interrupteurs ... 537

Button ... 538

ToogleButton ... 538

PopUpButton et PopUpMenuButton ... 538

Permettre le choix ... 540

CheckBox : les cases à cocher ... 540

Les listes simples et déroulantes ... 541

Les boutons radio ... 544

Les zones de texte ... 546

Les libellés : Label ... 546

Les zones de saisie 546
Les zones de texte « riches » 547

Le tableau AdvancedDataGrid 549
Implémentation et alimentation en données 549
Interaction avec l'utilisateur 552

Un outil décisionnel : le cube OLAP 553
Qu'est-ce qu'un cube OLAP ? (OnLine Analytical Processing) 554
Création d'un cube OLAP 555
Requête sur le cube 557

Insérer des images, des animations et des vidéos Flash 562
Intégrer des animations Flash 562
Insérer des images 563
Le lecteur vidéo Flash 563
Le lecteur de vidéo avancé 564

Choisir dans une palette de couleurs avec ColorPicker 565

Contrôler le format de date avec DateChooser et DateField 566

Ajouter un curseur avec HSlider et VSlider 567

L'incrémentation avec NumericStepper 568

Réaliser des hyperliens grâce à LinkButton 568

Afficher des données sous forme d'arborescence 569
Implémentation et alimentation en données 569
Interaction avec l'utilisateur 570

Les composants graphiques 572
Liste complète des composants statistiques 572
Implémentation de PieChart 572

En résumé 575

ANNEXE 2

Flex, BlazeDS 4, Tomcat 7, Eclipse Helios et le framework Spring 3.0 577

Préparation de l'environnement 577
Une étude de cas à titre d'exemple 578

BlazeDS et Eclipse 579
Configurer le serveur Tomcat 579
Créer un projet utilisant BlazeDS 581
Un début d'implémentation de l'étude de cas 585
Créer les classes Java 585
Créer l'application Flex 587
Le framework Spring 591
Spring en quelques mots 591
Importer des bibliothèques de Spring 591
Importer des bibliothèques de Spring-Flex Integration 592
Organisation en package des classes Java 593
Modifier le chemin de compilation des classes Java 593
Configurer Spring et de BlazeDs 594
Tester l'application 595
En résumé 598
Index 599

Avant-propos

À l'ère du Web 2.0, toutes les applications web disposent de fonctionnalités avancées et affichent un design surprenant. Nous sommes aujourd'hui entourés de nouveaux termes tels que « client riche » (*RIA, rich Internet Application*) ou « application bureautique riche » (*RDA, Rich Desktop Application*).

Davantage de fonctionnalités se rapprochant autant que possible des clients lourds, un design très moderne et commercial, voilà ce que l'on demande aujourd'hui aux concepteurs et développeurs d'applications. Comment réaliser ceci dans des temps parfois record, car, ne l'oublions pas, il s'agit là de technologies novatrices et en plein essor ? C'est de ce besoin qu'est née la notion de framework, ensemble d'outils dédiés à la création et au développement de ces applications d'un nouveau genre. Flex 4, est un de ces frameworks, qui proposent de nombreuses fonctionnalités allant du design de l'interface à l'interaction avec les services métier jusqu'à l'exécution des tests unitaires.

C'est cet ensemble d'outils que nous allons découvrir à travers cet ouvrage qui, espérons-le, vous permettra d'apprécier tous les aspects de la conception, du développement et du déploiement d'une application avec le framework Flex 4.

De la théorie à la pratique

Lors de la rédaction de cet ouvrage, nous n'avons pas cherché à vous présenter un ensemble de concepts théoriques, qui, finalement, sont bien vite oubliés. En effet, un livre informatique est d'abord et avant tout une source d'enseignements où théorie et pratique s'entremêlent étroitement. Il accompagne le développeur tout au long de son travail, à chaque fois que celui-ci aura besoin d'une référence. C'est précisément dans cette optique que nous avons rédigé cet ouvrage. Chaque concept énoncé est donc illustré par un exemple commenté.

Cette nouvelle édition tient compte des remarques des lecteurs qui nous ont contacté par e-mail ou qui ont posté des commentaires sur les sites des librairies en ligne. Ainsi, nous avons modifié l'orientation de notre ouvrage de manière à l'axer davantage sur la fonction première de Flex, qui n'est autre que la réalisation d'interface graphique. Bien entendu, nous n'oublions pas de situer Flex dans son environnement et ne négligeons pas l'accès aux services de la couche métier. Vous découvrirez alors comment interfacer une application avec le framework .NET et une application Silverlight, mais également comment exploiter le framework Spring.

Nous avons également cherché à donner une dimension plus experte à notre ouvrage. De nombreuses parties seront donc plus amplement détaillées que dans l'édition précédente, et de nombreux encarts vous permettront d'approfondir les notions évoquées.

Structure de l'ouvrage

Pour qu'à l'issue de cet ouvrage, le développement d'application web et bureautique riche avec Flex n'ait plus de secret pour vous, nous en verrons donc toutes les facettes : de la conception de l'interface graphique au déploiement du projet en passant par la gestion des données et du contrôle de la saisie de l'utilisateur. La première partie de cet ouvrage couvre l'ensemble des concepts fondamentaux.

Présentation de Flex 4, le **chapitre 1** vous donnera une vue d'ensemble sur les capacités du framework, sa structure et ses points forts. Le **chapitre 2** se focalise ensuite sur la programmation de type MVC et situe Flex dans ce contexte.

Entrée en matière dans l'utilisation du framework, le **chapitre 3** propose l'étude du SDK et des différents compilateurs. Ce sera l'occasion de créer une première application Flex. Nous passerons alors au **chapitre 4** à la présentation d'un premier l'outil de développement Flash Developp. Nous poursuivrons ensuite par la découverte de l'environnement de développement Flash Builder 4 présenté dans le **chapitre 5.**

Le **chapitre 6** entre dans le vif du sujet des nouveautés du framework Flex 4 en présentant la bibliothèque graphique Spark.

Le **chapitre 7** présente les techniques d'agencement des composants afin de créer le design de nos applications. Dans le **chapitre 8,** nous verrons comment réaliser effets et transitions au sein de nos interfaces graphiques. Puis, le **chapitre 9** se penche sur PaperVision 3D, qui nous servira à intégrer des modélisations 3D réalisées avec des outils dédiés**.**

Au **chapitre 10**, nous étudierons l'état d'esprit qui sous-tend la version 4 du framework : le design des interfaces graphique doit être réalisé par des spécialistes. Nous transformerons ensuite leurs réalisations graphiques en interface utilisateur grâce à Flash Catalyst. À chacun sa spécialité, pas de mélange des genres !

Le **chapitre 11** se penche sur la création de composants personnalisés. Le **chapitre 12**, quant à lui, explore les possibilités qu'offre le framework en matière de gestion de données au sein d'une interface (saisie obligatoire, formatage…) et le **chapitre 13** étudie les méthodes d'accès aux données contenues dans une base de données.

Le **chapitre 14** présente l'interaction de Flex 4 avec PHP. Le **chapitre 15** aborde les relations de notre framework avec Java. Au chapitre 16, nous verrons comment interfacer notre framework avec le monde .NET.

Le **chapitre 17** détaille la notion d'application modulaire et la création de modules. Au **chapitre 18** nous verrons les méthodes qui permettent à une application Flex de communiquer avec la page web qui la contient. Le **chapitre 19** traite de l'internationalisation des applications.

Le **chapitre 20** explore les notions de débogage avant de passer, au **chapitre 21,** à l'étude des méthodes d'optimisation d'une application Flex. Le **chapitre 22** nous permettra d'aborder la notion de tests unitaires.

Enfin, le **chapitre 23** terminera la première partie avec le runtime Adobe AIR 2.

La seconde partie met en œuvre l'ensemble des concepts étudiés dans la première partie dans des cas pratiques. Nous détaillerons le développement d'une application e-commerce sur quatre chapitres mêlant application traditionnelle et runtime Adobe AIR 2.

Le **chapitre 24** définit le cadre du projet qui sera développé au cours des chapitres suivants. Le **chapitre 25** décrit le développement de la partie d'administration de l'application.

Le **chapitre 26** est consacré à la réalisation de son module principal, et le **chapitre 27** détaille le déploiement de l'application et apporte de nouvelles notions pratiques à l'ouvrage.

Le **chapitre 28** propose l'utilisation de l'API Google Maps.

Enfin, cet ouvrage se termine par l'**annexe 1** présentant les composants du framework, puis par l'**annexe 2** dédiée à l'interaction du framework avec un environnement JEE utilisant le framework Spring.

En pratique

Chaque chapitre de la seconde partie commence par une partie d'accompagnement à la réalisation et se termine par une rubrique « À vos claviers » : en situation, vous pourrez ainsi vérifier que vous avez bien assimilé les différents concepts énoncés tout au long de cet ouvrage.

À qui s'adresse ce livre ?

Cet ouvrage s'adresse autant :

- Aux connaisseurs de Flex 3, souhaitant aborder les nouvelles notions du framework dans sa version 4.
- Aux étudiants en programmation qui veulent apprendre à réaliser des applications clientes riches basées sur Flash.
- Aux développeurs aguerris qui souhaitent réaliser des interfaces graphiques performantes intégrables à des projets Java, PHP ou bien .NET existant.
- Aux chefs de projets qui désirent se faire une idée de la richesse technologique et des possibilités techniques de ce framework notamment au niveau de la productivité.
- Aux décideurs qui cherchent à rendre leurs applications plus attrayantes du point de vue utilisateur.

Remerciements

Écrire un ouvrage, nécessite de faire des concessions à tout point de vue durant une période assez longue, c'est donc tout naturellement que mes premiers remerciements vont à ma femme qui su faire preuve de compréhension pour les nombreux week-ends et soirées monopolisées, aux projets programmés puis déprogrammés pour raison de relecture, de modification ou de sortie d'ouvrage. C'est donc à toi Delphine que je dis un énorme merci et dédie cet ouvrage.

Je tiens également à remercier l'ensemble des personnes de l'équipe des éditions Eyrolles et plus particulièrement à Muriel Shan Sei Fan et Sandrine Paniel qui ont cru en ce projet et ont consacré leur temps aux taches de conseil et de relecture.

Je n'oublie pas le professeur Jean François Timsit de l'association Outcomera (*www.outcomerea.org*), qui m'a offert la possibilité de travailler sur des projets mettant en œuvre Flex ce qui m'a permis de me forger une solide expérience dans l'utilisation de ce framework et de vous la faire partager tout au long de cet ouvrage.

