



Expert
EXPERT

Apache NetBeans

Développez vos applications
en **Java**

En téléchargement

 projets Java

 + QUIZ

Version en ligne
OFFERTE !
pendant 1 an

Thomas BROUSSARD
Romain LEMOUNEAU



Les éléments à télécharger sont disponibles à l'adresse suivante :
<http://www.editions-eni.fr>
Saisissez la référence de l'ouvrage **EIJAVNETB** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

Chapitre 1

Installation et configuration initiale

- 1. Introduction 13
- 2. Installation du kit de développement Java 14
 - 2.1 Avant de commencer : note sur la plateforme Java 14
 - 2.1.1 Les éléments de la plateforme : JVM, JRE, JDK 14
 - 2.1.2 Focus sur la JVM. 14
 - 2.1.3 Un environnement pour chaque utilisation 15
 - 2.2 Quel JDK sélectionner? 15
 - 2.3 Quelle version sélectionner? 17
- 3. Installation de NetBeans 19
- 4. Configuration initiale 23

Chapitre 2

Créer son premier projet avec NetBeans

- 1. Introduction 35
- 2. Les types de projets Java 36
- 3. Création d'un projet Java simple avec Maven 42
- 4. Notions de programmation orientée objet 44
 - 4.1 Définition d'une classe 44

2 _____ Apache NetBeans

Développez vos applications en Java

4.2	Modélisations, classes, types et exécution en Java	45
4.2.1	Les classes métier	46
4.2.2	Les points d'entrée d'exécution du programme	47
4.3	Notion de package	47
4.3.1	Définition	47
4.3.2	Règle de nommage	48
5.	Présentation de l'interface de NetBeans	49
6.	Création d'une classe Java avec NetBeans	51
6.1	Création d'une classe	51
6.2	Création d'un attribut	53
6.3	Création d'un accesseur et d'un mutateur	54
6.4	Création d'une méthode	56
7.	Définition d'un point d'entrée pour l'exécution de son programme	57
8.	Compilation d'un projet	58
8.1	Définition du compilateur	58
8.2	Compilation d'un projet avec NetBeans	58
9.	Exécution d'un projet	59
9.1	Sélectionner le point d'entrée du programme	59
9.2	Construire son projet avec NetBeans	60
10.	Documentation d'une application	63
10.1	Tags Javadoc	63
10.2	Analyse de la Javadoc	67
10.3	Exécution de la Javadoc	68
11.	Exercice corrigé : Calcul géométrique	72

Chapitre 3
Les outils du développeur

- 1. Introduction 83
- 2. Débogage d'une application grâce à NetBeans 83
 - 2.1 Découvrir le mode debug 84
 - 2.2 Apprendre les notions importantes 87
 - 2.2.1 Les déplacements pas à pas 87
 - 2.2.2 Les points d'arrêts 88
 - 2.2.3 Les observateurs 89
 - 2.3 Déboguer un programme 90
- 3. Capacités de génération de code 93
 - 3.1 Les attributs 93
 - 3.2 Les constructeurs 94
 - 3.3 Les méthodes equals() et hashCode() 95
 - 3.4 La méthode toString() 97
 - 3.5 Les méthodes comportementales 98
 - 3.6 La méthode clone() 100
 - 3.6.1 Le clonage en surface 101
 - 3.6.2 Le clonage en profondeur 103
- 4. Refactoring avec NetBeans 104
 - 4.1 L'introduction de variables 104
 - 4.2 L'introduction de constantes 105
 - 4.3 L'introduction d'attributs 106
 - 4.4 L'introduction de méthodes 107
 - 4.5 L'introduction de paramètres 109
- 5. Internationalisation d'un projet 110
- 6. Diagrammes de classes UML avec NetBeans 117
 - 6.1 Installation du plugin easyUML 120
 - 6.2 Génération d'un diagramme de classes 122
- 7. Groupement de projets 128