Merci à David Feugey pour son travail de relecture et de réflexion préliminaires.

Merci également à Fabien Nicollet (*http://www.flex-tutorial.fr/*) pour sa relecture, son regard extérieur sur l'ouvrage et pour sa contribution.

Merci à Ruben Swieringa de m'avoir autorisé à utiliser son composant FlexBook.

Suivi du lecteur

Naturellement, l'auteur se tient à la disposition des lecteurs, pour les aider dans les difficultés qu'ils pourraient rencontrer tout au long de cet ouvrage. Soulignons que, malgré tout le soin apporté à cet ouvrage, il est possible que certains problèmes apparaissent, indépendants de notre volonté, dus à une évolution du framework, du Flash Player, ou à tout autre élément annexe (oubli de saisie de code, mauvaise configuration…). Il serait donc dommage de sanctionner cet ouvrage sur le simple fait qu'un exemple ne fonctionne plus ou, comme cela nous a été signalé, qu'étant l'auteur de l'ouvrage, nous nous devions d'assurer le « service après vente ». Nous sommes tout à fait ouverts aux remarques et vous remercions d'avance pour tous vos retours. Nous sommes prêts à vous aider, dans la mesure où cela reste constructif pour les deux parties. Soulignons que l'expérience se forge par la pratique et que résoudre certains problème permet de la renforcer.

1

Pourquoi choisir Flex 4 ?

Ce premier chapitre fut rédigé sous forme de prise de position. En tant qu'architecte ou expert Flex, nous sommes souvent confrontés à la question « Pourquoi choisir Flex 4 ? »

Cette question vous pouvez également la rencontrer, si vous parlez de Flex autour de vous. Nous allons donc répondre à l'ensemble des interrogations afin que vous ou votre interlocuteur ait une vue d'ensemble du framework Flex 4 et de son utilité.

Les enjeux du développement des applications web

Commençons par une constatation : 75 % des nouvelles applications sont de type web, c'est-à-dire qu'elles ont pour support le navigateur Internet. Si nous analysons les raisons de cet engouement pour ce mode de développement, nous pouvons avancer trois points de réflexion :

- Une application web est disponible à tout moment et en tout point du globe. En effet, dès lors que l'utilisateur dispose d'une connexion Internet, il peut profiter de son application. Citons comme exemple la possibilité de consulter les informations de son compte bancaire, indépendamment de l'endroit où l'on se trouve.
- Une application web est compatible avec tous les systèmes d'exploitation. Ainsi, les problèmes d'incompatibilité rencontrés avec les clients lourds (applications s'exécutant sur le système d'exploitation) sur les environnements Windows, Linux ou Macintosh sont maintenant révolus. Cette compatibilité est rendue possible notamment par la mise en place de recommandations de développement (par le W3C, entre autres) et l'utilisation de langages normalisés, tels que le JavaScript ou le HTML, interprétables par l'ensemble des navigateurs web.

W3C (*World Wide Web Consortium*)

Le World Wide Web Consortium, fondé en 1994 par Tim Berners-Lee, n'a pas pour fonction première de créer des normes de développement mais plutôt des recommandations. Son domaine d'activité se concentre sur les langages orientés vers le développement d'applications Internet (HTML, XML…) afin d'assurer leur interopérabilité (*http://www.w3.org/*).

- Enfin, le dernier point est la légèreté du mode de distribution de ces applications. Si pour une application lourde nous devions posséder un CD-Rom d'installation, nous n'avons plus qu'à saisir une URL dans un navigateur. Notons également que ceci réduit d'autant le coût de distribution de l'application.

Néanmoins, ce phénomène induit tout de même un effet secondaire néfaste. Les applications disponibles depuis le Web ou les différents intranets d'entreprise sont aujourd'hui tellement nombreuses que se développe le syndrome d'applications « jetables ». Ainsi, comme il suffit de saisir une simple URL dans un navigateur pour exécuter une application, si celle-ci ne nous convient pas pour des raisons X ou Y, nous n'aurons qu'à lancer une requête dans un moteur de recherche pour en trouver une autre, similaire.

Dès lors, pour développer une application web viable et durable, les outils que nous pouvons désormais qualifier, de façon réductrice, de « simplistes » dédiés uniquement au développement et ne prenant pas en compte le processus complet de réalisation de l'application (de la conception au déploiement) ne sont plus de mise. Nous disposons maintenant de frameworks qui sont un ensemble d'outils réunis dans une seule application. Grâce à ces derniers, nous pouvons à présent déboguer une application web, disposer d'autocomplétion de code et designer des interfaces. L'utilisation d'un framework facilite la mise en place de la standardisation du code et de méthodes d'optimisations. Les applications développées sont ainsi homogènes, ce qui diminue le temps d'analyse et facilite l'apport de correctifs, augmentant donc la productivité.

Alternatives

Notons que nous avons parlés de framework au pluriel. En effet, il n'existe pas un mais des frameworks. Citons par exemple le framework ZendStudio pour le langage PHP, le Google Web Toolkit et bien entendu Flex 4.

Pourquoi choisir le framework Flex 4 plutôt qu'un autre ? Sans entrer dans le détail, car nous aurons l'occasion d'y revenir tout au long de ce chapitre, le framework Flex permet de créer des applications web basées sur la technologie Flash (employée dans les applications graphiques) procurant une qualité d'interface graphique très appréciée des utilisateurs. De plus, sa prise en main est facilitée par l'outil de développement Flash Builder basé sur le célèbre IDE Eclipse, très largement utilisé par les développeurs de nouvelles technologies.

Cette facilité de prise en main séduira donc les développeurs juniors et seniors qui verront dans ce framework la possibilité de créer rapidement des applications web alliant esthétique et technique. Les jeunes développeurs seront sans doute intéressés par le coté « nouveau » de cette technologie, source de motivation et moteur de créativité, les plus aguerris y verront le moyen de créer des interfaces graphiques très performantes tout en minimisant le temps de développement. Notre framework ravira également les décideurs (chef de projets ou d'entreprise), car il permet, d'une part, de produire des applications esthétiques appréciées des utilisateurs et, d'autre part, de réduire les délais de développement grâce aux avantages énoncés précédemment.

Dans sa version 4, le framework ravira sans doute les infographistes quelque peu délaissés dans les précédentes versions. En effet, il est désormais possible d'importer les créations réalisées dans Adobe Photoshop, Adobe Illustrator et de les transformer en interface graphique, via bien entendu un outil d'interfaçage portant le nom de Flash Catalyst.

Import des créations réalisées à l'aide d'autres outils graphiques

À notre connaissance, il n'existe malheureusement pas de possibilité d'importer des créations faites à partir de Gimp ou bien encore d'Inkscape.

Nous sommes sûrs qu'à la suite de ces quelques lignes, vous vous posez certaines questions. C'est pourquoi nous allons poursuivre ce chapitre en tentant d'y répondre.

Flex en questions

En tant que spécialiste, lorsque vous évoquez le framework avec des chefs de projet, développeurs ou tout autre acteur d'un projet de développement informatique, plusieurs questions reviennent immanquablement :

- Est-ce une technologie récente ?
- Qu'est-ce que le framework Flex ?
- Comment fonctionne-t-il ?
- De quoi est-il composé ?
- S'agit-il d'un logiciel open source ou d'une solution propriétaire ?

Voyons comment répondre à ces questions.

Est-ce une technologie récente ?

Flex vit le jour en mars 2004. Initié par Macromedia, repris en 2006 par Adobe, le projet a connu une évolution rapide comme le montre le tableau 1-1 qui récapitule les mises en production des différentes versions.

Tableau 1-1 – Historique des versions du framework Flex

Date	Événement
Mars 2004	Sortie de Flex version 1.0
Octobre 2004	Flex 1.5
Octobre 2005	Flex 2.0 alpha
Février 2006	Flex 2.0 bêta 1
Mars 2006	Flex 2.0 bêta 2
Mai 2006	Flex 2.0 bêta 3
Janvier 2007	Flex 2.01
Février 2008	Flex 3
Avril 2010	Flex 4
Octobre 2010	Flex Hero (Version bêta) chargée de porter la technologie sur le monde mobile

Si l'on considère cette chronologie, on constate la mise en production d'une nouvelle version par an. On peut en déduire que cette technologie est chère à Adobe, qui investit largement dans ce projet.

Sans chercher à décrire les différentes versions du framework, il est tout de même important de noter qu'à l'origine, Flex a été créé pour le développement de la couche graphique d'une application (partie front end) JEE. Ceci nécessitait la présence d'un serveur Flex qui a totalement disparu dès la version 2. La version 3 apporte également son lot de nouveautés comme l'ouverture vers le monde libre de certaines parties du SDK, ou l'apparition de nouveaux composants. Avril 2010 est marqué par la sortie de la version 4, que l'on peut considérer comme révolutionnaire pour le framework Flex, la volonté principale d'Adobe pour cette version étant de dissocier la partie graphique de la partie développement.

Qu'est-ce que Flex ?

Flex est un framework qui sert à la création d'interfaces client riches (*front end* de l'application) basé sur l'usage de la technologie Adobe Flash. En effet, si celle-ci apporte une grande souplesse pour la création d'interfaces graphiques riches, elle est souvent perçue par les non-initiés comme complexe.

De plus, pour bon nombre de techniciens spécialisés dans le développement d'application, la conception et la réalisation de l'interface graphique de l'application semblent souvent secondaires : ils préfèrent se focaliser complètement sur la fonctionnalité à développer plutôt que sur la présentation du résultat. Le framework Flex a ainsi pour objectif de faciliter cette partie du développement, en supprimant la notion de time line.

Time line

Bien connu des développeurs Flash, le time line sert à définir des actions sur les composants selon un axe temporel. Ce mode de développement est réservé aux spécialistes de Flash et diffère du mode de développement standard basé sur la disposition de composants sur une page web.

En effet, pour créer du contenu lisible par Flash Player, il n'y avait auparavant que peu d'alternatives. Les développeurs devaient utiliser le logiciel Adobe Flash Professional dédié à la création d'animations. Une application étant constituée de différents écrans, les développeurs ont donc essayé de mettre chaque écran sur une image de l'axe temporel. Pour naviguer parmi les écrans, il suffisait ensuite de parcourir l'axe temporel. Même si cette solution pouvait suffire pour des applications simples (formulaire de contact), nous atteignions rapidement les limites de ce type de développement pour des applications dédiées au monde de l'entreprise :

- interface conçue pour le dessin vectoriel et non pour la programmation ;
- organisation des classes rapidement être anarchique car aucune navigation dans les sources de l'application n'est proposée ;
- programmation liée à la navigation dans l'axe temporelle pouvant rapidement devenir ingérable ;
- possibilités de débogage restreintes ;
- aide à la programmation quasi inexistante ;
- bibliothèques de composants assez pauvres et rigides.