Chapitre 4

Le développement d'applications riches

1. Introduction	133
2. Objectif des applications riches	134
3. Programmation graphique : composants et évènements	135
4. Bonnes pratiques pour la programmation au sein des frameworks graphiques	136
4.1 Séparation de la couche métier et de la couche de présentation	137
4.2 Intrication du code métier et du code de présentation	138
4.3 Débordement de la couche métier dans la couche graphique .	139
5. Cas concret d'utilisation d'une librairie graphique et implémentation avec NetBeans	139
6. Implémentation d'un exemple avec Swing	141
6.1 Maquettage avec Swing	141
6.1.1 Création d'un projet Java pour démarrer avec Swing . .	141
6.1.2 Fonctionnement global de Swing et intégration dans NetBeans	145
6.1.3 Édition des propriétés, génération d'évènements	146
6.1.4 Utilisation de l'historique	148
6.2 Composants et programmation des interactions	149
6.2.1 Manipulation du JTable	152
6.2.2 Manipulation du JFileChooser	154
6.2.3 Mise en place d'un thème	155
7. Implémentation d'un exemple avec JavaFX	157
7.1 Différence d'approche entre Swing et JavaFX	157
7.2 Maquettage avec JavaFX	158
7.2.1 Création d'un projet JavaFX	159
7.2.2 Configuration de Scene Builder dans NetBeans	161
7.2.3 Première exécution	162
7.2.4 Fonctionnement global de JavaFX	164
7.2.5 Prise en main de SceneBuilder	165

7.2.6	Création d'un panneau (Scene)	167
7.2.7	Programmation des interactions avec l'utilisateur	171
7.3	Composants et programmation des interactions	174
7.3.1	Manipulation du TableView	174
7.3.2	Manipulation du FileChooser	177
8.	Swing ou JavaFX?	178

Chapitre 5

Vers l'industrialisation du logiciel

1.	Introduction	181
2.	Travail en équipe : intégration avec Git	181
2.1	Fonctionnement de Git	182
2.1.1	Git, un système distribué	182
2.1.2	L'aspect concurrent de Git	183
2.2	Git : branches et tags	184
2.3	Partage et modification d'un projet avec Git et NetBeans	186
2.3.1	Intégration initiale	186
2.3.2	Réalisation et intégration (locale et distante) de modifications	188
2.3.3	Intégration de la modification sur le repository local	189
2.3.4	Intégration de la modification dans un repository distant	190
2.4	Suivi des modifications	191
2.4.1	Intégration dans l'éditeur de code	191
2.4.2	Affichage de l'historique d'un fichier	191
2.4.3	Gestion des conflits	193
3.	Construction du projet : utilisation de Maven dans NetBeans	194
3.1	Fonctionnement de Maven	194
3.1.1	Maven : la convention comme philosophie	194
3.1.2	Structuration du projet	195
3.1.3	Gestion des dépendances	196
3.1.4	Système de coordonnées	196

6 Apache NetBeans

Développez vos applications en Java

3.1.5	Référentiels (repository) d'artefacts	197
3.1.6	Nature d'une dépendance : packaging et classifier	199
3.2	Configuration de la construction du projet	200
3.2.1	Flux de construction d'un projet Maven	200
3.2.2	Instructions de construction grâce aux plugins	201
3.2.3	Exemple : mise en œuvre avec la construction de la Javadoc	202
3.2.4	Configuration des plugins Maven	204
3.3	Déclenchement de constructions depuis NetBeans	205
3.3.1	Première construction	205
3.3.2	Historiques et enregistrement des paramètres de construction dans NetBeans	206
3.4	Ajout de dépendances	208
3.4.1	Ajout assisté dans le pom.xml	208
3.4.2	Visualisation de l'arbre des dépendances	208
3.4.3	Conflits et exclusion de dépendance	211
3.5	Réutilisation de projets : héritage et composition	212
3.5.1	Définition d'un projet parent	213
3.5.2	Déclaration du parent dans les projets fils	214
3.5.3	Effet de la déclaration d'un parent sur le projet	215
3.5.4	Limites de l'héritage	216
3.5.5	Composition grâce au BOM	216
3.6	Construction groupée de projet : projets multimodules	217
3.6.1	Conception d'un projet multimodule	217
3.6.2	Distinction entre le réacteur et le parent	218
4.	Exécution des tests unitaires dans NetBeans	219
4.1	Méthodologie de définition du test unitaire	219
4.1.1	Approche systématique : hypothèse, action, vérification	219
4.1.2	Principe d'indépendance et de transparence des tests	220
4.1.3	Méthodologie de développement intégrant les tests : BDD et TDD	221
4.1.4	Exemple de formalisation du test BDD	222

- 4.1.5 Différence entre une erreur (error) et un échec (failure) . 223
- 4.2 Définition d'un test unitaire avec JUnit 224
 - 4.2.1 Écriture d'un premier test 225
 - 4.2.2 Exécution des actions systématiques
grâce aux annotations 229
 - 4.2.3 Assertions 230
 - 4.2.4 Interprétation du résultat des tests 231
 - 4.2.5 Génération des tests unitaires par NetBeans 232
- 5. Analyse de la qualité de code intégrée à l'outil 235
 - 5.1 Utilisation de l'analyseur de code intégré de NetBeans 235
 - 5.1.1 Analyse globale 236
 - 5.1.2 Analyse sur critère unique 237
 - 5.1.3 Analyse en vue d'une migration 237
 - 5.2 Couverture des tests 237
 - 5.2.1 Couverture en lignes et couverture en branches 237
 - 5.2.2 Configuration et analyse de la couverture de test
dans NetBeans 239
 - 5.2.3 Comparaison avec la couverture de test
de JaCoCo - Java Code Coverage 244
 - 5.3 Intégration de SonarQube 246
 - 5.3.1 Introduction à SonarQube 246
 - 5.3.2 Installation d'un serveur SonarQube local 247
 - 5.3.3 Les sept axes de la qualité selon SonarQube 248
 - 5.3.4 Installation de SonarLint4Netbeans 249
 - 5.3.5 Analyse d'un projet grâce au plugin Maven sonar 251
- 6. Exercice corrigé : Conception d'un service de lecture
de fichiers CSV en approche TDD 254
 - 6.1 Spécifications de l'exercice 254
 - 6.1.1 Préparation du projet 254
 - 6.1.2 Modélisation des données 254
 - 6.1.3 Désérialisation grâce à la classe CSVReader 255

6.2	Solution de l'exercice	255
6.2.1	Préparation du projet	255
6.2.2	Modélisation des données	257
6.2.3	Désérialisation grâce à la classe de service CSVReader	258
6.2.4	Recherche d'amélioration	263

Chapitre 6

La conception et l'exploitation de services

1.	Introduction	265
2.	Créer une base de données	266
2.1	Créer une base de données H2	266
2.2	Initialiser la base de données H2	269
3.	Développer des services REST pour exposer des APIs	272
3.1	Notion d'API	274
3.2	Principe KISS	275
3.3	Veiller à l'aspect Stateless de l'API	276
3.4	Verbes de la grammaire REST	277
3.5	Installation d'un serveur d'applications WildFly	279
3.6	Création d'un projet d'application Web	281
3.7	Génération d'entités et services REST	283
3.7.1	À propos des entités et de JPA	283
3.7.2	Manipulation	283
3.7.3	Bonnes pratiques de conception	287
3.8	Configuration et test de services REST	288
4.	Développer des services SOAP	292
4.1	Créer un projet Spring Boot	294
4.2	Générer les entités Java	295
4.3	Créer des objets de transfert de données	298
4.4	Développer des services Soap	301
4.5	Définir l'adresse d'un service web	304
4.6	Exécuter un projet	305