Nous étions donc loin des environnements de développement auxquels les développeurs étaient habitués. Adobe (et Macromedia en tête) a donc conçu le framework Flex et le logiciel Flash Builder (anciennement Flex Builder). À partir de sa version 2, Flex a pour but de répondre à ces problématiques, en proposant un ensemble de classes ActionScript 3 axées développement applicatif.

Quel est son fonctionnement ?

Une application Flex n'est autre qu'un fichier portant l'extension `.swf` (Shockwave Flash) issu de la compilation de fichiers MXML et ActionScript, insérés dans une page web.

Les fichiers MXML (Macromedia XML) servent à la description des interfaces graphiques, en définissant la position des composants et leurs propriétés. Quant aux fichiers ActionScript 3, ils ont la charge de la logique applicative : gestion d'événements, écriture de procédure, fonctions…

Sur quelles technologies repose-t-il ?

Voyons à présent le socle sur lequel repose Flex 4 et les technologies associées. Le framework Flex 4 s'articule autour des quatre éléments suivants :

- Le Flash Player 10 – Équipant près de 97 % des ordinateurs (Windows, Mac OS et Linux), le Flash Player assure la compatibilité des applications développées en Flash sur l'ensemble des environnements et navigateurs web.
- Le Flex 4 SDK – Ouvert au monde open source, le SDK (*Software Development Kit*) permet de créer des applications Flex gratuitement à l'aide des compilateurs et du débogueur qu'il intègre.
- Flash Builder 4 – Environnement de développement basé sur le célèbre IDE (*Integrated Development Environment)* Eclipse, il propose un ensemble de fonctionnalités permettant de réaliser facilement des interfaces graphiques conviviales. Depuis sa version 3, il offre également des fonctions avancées de débogage comme le profiling d'application (qui intervient notamment dans la description de la charge mémoire utilisée par l'application). Cet outil est disponible en version standard et professionnelle mais nécessitera cependant un investissement (environ 200 € pour la version Standard et 600 € pour la version Premium). À noter qu'il existe une version gratuite, à utilisation non commerciale pour les étudiants.
- BlazeDS s'adresse à celles et ceux qui souhaitent créer des applications Flex communicant avec un *back-end* JEE. Il s'agit d'un sous-ensemble de projets open source basé sur les briques principales de son homologue LiveCycle Data Services d'Adobe, qui est payant.

Front end et back end d'une application

Le back end est chargé de contenir la logique métier de l'application (règles de gestion, traitements des données...). Le front end, quand à lui, gère la partie interface homme machine (IHM) de l'application.

Choisir la méthode open source ou propriétaire ?

Voilà la grande question que tout le monde est en droit de se poser. Comportant l'ensemble des bibliothèques et compilateurs nécessaires au développement, certaines parties du Flex SDK sont à présent sous licence open source. Néanmoins, nous verrons, lors du chapitre 3, que son utilisation n'est pas des plus aisées. Cependant, pour faciliter le travail du développeur, Adobe propose l'environnement de développement propriétaire Flash Builder. Vous disposez ainsi d'un environnement de développement, de test et de déploiement efficace.

Bien entendu, cet environnement de développement a un prix : il faut débourser environ 600 € pour acquérir une licence Premium. Mais, comme nous venons de l'indiquer, le Flash Builder n'est qu'un « plus » qui facilite l'utilisation du framework. Il est donc tout à fait possible de développer vos applications avec Flex sans débourser un sou, soit en utilisant un simple éditeur de texte et un compilateur en ligne de commande, soit en passant par un outil open source tel que Flash Develop.

Flex : quelles différences avec Ajax ?

Comme nous venons de l'indiquer, si vous souhaitez recourir à l'IDE Flash Builder, il faudra bourse délier. Bien entendu, cet investissement financier sera d'autant plus important que les développeurs amenés à l'utiliser seront nombreux. Vous êtes donc en droit de vous demander pourquoi effectuer une telle dépense, alors qu'en optant pour un framework Ajax, vous pourriez tout à fait réaliser des RIA (*Rich Internet Application*), et ce, de façon totalement gratuite tout en profitant des forces des projets open source.

Qu'est-ce qu'Ajax ?

Ajax (*Asynchronous Javascript and XML*, XML et JavaScript asynchrone) est une méthode de développement d'application web combinant différentes technologies telles que HTML, Javascript, XML ou encore le protocole HTTP. On l'utilise notamment pour créer des interfaces interactives. Pour en résumer le principe, Ajax permet d'interagir avec un serveur web sans devoir recharger l'intégralité de la page. De nombreux frameworks Ajax – comme Openlaszlo *(http://www.openlaszlo.org/)* – mettent à profit cette méthode de développement pour la réalisation d'interfaces client riches.

Le principal argument que nous pouvons avancer est le suivant : Ajax se base sur le langage JavaScript dont l'interprétation peut varier d'un navigateur à l'autre. Ceci implique donc l'adaptation du code pour chaque navigateur. Une application Flash reposant exclusivement sur le Flash Player, son comportement sera identique quelle que soit la plate-forme ou l'environnement dans lequel elle est exécutée.

Un second argument peut également être avancé : il est plus facile et rapide de développer une interface graphique à fonctionnalités équivalentes avec le framework Flex qu'avec la technologie Ajax. Comme nous l'avons évoqué au début de ce chapitre, nous sommes dans une société de consommation d'applications web, où la durée de vie de celles-ci est limitée et où l'aspect esthétique est devenu primordial. Par conséquent, si nous souhaitons réaliser une application ayant une interface attrayante tout en réduisant son coût de développement, nous avons tout intérêt à nous diriger vers l'usage du framework Flex.

Le tableau 1-2 établit une comparaison entre l'utilisation du framework Flex et la méthode de développement Ajax. Bien entendu, il s'agit là moins de comparer le framework Flex à l'ensemble des frameworks Ajax (tels que OpenLazlo ou bien encore jQuery), que de mesurer le résultat obtenu.

Tableau 1-2 – Comparaison entre une application développée avec Flex et avec Ajax

Critère	Flex	Ajax
Nécessité d'un plug-in sur le poste client pour l'exécution de l'application.	Oui.	Non, même s'il nécessite l'activation de JavaScript sur le poste client.
Coût du développement de l'application.	Gratuit pour l'utilisation du SDK. Payant pour l'utilisation de l'IDE.	Gratuit.

Tableau 1-2 – Comparaison entre une application développée avec Flex et avec Ajax *(suite)*

Critère	Flex	Ajax
Portabilité de l'application.	Comportement identique sur l'ensemble des navigateurs.	Le comportement de l'application peut différer selon le navigateur web (hors utilisation d'un framework).
Insertion de contenu multimédia (vidéo, son…) dans l'application à l'aide de fonctionnalités pré-existantes.	De par son lien avec Flash, les fonctionnalités multimédias sont incluses dans le framework.	Les fonctionnalités multimédias ne sont pas incluses.
Composants graphiques prêts à l'emploi.	Flex intègre de nombreux composants graphiques (menus accordéon…).	Ajax ne comporte pas de composants graphiques, c'est au développeur de les créer. Certains frameworks proposent cependant des composants prêts à l'emploi.
Nombre de langages utilisés.	Deux : MXML et ActionScript.	Deux : JavaScript et XML.

Comme vous le constatez, la frontière séparant le framework Flex de la méthode de développement Ajax est très mince. Mais si nous faisons le bilan de cette comparaison, il est évident que l'atout majeur de Flex est sa faculté à rendre les applications compatibles avec l'ensemble des navigateurs, grâce à Flash tout en proposant un panel de composants prêts à l'emploi. C'est également grâce à Flash et ses possibilités multimédias et de mise en forme des composants que Flex se démarque d'Ajax.

Néanmoins, le principal inconvénient des applications Flex est qu'elles sont hermétiques à leur environnement : au sein d'une page web, la communication avec les éléments HTML, JavaScript, etc. est inexistante, à moins de passer par un pont de communication réalisé en JavaScript et portant le nom de Flex-Ajax Bridge. Mais nous aurons l'occasion de revenir sur cette notion dans le chapitre 18. Plutôt que de chercher à se démarquer de la solution Ajax, les développeurs du framework Flex ont pris conscience de ses faiblesses et ont opté pour une alliance de technologies.

Les applications Flex sont souvent plus rapides que leur équivalent réalisé en Ajax. En effet, le code contenu dans le fichier `.swf` est compilé, alors qu'en technologie Ajax, on utilise du code JavaScript, qui est lui interprété.

À l'heure où l'on parle de la « révolution HMTL 5 », rappelons que l'utilisation d'Internet Explorer (souvent dans sa version 6) est une réalité dans le monde de l'entreprise, souvent réticente au changement pour des raisons économiques. La compatibilité Internet Explorer 6/7 est donc souvent demandée dans les cahiers des charges des applications web.

Afin d'assurer cette compatibilité, il n'y a qu'une chose à faire : il faut réaliser des tests manuellement sur l'ensemble des navigateurs. Ces tests engendrant un coût en temps de travail et par conséquent également un coût financier. Flash Player vous assure quant à lui, une cohérence sur tous les navigateurs et systèmes d'exploitation du marché, car son installation est compatible avec ces derniers.

Flex et ses concurrents

Encore restreint il y a quelques années, le marché de la RIA, se développe peu à peu :

- La technologie XUL (*XML-based User interface Language*) initiée par Mozilla consiste, tout comme le framework Flex, à décrire les interfaces graphiques à l'aide de XML.
- Le framework OpenLazlo se rapproche du framework Flex par la génération d'interfaces graphiques basées sur la technologie Flash (*http://www.openlaszlo.org/).* Ce framework a connu ses heures de gloire lorsque Flex n'était pas encore abouti, il y a quelques années. OpenLazlo permettait ainsi de générer du contenu SWF ou une plate-forme Ajax. Mais comme le but est de créer aussi bien une application Ajax qu'une application SWF, OpenLazlo a dû se contenter du « dénominateur commun », c'est-à-dire l'Ajax. Impossible donc de faire des applications SWF avancées, car OpenLazlo se trouve limité par rapport au framework Flex. Il reste cependant une alternative si vous souhaitez développer une application Ajax.
- Entièrement basé sur la technologie Java, le framework Google Web Toolkit (GWT), permet de créer également des interfaces client riches Ajax assez intéressantes en termes de fonctionnalités. Le développement s'effectue par programmation de code Java, et le code est ensuite « converti » en code HTML et JavaScript pour faire une application Ajax. Un des exemples de l'utilisation de cette technologie est l'application de géolocalisation Google Maps (*http://maps.google.fr/*).