- 4.7 Tester des services Soap 306
- 5. Exercice corrigé : développer une API REST avec Spring 309
 - 5.1 Créer un projet Spring Boot 310
 - 5.2 Créer l'entité Etudiant 311
 - 5.3 Créer la classe EtudiantDAO 313
 - 5.4 Créer la classe EtudiantController 316
 - 5.5 Tester l'API REST 319

Chapitre 7
NetBeans, Java et le Web

- 1. Introduction 323
- 2. Présentation de l'architecture Java EE 324
 - 2.1 Le pattern Modèle - Vue - Contrôleur 325
 - 2.1.1 Composants de base du Web en Java 327
 - 2.1.2 MVC, MVC 2 et les frameworks Java 329
 - 2.2 Le pattern MVP 330
- 3. Création d'une interface web avec JSF 332
 - 3.1 Créer une application web 333
 - 3.2 Générer les entités Java 334
 - 3.3 Générer les pages JSF 336
 - 3.4 Tester l'interface web 339
 - 3.5 Manipuler la palette JSF de NetBeans 344
- 4. Création d'une interface JSP 350
 - 4.1 Créer une première page JSP 350
 - 4.2 Développer une application en JSP 353
 - 4.2.1 Les balises 353
 - 4.2.2 Les directives 354
 - 4.2.3 Le développement d'une page JSP 355
 - 4.2.4 Le développement d'une servlet 367
 - 4.2.5 Le développement du JavaScript 371

5. Exercice corrigé : développement d'une application web JSF	373
5.1 Créer une application Web	374
5.2 Créer l'entité Etudiant	375
5.3 Créer la classe EtudiantDAO	377
5.4 Créer la classe EtudiantPresenter	379
5.5 Créer les pages JSF	382
5.6 Tester l'application JSF	386
5.7 Aller plus loin	389

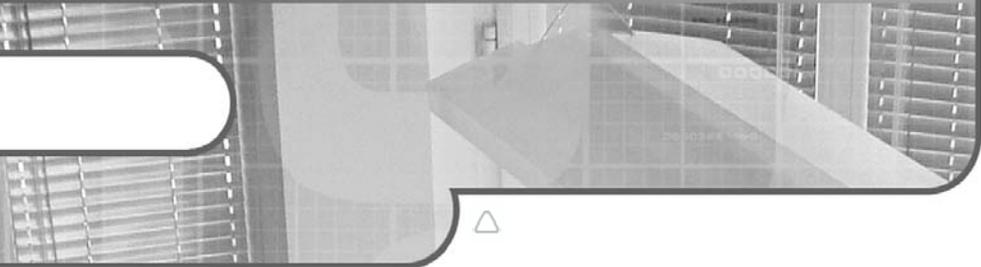
Chapitre 8

Le profilage d'applications Java

1. Introduction	391
2. Avant de commencer : les grands principes de fonctionnement de la JVM	392
2.1 Gestion automatique de la mémoire	392
2.1.1 Les grandes zones mémoire de la JVM	392
2.1.2 Les mécanismes de recyclage : garbage collector	394
2.2 Aspect multithread	395
3. Premier profilage d'une application avec NetBeans	396
3.1 Déclenchement et configuration du premier profilage	396
3.1.1 Lancement et calibration	398
3.1.2 Choix des métriques	400
3.1.3 Télémétrie globale de l'application	402
3.2 Techniques de relevés pour le profilage	403
3.2.1 Le sampling	404
3.2.2 L'observation continue	406
4. Analyses possibles relatives à la mémoire	406
4.1 Analyse globale de la mémoire	406
4.1.1 Observation et interprétation du comportement de la mémoire	407
4.1.2 Observation de la mémoire dans un cas de mauvaise conception	410

- 4.2 Analyse des instances en cours d'exécution 412
 - 4.2.1 Observation des instances en mode sampling 412
 - 4.2.2 Observation continue des instances en cours d'exécution 415
 - 4.2.3 Avantages et limites de l'observation de la mémoire en cours d'exécution 418
- 4.3 Analyses de la mémoire à froid 418
 - 4.3.1 Réalisation d'un dump et découverte du HeapWalker . . 419
 - 4.3.2 Comparaison de dumps 423
 - 4.3.3 Console OQL 425
- 5. Analyses possibles relatives à l'utilisation du processeur 427
- 6. Autres possibilités du profileur de NetBeans 429