  GWT permet de s'affranchir de certaines contraintes du développement Ajax, dont, notamment, la compatibilité entre les navigateurs.

Néanmoins, le principal concurrent de Flex 4 est la plate-forme de développement Silverlight de Microsoft. Elle offre un environnement de développement semblable en termes de fonctionnalités, tout en se différenciant par l'utilisation du langage XAML pour la description des interfaces et de JavaScript pour la partie logique applicative. L'exécution de ces applications diffère également par l'utilisation d'un plug-in à installer sur le navigateur.

Tentons à présent de réaliser un comparatif de ces deux frameworks.

Flex et Silverlight

Comme nous venons de l'évoquer, Silverlight est le concurrent majeur de Flex. Cependant, il ne s'agit pas ici de dénigrer la technologie concurrente, mais bien de vous apporter des éléments de comparaisons vous permettant de faire un choix pour l'une au l'autre technologie si le besoin venait à se présenter.

Silverlight en quelques mots

Tableau 1- 3 – Environnement d'exécution

	Framewok Flex	Framework Silverlight
Linux	Oui	Oui grâce au plug-in Moonlight
Mac OS	Oui	Oui
Windows Performance	Oui	Oui
Charge processeur	Peut élevée	Assez élevée
Multithreading	Impossible	Possible

Tableau 1-4 – Langage et développement

	Framewok Flex	Framework Silverlight
Description IHM	MXML	XAML
Logique métier	ActionScript 3	C Sharp (C#),
Possibilité de développer à l'aide d'outil open source	Oui	Oui

Tableau 1-5 – Multimédia

	Framework Flex grâce	Framework Silverlight
Support de fichier images Gif / BMP	Oui	Non
Utilisation de la webcam et du microphone	Oui	Oui
Lecture de fichier Vidéo	Capable de lire l'ensemble des fichier (avi,mov, mpeg)	Limité à la lecture des fichiers WMV

Tableau 1-6 – Conception

	Framework Flex	Framework Silverlight
Différentiation entre le design et le développement	Oui grâce à Flash Catalyst	Oui grâce à Expression Blend
Nombre de composants	Très fournies	Possède des composants de bases

Tableau 1-7 – Mobilité

	Framework Flex	Framework Silverlight
Portabilité sur le monde mobile	Flash ne semble pas encore très enclin à notion de portabilité	Utilisable sur les téléphone utilisant le système d'exploitation windows mobile

Tableau 1-8 – Communauté

	Framework Flex	Framework Silverlight
Existence d'une communauté	Forte communauté	Tend à se développer

Développé par Microsoft, Silverlight est un framework permettant de créer des applications RIA. Pour exécuter une application développée avec ce framework et la visualiser dans son navigateur, il convient d'installer sur son environnement de travail le *plug-in* Silverlight, tout comme nous le faisons avec le Flash Player.

Silverlight est actuellement commercialisé dans sa version 4, parue courant avril 2010. Dans cette version, notons quelques nouveautés telles que la prise en charge du micro et de la webcam, la multidiffusion de contenus multimédias, le mode *out of browser* permettant d'exécuter une application hors du navigateur.

Concernant la portabilité des applications, le plug-in Silverlight est exécutable sur les différents navigateurs, dont bien évidemment Internet Explorer et Google Chrome, mais également Fire-Fox et Safari. Le plug-in étant dédié à être exécuté sur un environnement Windows, il est tout même possible de visualiser une application Silverlight sur dans environnement Linux grâce au plug-in Moonlight développé par la société Novell.

À la lecture de ces quelques lignes, nous pouvons déjà faire de nombreux parallèles entre les deux plates-formes.

Un début de comparaison

Réaliser un tableau comparatif n'est pas tâche si aisée qu'il y paraît. En effet, si l'on observe de nombreuses similitudes entre les deux plates-formes, nous l'avons tout de même dressé, afin de vous fournir une base de comparaison.

> **Précision**
> Silverlight est avant tout destiné à un environnement Windows. Si votre plate-forme est orientée .Net, n'hésitez pas à vous tourner vers Silverlight qui vous offrira une compatibilité exemplaire.

Vers quelle solution faut-il se diriger ? La première chose à prendre en compte est la politique informatique de l'entreprise. Si la norme est de limiter l'utilisation de projets libres ou si

l'entreprise est en partenariat avec Microsoft, vous vous orienterez naturellement vers Silverlight, sinon Flex semble tout indiqué. Le second argument en faveur de la technologie Flex est qu'elle utilise le Flash Player, compatible avec toutes les plates-formes, pour exécuter l'application, alors qu'il faudra télécharger un plug-in propre à Microsoft, qui ne semble pas encore être officiellement compatible pour la plate-forme Linux.

Soulignons que pour Silverlight, il faut utiliser l'environnement de développement Visual Studio dont le prix avoisine les 1 000 €. Ce point peut éventuellement être compté en la défaveur de Silverlight et avantager Flex. En effet, là où nous avons besoin d'un module applicatif, nous sommes obligés d'acquérir l'ensemble de l'environnement de développement.

Il faut également prendre en compte les compétences des développeurs de l'équipe de développement ou les préférences pour les langages Microsoft .Net, qui font tendre à une utilisation naturelle deSilverlight.

Bibliographie

Gérard Leblanc, *Silverlight 2,* éd. Eyrolles, 2008.

Les limites de Flex

Comme toute solution informatique, Flex a ses propres limites, parfois décriées par les développeurs Ajax ou Flash désireux de remettre en question jusqu'à l'utilité de ce framework. Nous allons donc essayer d'apporter quelques éléments de réponse afin d'éclaircir tout ceci.

« L'exécution d'une application Flex nécessite la présence du Flash Player sur le poste client ! »

Cette condition sine qua non peut en effet être un frein à l'utilisation des applications Flex. Néanmoins, le lecteur Flash assure une stabilité d'exécution des applications sur l'ensemble des machines à l'inverse d'Ajax dont le comportement peut différer d'un navigateur à un autre. Il s'agit donc d'un mal pour un bien. Sachant que la plupart des machines sont équipées du Flash Player… est-ce une raison valable pour en dénigrer l'utilisation ?

« Le framework Flex ne propose pas d'outils dédiés aux graphisme ! »

Comme nous l'avons évoqué précédemment, Flex est basé sur Flash, ce qui simplifie beaucoup la réalisation d'interfaces graphiques riches. Il n'est nullement question de créer des animations comme nous pourrions le faire avec Flash Professional ! Pour pallier ce problème, Flex contient un composant, nommé SWFLoader, permettant d'importer des animations Flash.

« Le Flash Player n'est pas compatible avec les processeurs 64 bits ! »

En effet, à l'heure ou nous écrivons ces lignes, le Flash Player n'est actuellement pas supporté par les machines équipées d'un processeur 64 bits (sauf sous environnement Linux). Bien que cela concerne actuellement peu de machines, la solution préconisée par Adobe consiste

à exécuter un navigateur 32 bits compatible sur les machines 64 bits pouvant alors afficher les applications Flash.

« Dans une application Flash, il est impossible d'utiliser les boutons Précédent et Suivant du navigateur ! »

Ceci est tout à fait possible grâce à la notion de deep linking. En effet, elle permet de naviguer à l'intérieur d'une application à l'aide des boutons Suivant et Précédent du navigateur, comme il est possible de le faire dans toute application web. Nous développerons cette fonctionnalité dans le chapitre 18.

« Une application Flash est difficile à référencer sur les moteurs de recherche ! »

Certains moteurs de recherche comme Google sont dorénavant capables d'analyser le contenu d'un fichier SWF. Le référencement d'une animation Flash est donc possible. Dans tous les cas, n'oublions pas qu'une application Flash est généralement contenue dans une page HTML. Ceci permet alors d'y stocker l'ensemble des informations de référencement même si, à l'heure où nous écrivons ces lignes, les informations indexées ne sont pas aussi pertinentes que celles d'un site HTML.

Bien entendu, nous n'avons pas recensé l'ensemble des remarques soulevées par les « pro-Ajax » ou « pro-Flash ». Il est important de retenir qu'Adobe met tout en œuvre pour pallier l'ensemble des problèmes qui pourraient mettre son produit en difficulté. Ceci nous amène donc à dire que le projet Flex 4 a atteint un degré de maturité non négligeable et qu'il s'agit d'une valeur sûre.

« L'utilisation d'applications Flash n'est pas destinée au monde mobile »

Vous n'êtes pas sans savoir qu'une guerre ouverte a été déclarée entre Apple et Adobe concernant le portage de Flash sur l'iPhone. Cette technologie est cependant tout à fait portable sur le monde des Smartphones. De plus, avec la naissance prochaine du framework Flex Hero (*http://labs.adobe.com/technologies/flex/mobile/faq.html#desktop*) tout porte à croire que l'objectif vers la mobilité sera sans doute atteint très prochainement.

Pourquoi choisir Flex ?

Il ne s'agit pas de reprendre les arguments commerciaux édictés par Adobe mais de vous faire part de nos constatations suite à notre expérience découlant de l'utilisation de ce framework à travers des projets de différentes natures.

Si nous devions vous convaincre d'utiliser Flex plutôt qu'une autre technologie, voici ce que nous vous avancerions :

- Flex apporte une valeur ajoutée indéniable au niveau commercial par la richesse des interfaces développées (ce qui est beau est vendeur). Pour vous en rendre compte, il vous suffit de visiter la page d'exemple du site d'Adobe : *http://www.adobe.com/devnet/flex/?tab:samples=1.*
- Flash Builder est un produit mature, vous permettant de réaliser des développements de qualité professionnelle. Il est aussi idéal pour le travail en équipe grâce aux extensions Eclipse disponibles.

- Il s'adapte aux besoins en offrant la possibilité de créer ses propres composants. Nous nous pencherons sur ce point au chapitre 11.
- Par expérience, l'utilisation du framework Flex va diviser les temps de développements car le développeur pourra se concentrer sur les aspects métier, et non sur la compatibilité de son travail avec l'environnement où sera exécuté l'application. Les applications développées se comportent de manière identique quelles que soient les plates-formes et les conditions d'utilisation.
- Flex permet de réaliser des applications bureautiques riches (ou RDA, *Rich Desktop Application*) grâce à la machine virtuelle Adobe AIR (ce que nous verrons au chapitre 23).

Qu'apporte Flex 4 ?

Cet ouvrage traitant de l'ensemble des nouveautés Flex 4 par rapport à la version précédente, il serait prématuré en début d'ouvrage d'entrer dans le détail de chacune d'entre elles. Notons dans un premier temps un changement de philosophie du produit résumé dans ces trois points :

- *Design in Mind :* améliorer la communication entre les développeurs et les infographistes en permettant à chacun de s'exprimer selon leur domaine d'expertise. Le développeur développe et l'infographiste créé, le travail de chacun est ensuite mis en commun.
- *Developer Productivity :* mise en place de nouvelles notions améliorant la qualité et l'efficacité du développement tel que le databinding bi-directionnel.

> **Notion de databinding**
> Le databinding consiste, en résumé, à la liaison de variables. Lorsque l'une est mise à jour, la seconde l'est également. Dans Flex 3, seule la variable destination était mise à jour lorsque nous modifions la variable source. Dans le framework Flex 4, la réciprocité est maintenant éprouvée.

- *Framework Evolution :* le framework Flex 4 doit prendre en compte l'ensemble des évolutions du Flash Player telles que la 3D.

Nous pouvons également évoquer ces nouveautés par le biais de cette liste non exhaustive :

- présence de thèmes graphiques Flex ;
- création de nouveaux espaces de noms, permettant de dissocier la partie composant de la partie graphique ;
- existence de nouveaux composants Spark ;
- nouvelle gestion des états ;
- nouvelle gestion de mise en forme de composant (*layout*).

Pour utilisateurs aguerris au framework Flex 3, nous avons présentés les équivalences entre les composants Flex 3 et ceux maintenant utilisés par Flex 4. Cependant, il existe des composants Flex 3 non portés en Flex 4. Mais rassurez-vous, la compatibilité est assurée. Le portage des composants Flex 3 vers une application Flex 4 est possible.

Tableau 1-9 – Équivalence des composants Flex 3 et Flex 4 (source Adobe)

Composants Flex 3	Composants Spark Flex 4
mx.controls.Button	spark.components.Button
mx.controls.ButtonBar	spark.components.ButtonBar
mx.controls.CheckBox	spark.components.CheckBox
mx.controls.ComboBox	spark.components.DropDownList (w/o editability)
mx.controls.HorizontalList	spark.components.List (with a HorizontalLayout)
mx.controls.HRule	spark.primitives.Line
mx.controls.HScrollBar	spark.components.HScrollBar
mx.controls.HSlider	spark.components.HSlider
mx.controls.Image	spark.primitives.BitmapImage (w/o support for externalimages)
mx.controls.LinkBar	spark.components.ButtonBar (with a custom skin)
mx.controls.LinkButton	spark.components.Button (with a custom skin)
mx.controls.List	spark.components.List
mx.controls.NumericStepper	spark.components.NumericStepper
mx.controls.RadioButton	spark.components.RadioButton
mx.controls.RadioButtonGroup	spark.components.RadioButtonGroup
mx.controls.TextArea	spark.components.TextArea
mx.controls.TabBar	spark.components.TabBar
mx.controls.TextInput	spark.components.TextInput
mx.controls.TileList	spark.components.List (with a TileLayout)
mx.controls.ToggleButtonBar	spark.components.ButtonBar
mx.controls.VideoDisplay	spark.components.VideoPlayer
mx.controls.VRule	spark.primitives.Line
mx.controls.VScrollBar	spark.components.VScrollBar
mx.controls.VSlider	spark.components.VSlider
mx.core.Application	spark.components.Application
mx.core.Window	spark.components.Window
mx.core.WindowedApplication	spark.components.WindowedApplication
mx.containers.ApplicationControlBar	spark.components.Application (with the controlBarContent)
mx.containers.Canvas	spark.components.Group

Tableau 1-9 – Équivalence des composants Flex 3 et Flex 4 (source Adobe) *(suite)*

Composants Flex 3	Composants Spark Flex 4
mx.containers.ControlBar	spark.components.Panel (with the controlBar-Content property)
mx.containers.HBox	spark.components.HGroup
mx.containers.Panel	spark.components.Panel
mx.containers.Tile	spark.components.Group (with a TileLayout)
mx.containers.VBox	spark.components.Vgroup

Les composants Flex 3 n'ayant pas d'équivalent en Flex 4 sont les suivants (source Adobe) :

- mx.controls.Alert
- mx.controls.ColorPicker
- mx.controls.DataGrid
- mx.controls.DateChooser
- mx.controls.DateField
- mx.controls.Menu
- mx.controls.MenuBar
- mx.controls.PopUpButton
- mx.controls.PopUpMenuButton
- mx.controls.ProgressBar
- mx.controls.RichTextEditor
- mx.controls.Tree
- mx.containers.Accordion
- mx.containers.DividedBox
- mx.containers.Form
- mx.containers.Grid
- mx.containers.TabNavigator
- mx.containers.TitleWindow
- mx.containers.ViewStack

Faut-il migrer les applications Flex 3 vers Flex 4 ?

Au vu de ces changements, nous sommes en droit de nous demander s'il est judicieux de migrer une application Flex 3 vers Flex 4. Ne risquons-nous pas de perdre des informations, de devoir réaliser à nouveau le design de nos applications ?

Selon nous, (notre avis n'engage que nous), il pourrait en effet être intéressant de migrer une application Flex 3 vers Flex 4, afin de profiter de l'ensemble des nouveautés. Cependant, Flex 4

est encore assez jeune, et la migration n'est réellement envisageable que si vous avez vraiment besoin d'un élément ou d'une fonctionnalité non existante dans le framework 3.

Il est en effet sage de préconiser de maintenir une application Flex 3 fonctionnelle dans cette version et de créer de nouveaux projets à l'aide du framework 4, afin de s'approprier les nouveautés. Après un temps d'adaptation, l'application peut ensuite être migrée ou récrite si cette dernière a atteint la fin de son cycle de vie.

Cependant n'oublions pas que petit à petit, les anciennes versions du framework vont être abandonnées au profit de la nouvelle version. Mais au regard de se qui se passe en entreprise, le framework 3, voire 2, est encore largement utilisé.

Les présentations étant à présent faites, nous pouvons commencer à détailler l'utilisation et le fonctionnement du framework avec le chapitre suivant qui décrit le framework Flex 4.

2

À la découverte du framework Flex 4

Dans ce chapitre – peut être l'un des plus importants de cet ouvrage – nous allons disposer les premières briques qui vont nous permettre de comprendre les fondements essentiels du framework Flex. Ces fondements sont la base de l'utilisation du framework car ils permettent d'en connaître le fonctionnement, les possibilités mais également les limites.

Rôle du front end

La notion de découpage d'une application en trois couches distinctes est bien connue des développeurs de nouvelles technologies (JEE, PHP…). Néanmoins, si vous êtes néophyte en la matière et que cela ne vous évoque rien, ne vous inquiétez pas, nous allons tout de suite remédier à cela.

Architecture n-tiers

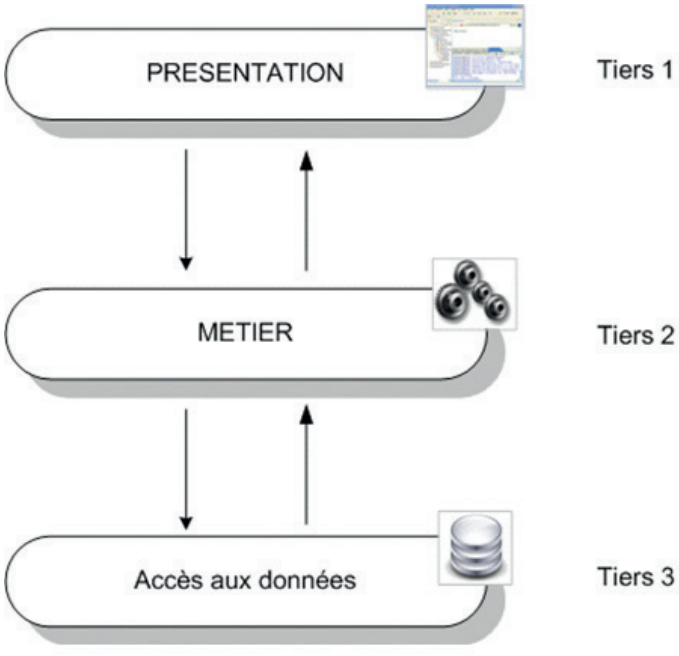
Une application peut être divisée en plusieurs couches ou tiers applicatifs, nous parlons alors d'une architecture n-tiers. Ainsi, si nous subdivisons une application en trois couches distinctes, nous obtenons une architecture 3-tiers :

- La couche Présentation est chargée, comme son nom l'indique, de présenter les données (Interface Homme-Machine ou IHM).
- La couche Métier implémente la logique métier (règles de gestion, etc.).
- La couche d'accès aux données réalise la tâche de persistance des données (stockage) via l'utilisation d'une base de données, par exemple.

Chaque couche propose des services qu'elle met à disposition de celle qui lui est directement supérieure. Ainsi, la couche la plus élevée ne peut communiquer avec la couche la plus basse de l'architecture qu'au travers de la couche intermédiaire, comme l'illustre la figure 2-1.

Figure 2-1

Architecture 3-tiers



Positionnement d'une application Flex

Au cours du chapitre précédent, nous avons défini le rôle du framework Flex comme étant un outil de création d'interfaces clientes riches (RIA). De ce fait, le positionnement des applications réalisées à l'aide de ce framework dans une architecture de type 3-tiers, se fera naturellement au niveau de la couche Présentation (tiers 1).

Ce positionnement en haut de l'architecture implique donc une notion que nous devrons toujours prendre en compte lors de la création d'une nouvelle application : une application réalisée à l'aide du framework Flex ne peut pas accéder directement aux informations contenues dans une base de données, comme c'est le cas pour les applications Java ou PHP.