- Index 431



Chapitre 2

Créer son premier projet avec NetBeans

1. Introduction

Le développement de projets informatiques répond à tout type de besoin. De la création d'un outil personnel automatisant une tâche quotidienne répétitive au développement de logiciels complexes utilisés par des centaines d'employés, l'environnement de développement intégré NetBeans peut être utilisé pour réaliser ces différents projets. Par conséquent, afin de sélectionner le projet le plus adapté au cahier des charges, il est nécessaire de découvrir les possibilités offertes par NetBeans.

Ce chapitre a pour objet de présenter les différents types de projets mis à disposition des développeurs, créer un projet Java simple, définir les bases de la programmation orientée objet à travers ce projet, présenter les vues de développement de NetBeans, développer une classe, rendre le projet exécutable, compiler le projet, l'exécuter, le documenter et proposer un exercice corrigé.

2. Les types de projets Java

Dans le monde professionnel, les projets Java sont régulièrement utilisés pour développer des applications de gestion web ou de bureau et pour réaliser des traitements manipulant un nombre important de données.

Le framework Java le plus récent régulièrement utilisé pour développer des applications de bureau est JavaFX . Sa mise en service a été réalisée pour remplacer les anciennes bibliothèques Swing et Abstract Window Toolkit (AWT). Les applications web actuelles sont fréquemment développées en deux parties distinctes : la partie interface utilisateur, en Angular, React ou Vue.js, et la partie serveur, en Java avec des micro-services Rest et les différentes bibliothèques Spring. Les traitements gérant un nombre conséquent de données sont ordinairement des projets Java utilisant la bibliothèque Springbatch.

NetBeans offre un choix important de projets ; dans un premier temps, leurs caractéristiques principales seront présentées.

■ Remarque

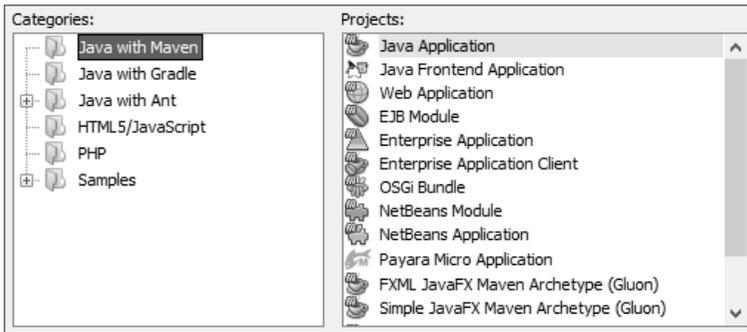
Note sur les projets NetBeans

NetBeans permet de créer un très grand panel de projets et contrairement à d'autres IDE, NetBeans force le développeur à utiliser un standard pour sa gestion de configuration.

Parmi les standards disponibles en Java, les trois majeurs sont Ant, Maven et Gradle (dans leur ordre historique d'apparition). L'utilisation de l'un de ces gestionnaires de configuration est une bonne approche, cela rend le projet indépendant de l'IDE utilisé et facilite ainsi la reprise de code par des gens habitués à ces standards.

L'installation de l'IDE terminée, à partir de l'écran d'accueil de NetBeans, effectuez la manipulation suivant.

■ Cliquez sur le menu **File - New Project** : l'écran ci-dessous vous permet de sélectionner le type de projet Maven que vous souhaitez créer.



La première catégorie, *Java with Maven*, est composée de quinze types de projets différents. Maven est un outil open source d'Apache permettant de faciliter la gestion d'un projet. Il permet par exemple d'automatiser des tâches de compilation, de déploiement ou bien de gérer les dépendances entre les bibliothèques utilisées par le projet.