Par conséquent, si nous souhaitons développer une application utilisant des informations contenues dans une base de données, nous devrons implanter une couche Métier intermédiaire offrant un certain nombre de services écrits dans un langage serveur (PHP, Java, ColdFusion…). Ces services seront capables de se connecter à la base de données et d'y effectuer des requêtes pour

ensuite renvoyer les résultats à notre application. L'appel de ces services à partir de l'application se fera via des composants spécifiques du framework Flex ou via des passerelles de communication telles que Zend_AMF que nous verrons dans le chapitre 14.

Dans cette section, nous avons donc vu un élément essentiel du framework Flex dont l'impact n'est pas négligeable si vous souhaitez créer des applications accédant à des données : la mise en place impérative d'une couche Métier écrite dans un langage serveur. Néanmoins, toutes les applications Flex n'utilisent pas forcément des données, c'est le cas par exemple d'une calculatrice ou d'un simulateur d'emprunt. Qui plus est, il nous est également possible de stocker des informations de façon temporaire au sein de notre application à l'aide d'un modèle de données qui, là encore, est bien connu des développeurs Java.

L'application et le modèle-vue-contrôleur (MVC)

Maintenant que nous savons que le framework Flex a pour but premier la création de la partie graphique d'une application, nous pouvons entrer dans le détail de la couche graphique en abordant la notion de modèle-vue-contrôleur (MVC). Cette couche peut elle aussi se découper en trois éléments :

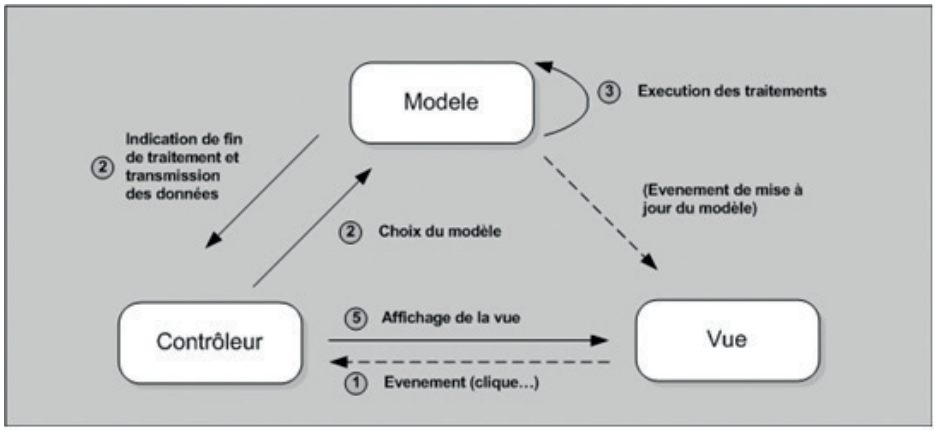
- Le modèle représente les données et les règles métier via un ensemble de classes et de méthodes associées. Il permet également l'accès aux services décrits dans la couche Métier de l'architecture 3-tiers. Enfin, c'est lui qui se chargera de prévenir la vue d'un changement de données afin que celle-ci se mette à jour.
- La vue va afficher les données du modèle et permettre l'interaction avec l'utilisateur, c'est l'interface homme-machine (IHM) de votre application. Notons, qu'il est tout à fait possible d'avoir plusieurs vues affichant les données d'un même modèle.
- Le contrôleur, quant à lui, va jouer le rôle d'interface entre la ou les vues et le modèle. Il récupère les actions de l'utilisateur pour ensuite les rediriger vers le modèle sous forme d'événements et mettre à jour la vue appropriée.
- Si vous n'avez jamais été confronté à la notion de MVC, ce qui vient d'être énoncé doit vous paraître assez obscur. Pour vous aider à mieux comprendre cette notion, nous allons l'illustrer à l'aide d'un petit exemple. Considérons l'action sur un bouton permettant de lister l'ensemble des données contenues dans une table et de les afficher dans une grille de résultats (figure 2-2).

  1. L'utilisateur clique sur le bouton Afficher les données. Cette action déclenche alors un événement que la vue se charge de faire suivre au contrôleur.
  2. Le contrôleur approprié à cette action traite l'information reçue. Ce traitement consiste à choisir la vue appropriée et à lui fournir les informations dont elle a besoin en les choisissant à partir du modèle.
  3. Le modèle fait appel au service de la couche Métier de l'architecture 3-tiers permettant d'exécuter la requête de sélection, et se met à jour en fonction des données reçues. Cette mise à jour peut être notifiée à la vue afin que cette dernière puisse être actualisée.

4. Le modèle indique au contrôleur que le traitement est terminé et envoie les données au contrôleur.
5. Le contrôleur met à jour la vue afin de présenter les données à l'utilisateur.

Figure 2-2

Architecture MVC



Le modèle MVC et la notion de design pattern

Le modèle MVC est un design pattern (modèle de conception), ce qui signifie qu'il peut être appliqué dans tout type d'applications et ne se limite donc pas aux applications Flex.

Si nous transposons cette explication à l'utilisation du framework Flex, l'implémentation du modèle peut s'effectuer à l'aide du langage ActionScript, dont l'orientation objet, très poussée, permet de créer facilement des classes et des méthodes associées. Le contrôleur, dont le rôle principal est l'interfaçage entre le modèle et la vue, sera lui aussi développé à l'aide de ce langage et souvent mis en application grâce à la notion de DataBinding (voir chapitre 12). La vue sera quant à elle implémentée à l'aide du langage MXML prévu à cet effet ; celui-ci ayant en effet pour fonction principale la description des interfaces graphiques.

Balise MODEL

Le langage MXML met à disposition la balise `MODEL` qui permet, comme son nom l'indique, de déclarer un modèle de données.

Les langages ActionScript et MXML peuvent vous être totalement inconnus si vous n'êtes pas habitués à développer des applications Flash. Comme nous l'avons déjà évoqué, une application Flex est le résultat du mariage de ces deux langages où l'un permet de créer l'interface graphique et l'autre de mettre en place la logique applicative. C'est donc tout naturellement que nous allons poursuivre par la présentation de ces deux langages.

Le langage MXML

Le langage MXML (*Macromedia fleX Markup Language*) est avant tout un langage de balises basé sur XML.

Dans sa version 2009, ce langage n'inclut plus la fonctionnalité d'agencement des composants graphique (réalisée à présent à l'aide de la bibliothèque Spark) mais permet d'implémenter des composants de type métier à l'aide de balises spécifiques : `<fx :Script>` pour la mise en place de scripts ActionScript, `<fx :Array>` pour la déclaration d'un tableau, etc.

Dans la mesure où le framework Flex a pour rôle de créer la couche graphique d'une application, il est indispensable que toute application soit composée d'au moins un fichier portant l'extension `.mxml` et dont le contenu de base sera le suivant (cas d'une application exécutée dans le navigateur web) :

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
           xmlns:s="library://ns.adobe.com/flex/spark"
           xmlns:mx="library://ns.adobe.com/flex/mx" minWidth="955"minHeight="600">

</s:Application>
```

La balise `s:Application` permet d'indiquer qu'il s'agit du fichier applicatif principal. Juxtaposée à cette balise, nous trouvons la déclaration des différents espaces de noms (`xmlns:xmlNameSpace`) utiles à la création d'application Flex.

Le premier d'entre eux est l'espace de noms permettant de déclarer l'usage du MXML 2009, qui sera utilisé dans l'application par le biais du préfixe `fx` (`<fx:boolean/>`….), le second spécifie l'utilisation de la bibliothèque Spark, qui sert au design de l'application. Cette bibliothèque sera utilisée par l'usage du préfixe `s` (`<s :Button/>`). Enfin, nous trouvons l'espace de noms se référant au langage MXML 2006 utilisé par les anciennes versions de Flex que nous pouvons mettre en œuvre à l'aide du préfixe `mx`. (`<mx:Accordion/>`).

Chaque balise MXML correspond à une classe ou à des propriétés de classes d'ActionScript. Lors de la phase de compilation du fichier applicatif principal, l'ensemble du fichier MXML est analysé et transformé en classes ActionScript, ces dernières étant précompilées en code-octet (pour être exécutées par le Flash Player) et stockées dans un fichier au format SWF. Vous l'aurez compris, tout ce qui est effectué à l'aide du langage MXML pourrait être réalisé directement avec ActionScript.

Définition d'un espace de noms XML

Un espace de noms XML permet d'associer toutes les balises d'un langage à un groupe afin de mêler différents langages au sein d'un unique document XML.

Multiplicité des espaces de noms dans le framework Flex

Comme nous venons de l'indiquer, il est nécessaire de préfixer les espaces de noms afin d'éviter toute confusion quant à leur utilisation dans le code. La question que l'on est en droit de se poser est l'utilité d'en déclarer plusieurs au sein d'une même application Flex. Au chapitre 11, nous verrons que pour utiliser des composants personnalisés, nous aurons recours à plusieurs espaces de noms, chacun étant associé à un composant.

Le langage ActionScript

L'ECMA (*European Computer Manufacturers Association*) est une organisation travaillant dans le domaine de la standardisation informatique, notamment des langages de script grâce au standard ECMAScript. Ce standard couvre, entre autres choses, les langages ActionScript et JavaScript. C'est la raison pour laquelle leurs syntaxes se ressemblent en bien des points.

ActionScript est donc un langage standardisé (via le standard ECMA-262) dont l'usage est restreint aux applications Flash et donc à celles créées à l'aide du framework Flex. Ce langage est orienté objet et permet le développement de la partie dynamique de l'application ainsi que l'implémentation de la notion de modèle abordée précédemment.

Afin que vous puissiez lire facilement les extraits de code ActionScript présents dans cet ouvrage, nous avons jugé bon de vous apporter les bases de ce langage en abordant des notions communes à tout apprentissage d'un nouveau langage :

- les variables ;
- les boucles et les conditions ;
- les opérations booléennes ;
- les procédures et les fonctions ;
- la programmation objet.

Version d'ActionScript

Le framework Flex 4 utilise le langage ActionScript dans sa version 3.

Les variables

Les variables permettent de stocker des informations en vue de les utiliser dans divers traitements.

Une variable est déclarée par un nom précédé de la mention `var`. Il est également indispensable de spécifier le type de la variable déclarée :