Type	Description
Java Application	Permet la création de programmes basiques en java. L'archive finale de ce projet est un JAR (<i>Java Archive</i>).
Java Frontend Application	Permet la création d'applications front exécutable par exemple depuis un navigateur. L'application est composée de deux parties, la partie cliente (front-end) et la partie serveur (back-end) ; les deux sont articulées par l'intermédiaire du pattern MVVM. L'archive finale de ce projet est un JAR (<i>Java Archive</i>).
Web Application	Permet la création d'applications web exécutable depuis un serveur d'applications, par exemple Apache Tomcat. L'archive finale de ce projet est un WAR (<i>Web application Archive</i>).
EJB Module (<i>Enterprise java-Beans</i>)	Permet la création de composants Java dédiés principalement à la gestion des transactions, de la sécurité. Ils sont exécutés exclusivement sur le serveur.

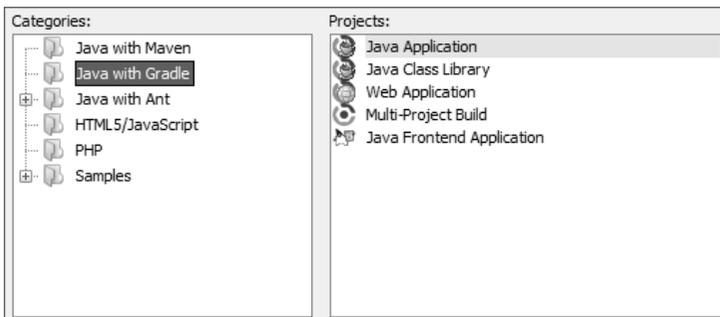
Type	Description
Enterprise application	Permet la création d'applications agrégeant des applications web et de modules EJB. L'archive finale de ce projet est un EAR (<i>Enterprise application Archive</i>).
Enterprise application Client	Permet la création d'applications web et des services web de type <i>Representational state transfer</i> (Rest). L'archive finale de ce projet est un WAR (<i>Web application Archive</i>).
OSGi Bundle (<i>Open Services Gateway initiative</i>)	<i>Framework</i> utilisé pour la programmation modulaire en Java. Il permet une meilleure gestion des versions et dépendances entre les composants installés dans la machine virtuelle Java (JVM).
NetBeans Module	Permet la création d'un composant d'une application basée sur la plateforme NetBeans.
NetBeans Application	Permet la création d'une application de bureau basée sur la plateforme NetBeans. Cela permet de réutiliser toute l'infrastructure applicative prévue par NetBeans (gestion des menus, des vues, des outils, des enregistrements et des notifications) sans avoir à le développer à nouveau.
Paraya Micro Application	Permet la création de plateformes utilisées pour le déploiement de micro-services Java.
FXML JavaFX Maven Archetype (Gluon)	Permet la création d'applications de bureau FXML Java. Ce type de projet permet de développer des écrans grâce à des fichiers de type fxml et de gérer leur comportement dans des classes Java.

Type	Description
Simple JavaFX MAven Archetype (Gluon))	Permet la création de projets utilisant le <i>framework</i> JavaFX. Il donne accès aux développeurs à une bibliothèque d'interface utilisateur utilisée pour créer des applications de bureau ou des applications exécutées sur une JVM et lancées dans un navigateur web.
POM Project	Permet la création d'un package contenant un fichier POM parent pouvant être ensuite utilisé pour assembler plusieurs modules.
Project from Archetype	Permet la création d'un projet Maven simple à partir d'un archétype précis, sélectionné à la création du projet.
Project with Existing POM	Permet l'ouverture d'un projet Maven à partir d'un fichier POM existant.

Tableau. Description des projets disponibles

❑ Cliquez sur le menu **Java with Gradle** ; l'écran ci-dessous vous permet de sélectionner le type de projet Gradle que vous souhaitez créer.