```
var nomVariable:type
```

Le tableau 2-1 présente des exemples de déclaration de variables.

Tableau 2-1 – Déclaration de variables

Code	Description
`var maChaine:String`	Chaîne de caractères.
`var monNombre:Number`	Nombre décimal pouvant être positif ou négatif.
`var monEntier:int`	Nombre entier pouvant être positif ou négatif.
`var monBooleen:Boolean`	Booléen prenant pour valeur `0`, `1`, `true` ou `false`.
`var monTableau:Array = new Array("1","2","3")`	Tableau contenant trois éléments.

Les boucles et les conditions

Nous sommes souvent confrontés à la mise en place de boucles de traitement et à la vérification de conditions. Voyons comment les implémenter.

Les boucles

Le tableau 2-2 présente la transcription en langage ActionScript des principaux algorithmes de boucles. Ils permettent de réaliser un traitement selon un nombre d'itérations prédéfinies.

Tableau 2-2 – Les boucles

Algorithme	Transcription

`Pour i est un entier = 1 à i<10 Faire`
`[Traitement]`
`i = i +1`
`Fin pour`
	`for (var i:int=1; i<10;i++) {`
	`[Traitement]`
	`}`

`I est un entier = 1`
`Tant que I<10 faire`
`i = i + 1`
`Fin tant que`
	`var i:int = 1;`
	`while (i<10){`
	`  i++`
	`}`

`Pour chaque élément E contenu dans un tableau T`
`[Traitement]`
`Fin pour`
	`for each(var E:Object in T){`
	`[Traitement]`
	`}`

Précisions sur for each

Comme nous venons de le voir, les boucles de type `for each` servent à parcourir les éléments d'un tableau. Ce principe d'itération est également très pratique si le tableau contient des objets, car nous accédons alors directement aux attributs de ces objets avec la syntaxe suivante : `o.nomattribut`

Les conditions

Il existe deux types de conditions : la première consiste à vérifier si une situation est vraie, la seconde à exécuter un traitement selon un cas précis. Le tableau 2-3 présente leur implémentation.

Tableau 2-3 – Les conditions

Algorithme	Transcription

`x est un entier = 2`
`Si x est supérieur à 10 Alors`
`[Traitement]`
`Sinon`
`[Traitement]`
`Fin Si`
	`var x:int =2`
	`if (x>10) {`
	`[Traitement]`
	`} else {`
	`[Traitement]`
	`}`

`x est un entier = 10`
`Selon x`
`Cas 10 :`
`Cas 20 :`
`Fin selon`
	`var x:int = 2`
	`Switch (i) {`
	`Case 10 :`
	`[Traitement]`
	`Break;`
	`Case 20 :`
	`[Traitment]`
	`Break;`
	`}`

Vous pourriez très bien utiliser une suite de `if`, `else… if` et `else` à la place d'un `switch`, mais ce dernier vous apporte un plus non négligeable en matière de lisibilité du code.

Pour aller plus loin

Si vous ne précisez pas de condition dans une instruction `if`, le langage ActionScript réalise des vérifications par défaut. Ainsi, si vous testez une variable de type `Boolean`, les instructions suivantes sont équivalentes :

```
if (bool == true)
if (bool)
```

Les variables de type `Number`, `int` ou `uint` ne pouvant avoir la valeur `null`, si vous n'ajoutez pas de comparateur, ils seront traités comme `Boolean (1 = true)`.

Les autres types (`String, Object`,…) sont toujours comparés à la valeur `null`. Ainsi, les notations suivantes sont également équivalentes :

```
var myString:String = "Test"; if (myString != null){…}
var myString:String = "Test"; if (myString){…}
```

Pour pouvoir vérifier si une condition est vraie, nous avons besoin d'opérateurs de comparaison. Le tableau 2-4 en dresse la liste exhaustive :

Tableau 2-4 – Les opérateurs

Expression	Description
`X == Y`	X est égal à Y.
`X != Y`	X est différent de Y.
`X === Y`	X est strictement égal à Y.
`X !== Y`	X est strictement différent de Y.
`X > Y`	X est supérieur à Y.
`X < Y`	X est inférieur à Y.
`X >= Y`	X est supérieur ou égal à Y.
`X <= Y`	X est inférieur ou égal à Y.

Pour aller plus loin

En ActionScript 3 comme en Java, vous pouvez utiliser l'opérateur dit ternaire. Il effectue des tests simples sur une seule ligne. L'utilisation d'un opérateur ternaire est équivalente à un `if/else` mais plus compacte. Les notations suivantes sont équivalentes :

```
if(myVar != "Test"){myVar= "Test1";} else { myVar= "Test2";}
myVar = myVar != "Test" ? "Test1" : "Test2";
```

Dans la première partie, avant l'opérateur `?`, on précise la condition. Avant l'opérateur `:` se trouve la partie de code qui sera exécutée sur la condition. Si la condition n'est pas vérifiée, la partie de code qui se trouve après l'opérateur `:` sera exécutée.

Attention, cette notation améliore la lisibilité de votre code uniquement si la condition est simple. Pour celles plus complexes, avec opération booléenne par exemple, préférez le couple `if/else`.

Les opérations booléennes

Comment déclencher un traitement lorsque deux conditions sont vraies ou lorsque l'une des deux est vraie ? Le tableau 2-5 présente les opérateurs capables de réaliser ces opérations de vérification.

Tableau 2-5 – Les opérations booléennnes

Expression	Description
`! X`	`X` prend la valeur opposée. Ainsi, si `X = vrai` alors `! X = faux`.
`X && Y`	X ET Y doivent être vérifiés (égaux à vrai) pour que le traitement puisse être effectué.
`X || Y`	X OU Y doit être vrai pour que le traitement soit effectué.

Les procédures et les fonctions

Les procédures et les fonctions sont déclarées par un nom, précédé de la mention `function`. Elles ont pour rôle d'exécuter un traitement susceptible d'être appelé plusieurs fois dans un script. Une fonction retourne un résultat dont le type est spécifié en fin de déclaration. A contrario, une procédure ne retourne rien, nous spécifierons donc `void` à la place d'un type de retour.

Portée des procédures et des fonctions

La portée des procédures et des fonctions doit obligatoirement être mentionnée :

- `private` (privée) : la procédure ou la fonction ne peut être appelée que par la classe qui la contient.
- `protected` (protégée) : la procédure ou la fonction ne peut être appelée que par la classe qui la contient ou par la classe héritant de cette classe.
- `public` (publique) : la procédure ou la fonction peut être appelée par l'ensemble des procédures et fonctions du projet.
- `static` (statique) : la procédure ou la fonction n'est créée qu'une seule fois par classe et non pas dans chaque objet instancié par la classe. On peut donc l'appeler sans instancier l'objet dans lequel elle est déclarée.

Notons que ces portées peuvent également être appliquées aux variables.

L'exemple de code suivant présente la déclaration d'une procédure dont la portée est publique et qui accepte un paramètre de type chaîne.

```
public function enregistrerDonnees(donnees:string):void {
[Traitement d'enregistrement]
}
```

Dans l'exemple suivant, on identifie qu'il s'agit d'une fonction renvoyant une chaîne de caractères par la mention `:String`. On constate également que la portée de la fonction est privée par la présence du terme `private`.

```
private function afficherDonnee():String{
var laDonneeARenvoyer:String="info"
return laDonneeARenvoyer
}
```

La programmation objet

Décrire l'ensemble des fonctionnalités d'ActionScript en termes de programmation objet prendrait une place trop importante au sein de cet ouvrage. C'est par l'analyse de quelques lignes de code que nous vous proposons de découvrir comment implémenter une classe et ses méthodes.

```
package com.monpackage
{
import spark.components.Button;

public class MaClasse extends Button
{

    // Attribut
    private var _nom:String


    public function MaClasse(parametre:String)
    {
        super();
        _nom = parametre;
    }


    //Getter et Setter
    public function get nom():String
    {
        return _nom;
    }

    public function set nom(value:String):void
    {
        _nom = value;
    }

}
}
```

Analysons à présent ce code. La première ligne sert à définir le package dans lequel se situe la classe.

La notion de package

Le fait de séparer votre code en différents packages vous permettra par la suite de pouvoir le relire ou le modifier facilement. Il s'agit ni plus ni moins de le structurer en dossiers et sous-dossiers. En ActionScript 3, vous devez obligatoirement placer vos classes dans le sous-dossier correspondant au package déclaré en en-tête de votre classe. Par convention, les noms de package s'écrivent toujours en minuscules.

Vient ensuite l'importation des classes nécessaires aux traitements décrits dans les méthodes ou à la notion d'héritage. En effet, le compilateur ne connaissant pas votre code, il vous faut lui indiquer explicitement quelles classes seront utilisées.

On constate qu'une classe est déclarée par la mention `class` suivie du nom attribué à cette classe. La notion d'héritage est ici implémentée à l'aide du mot-clé `extends`. Dans notre cas, la classe `MaClasse` hérite de la classe `Button`, elle-même déclarée dans le package `spark.component`. Grâce à l'héritage, une classe fille possède les même attributs et méthodes que la classe mère, ce qui signifie que notre classe pourra réaliser les mêmes actions et possèdera les même attributs que celles définie dans la classe `Button` dont voici le contenu :

```
////////////////////////////////////////////////////////////////////////////////
//
//  ADOBE SYSTEMS INCORPORATED
//  Copyright 2008 Adobe Systems Incorporated
//  All Rights Reserved.
//
//  NOTICE: Adobe permits you to use, modify, and distribute this file
//  in accordance with the terms of the license agreement accompanying it.
//
////////////////////////////////////////////////////////////////////////////////

package spark.components
{

import flash.events.Event;
import spark.components.supportClasses.ButtonBase;
import mx.core.IButton;


/**
 *  Color applied to the button when the emphasized flag is true.
 *
 *  @default #0099FF
 *
 *  @langversion 3.0
 *  @playerversion Flash 10
 *  @playerversion AIR 1.5
 *  @productversion Flex 4
 */
[Style(name="accentColor", type="uint", format="Color", inherit="yes", theme="spark")]

[Exclude(name="textAlign", kind="style")]

//--------------------------------------
//  Other metadata
//--------------------------------------

[IconFile("Button.png")]