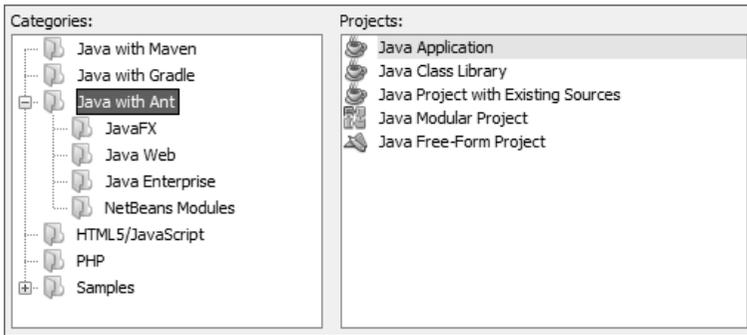
La seconde catégorie liste les projets Java avec **Gradle**. *Gradle* est un outil de construction d'applications Java, Groovy, Scala et Java EE. Cette alternative à Maven ou Ant intègre ses deux concurrents en mixant les avantages de chacun. Il peut être intégré à une infrastructure existante afin de pallier des problèmes, par exemple de *build*, posés par Maven ou Ant.



Type	Description
Java Application	Permet la création de programmes basiques en Java avec Gradle. Sélectionner ce projet permet la création automatique d'une main java Main utilisée lors de l'exécution du projet.
Java Class Library	Permet la création d'un projet Gradle contenant une librairie Java.
Web Application	Permet la création d'un projet web Java EE avec Gradle.
Multi-Project Build	Permet la création d'un projet autorisant la construction de plusieurs sous-projets. Ainsi, chacun de ces modules peut être développé séparément puis construit grâce à un projet unique.
Java Frontend Application	Permet la création d'applications en Java, HTML, CSS et JavaScript, exécutées depuis un navigateur web.

■ Cliquez sur le menu **Java with Ant** et déroulez les sous-menus ; l'écran ci-après vous permet de sélectionner le type de projet Ant que vous souhaitez créer.

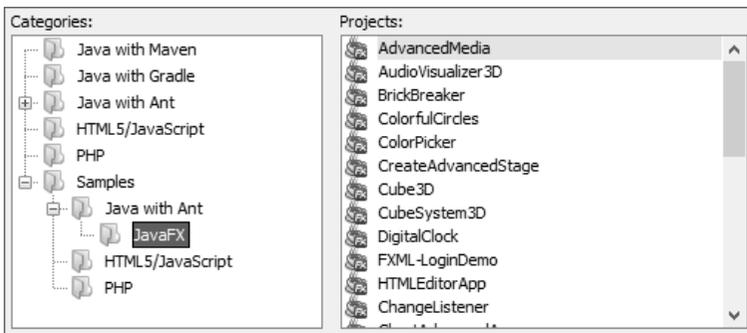
La troisième catégorie liste les projets Java avec *Another Neat Tool* (Ant). Ant est un outil de construction d'applications comme Maven ou Gradle. Il permet l'automatisation de tâches comme la compilation, la génération de documents, la création d'archives. Plus ancien que Maven, ses fonctionnalités sont moins nombreuses.



Les types de projets Java disponibles dans cette catégorie sont décrits dans les paragraphes précédents, il n'est pas nécessaire de les détailler. L'essentiel à retenir sur Ant est que, de la même manière que Maven ou Gradle, il permet la création d'applications Java basiques ou web.

Comme le montre l'image ci-dessus, l'utilisation de NetBeans pour des développements de projet HTML5/JavaScript et PHP est possible. Ces parties ne seront pas développées dans la suite de ce livre, consacré à Java avec NetBeans.

La dernière catégorie, Samples, donne accès à des exemples de projets JavaFX. Ils sont une base intéressante pour découvrir les différentes possibilités de création.



Les différents types de projets Java ont été introduits. La suite du chapitre a pour objet de présenter la création d'un projet Java basique, dont les modalités de développement et d'exécution sont les plus simples à aborder pour prendre en main NetBeans.