/**
```

```
*  The Button component is a commonly used rectangular button.
*  The Button component looks like it can be pressed.
*  The default skin has a text label.
*  Define a custom skin class to add an image to the control.
*
*  <p>Buttons typically use event listeners to perform an action
*  when the user selects the control. When a user clicks the mouse
*  on a Button control, and the Button control is enabled,
*  it dispatches a <code>click</code> event and a <code>buttonDown</code> event.
*  A button always dispatches events such as the <code>mouseMove</code>,
*  <code>mouseOver</code>, <code>mouseOut</code>, <code>rollOver</code>,
*  <code>rollOut</code>, <code>mouseDown</code>, and
*  <code>mouseUp</code> events whether enabled or disabled.</p>
*
*  <p>The Button control has the following default characteristics:</p>
*     <table class="innertable">
*        <tr>
*           <th>Characteristic</th>
*           <th>Description</th>
*        </tr>
*        <tr>
*           <td>Default size</td>
*           <td>Wide enough to display the text label of the control</td>
*        </tr>
*        <tr>
*           <td>Minimum size</td>
*           <td>21 pixels wide and 21 pixels high</td>
*        </tr>
*        <tr>
*           <td>Maximum size</td>
*           <td>10000 pixels wide and 10000 pixels high</td>
*        </tr>
*        <tr>
*           <td>Default skin class</td>
*           <td>spark.skins.spark.ButtonSkin</td>
*        </tr>
*     </table>
*
*  @mxml <p>The <code>&lt;s:Button&gt;</code> tag inherits all of the tag
*  attributes of its superclass and adds the following tag attributes:</p>
*
*  <pre>
*  &lt;s:Button
*    <strong>Properties</strong>
*    emphasized="false"
*  /&gt;
*  </pre>
```

```
 *
 *  @see spark.skins.spark.ButtonSkin
 *  @includeExample examples/ButtonExample.mxml
 *
 *  @langversion 3.0
 *  @playerversion Flash 10
 *  @playerversion AIR 1.5
 *  @productversion Flex 4
 */
public class Button extends ButtonBase implements IButton
{
    include "../core/Version.as";

    //--------------------------------------------------------------------------
    //
    //  Constructor
    //
    //--------------------------------------------------------------------------

    /**
     *  Constructor.
     *
     *  @langversion 3.0
     *  @playerversion Flash 10
     *  @playerversion AIR 1.5
     *  @productversion Flex 4
     */
    public function Button()
    {
        super();
    }

    //--------------------------------------------------------------------------
    //
    //  Properties
    //
    //--------------------------------------------------------------------------

    //----------------------------------
    //  emphasized
    //----------------------------------

    /**
     *  @private
     *  Storage for the emphasized property.
     */
    private var _emphasized:Boolean = false;

    [Bindable("emphasizedChanged")]
```

```
[Inspectable(category="General", defaultValue="false")]

/**
 * Reflects the default button as requested by the
 * focus manager. This property is typically set
 * by the focus manager when a button serves as the
 * default button in a container or form.
 * When set to true, the <code>emphasized</code> style
 * is appended to the button's <code>styleName</code>
 * property.
 *
 *  @default false
 *  @see mx.managers.FocusManager.defaultButton
 *  @langversion 3.0
 *  @playerversion Flash 10
 *  @playerversion AIR 1.5
 *  @productversion Flex 4
 */
public function get emphasized():Boolean
{
    return _emphasized;
}

/**
 *  @private
 */
public function set emphasized(value:Boolean):void
{
    if (value == _emphasized)
        return;

    _emphasized = value;
    emphasizeStyleName();
    dispatchEvent(new Event("emphasizedChanged"));
}

//--------------------------------------------------------------------------
//
//  Overridden methods
//
//--------------------------------------------------------------------------

/**
 *  @private
 */
override public function set styleName(value:Object):void
{
    super.styleName = value;
```

```
        // We need to ensure to re-apply the emphasized style if appropriate.
        if (value == null || value is String)
        {
            if (!value || (_emphasized && value.indexOf(" emphasized") == -1))
                emphasizeStyleName();
        }
    }

    //--------------------------------------------------------------------------
    //
    //  Methods
    //
    //--------------------------------------------------------------------------

    /**
     *  @private
     */
    private function emphasizeStyleName():void
    {
        var style:String = styleName is String ? styleName as String : "";

        if (!styleName || styleName is String)
        {
            if (_emphasized)
                super.styleName = style + " emphasized";
            else
                super.styleName = style.split(" emphasized").join("");
        }
    }
}
}
```

Les attributs spécifiques à la classe sont ensuite déclarés par des variables. Il est ainsi possible d'écrire les méthodes d'accès à cet attribut comme suit : `set nomattribut(value:type)` et `get nomattribut():type`.

Par convention…

Par convention, les variables (`private` ou `protected`) sont précédées du caractère de soulignement _ et l'attribut passé dans les méthodes `set` s'appellera `value`. L'application des conventions de nommage est primordiale pour un développement professionnel. Suivre ces conventions vous permet d'avoir un code cohérent, aussi bien dans vos classes qu'avec les classes du framework Flex qui suivent ces conventions. Si vous souhaitez approfondir vos connaissances sur le sujet des conventions de nommage, nous vous invitons à consulter le lien suivant :

http://opensource.adobe.com/wiki/display/flexsdk/Coding+Conventions

La méthode `set` a pour fonction d'affecter une valeur à un attribut, laquelle peut ensuite être récupérée par l'emploi de la méthode `get`.

Enfin, pour créer des objets à partir d'une classe, il est nécessaire d'instancier cette dernière. Pour cela, nous disposons d'un constructeur. Il s'agit d'une méthode portant le nom de la classe et qui sera automatiquement appelée au moment de l'instanciation de l'objet. À noter que le type de retour n'est pas précisé pour un constructeur.

Nous venons de décrire les éléments de bases du langage ActionScript dont vous aurez besoin pour comprendre au mieux les différents extraits de code présentés dans cet ouvrage. Néanmoins, les subtilités de ce langage n'ont pas été abordées, ces dernières ne pouvant être découvertes que par l'acquisition d'un ouvrage spécialisé ou par la pratique d'ActionScript. Nous vous conseillons également de consulter les sites web spécialisés tels que le site ActionScript.org (*http://www.actionscript.org/*), qui constitue une mine d'informations sur ce langage et propose des cours et tutoriaux en téléchargement.

Intéressons nous à présent au mécanisme client/serveur d'une application Flex.

Le mécanisme client/serveur

Comme nous l'avons vu, une application Flex est composée d'un fichier MXML principal faisant appel à des fichiers ActionScript permettant d'intégrer la notion d'architecture MVC. L'ensemble de ces fichiers est ensuite compilé, donnant naissance à un fichier SWF pouvant être intégré dans une page HTML. Ceci n'est qu'une petite partie du processus. Nous allons à présent rassembler l'ensemble des informations véhiculées tout au long de ce chapitre afin d'observer le mécanisme complet d'une application Flex.

Tout d'abord, nous avons vu qu'il est possible d'architecturer une application en trois couches distinctes, chacune d'entre elles pouvant être hébergée (sous sa forme extrême) sur trois serveurs différents :

- le serveur de base de données (couche Accès aux données) ;
- le serveur d'application (couche Métier) ;
- le serveur Flex (couche Présentation contenant le fichier SWF et la page web associée).

Imaginons à présent un utilisateur souhaitant utiliser l'application Flex située à l'adresse *http://www.monapplicationflex.com* et pointant bien évidemment sur le serveur de présentation qui sera le point d'entrée du processus (figure 2-6).

La première phase va consister à offrir l'application au client connecté (figure 2-6, ❶ et ❷). Nous entendons par « offrir » le fait que l'application sera téléchargée et exécutée sur le poste client, ce qui nécessite par conséquent que Flash Player soit installé sur celui-ci.

Attention

Le fait qu'une application soit exécutée sur le poste client induit un paramétrage consciencieux de cette dernière. Lors du développement de l'application, nous devons faire appel aux services situés sur des serveurs présents sur notre réseau local. Lorsque l'application est mise en ligne, n'oubliez pas que ces serveurs ne sont plus accessibles via l'adresse IP locale mais à l'aide de leur adresse publique ! Les schémas des figures 2-3, 2-4 et 2-5 illustrent ce principe.

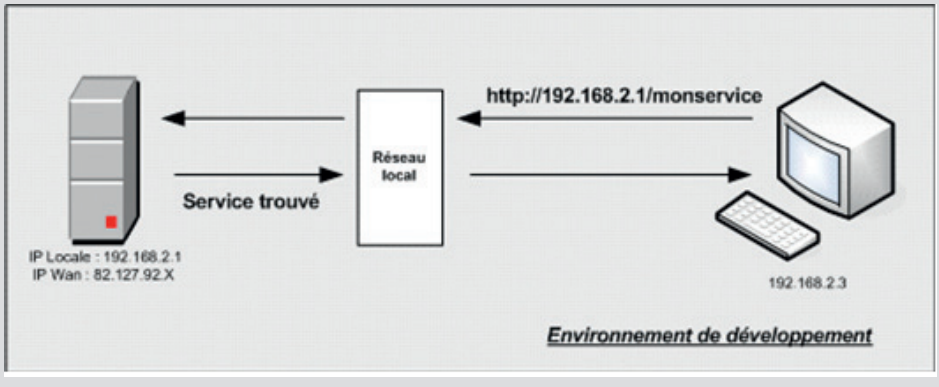


Figure 2-3

Accès aux services dans un environnement de développement

Sur ce schéma, nous pouvons accéder facilement aux services présents sur le serveur puisqu'il fait partie de notre réseau local.

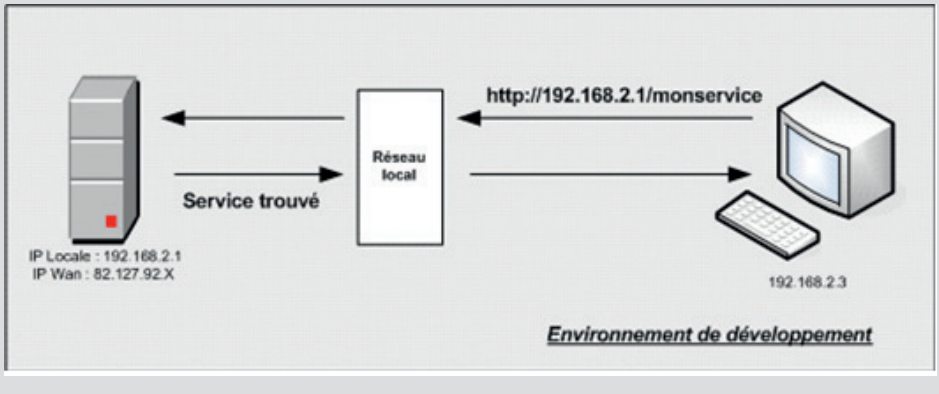


Figure 2-4

Paramétrage incorrect de l'application : recherche des services sur le réseau local du client