

Pierre Alexis et Hugues Bersini

1 étude de  
cas inspirée de  
Facebook !

# Apprendre la programmation web avec Python et Django

Principes et bonnes pratiques  
pour les sites web dynamiques

© Groupe Eyrolles, 2012,  
ISBN : 978-2-212-13499-5

EYROLLES

# Avant-propos

---

*Cet ouvrage se base sur Django, un outil de création de sites dynamiques écrit en Python, langage aujourd'hui très populaire pour apprendre à programmer. Ce livre enseigne la programmation web de façon aussi moderne que possible et s'adresse aussi bien aux enseignants qu'à leurs élèves. Il vous demande peu de pré-requis, mais reste ambitieux, de sorte que vous n'aurez pas à rougir devant les professionnels.*

## SOMMAIRE

- ▶ Résumer l'ambition de ce livre
- ▶ Resituer Django et Python dans l'évolution des technologies logicielles
- ▶ À qui s'adresse ce livre
- ▶ Plan de l'ouvrage

**CULTURE Du Web préhistorique au Web 2.0 moderne et « dynamique »**

Il y a vingt ans, on déambulait entre les pages du Web comme entre deux tableaux de maître dans un musée ; seule différence, on était assis face à son écran, souris à la main. Au fil des clics, les sites web s'affichaient : ils étaient déjà jolis et colorés, riches en information et éminemment utiles... mais inaltérables et rétifs à toute autre forme d'interférence que ce vagabondage de page en page.

Il était impossible pour l'internaute d'y écrire son numéro de carte de crédit pour acquérir une toile, de réserver sa place pour une visite du musée, de s'enregistrer comme utilisateur du site et d'épingler sur le sourire de Mona Lisa un commentaire ou des moustaches. Vous reveniez sur la même page quelques minutes plus tard ? Le contenu était inchangé. Tout était placidement statique ; en un mot, le Web fonctionnait pour l'essentiel dans une seule direction : des serveurs d'information vers les clients. Les utilisateurs étaient à la merci de ce que les développeurs côté serveur avaient choisi de leur dévoiler. Le serveur ne percevait que le clic d'entrée dans le site et se bornait à renvoyer en retour la page sollicitée ; il était sourd et indifférent à toute autre requête. Le Web d'alors, certes impressionnant et révolutionnaire, était statique.

James Gosling, le concepteur du langage de programmation Java (dont le succès devra beaucoup au Web), présenta en 1995 un nouveau concept de navigateur lors d'une conférence consacrée au Web. Il fit d'abord apparaître la représentation en 2D d'une molécule dans un navigateur web classique, laissant l'assistance de marbre. Quels ne furent la surprise et l'enthousiasme de ce même public lorsqu'à l'aide de sa souris, il se mit à bouger la molécule ! Elle réagissait aux stimuli de la souris, obéissait aux injonctions de l'internaute. Un programme Java, appelé « applet », s'exécutait dans le navigateur (Mosaic, à l'époque) rendant cette interaction effective. Le Web venait de gagner ses premiers galons de dynamisme et d'interactivité. Il en devenait chatouilleux.

En 2004, prenant conscience que l'utilisateur passait de spectateur passif à acteur essentiel du fonctionnement du Web, sursaturant ce dernier de ses vidéos de vacances, son savoir, ses copains de lycée ou ses prises de position politique, Tim O'Reilly décida de couronner le Web 2.0. Ce nouveau Web, aussi royal qu'il fût, devenait complètement assujéti au bon vouloir de ses sujets qui, au-delà des sempiternels clics de navigation, pouvaient dorénavant commenter, pérorer, refuser, voter, voler, acheter, vendre, négocier, réserver, prêter, jouer, écouter, visionner, projeter, et tant d'autres activités encore l'éloignant de l'écran de télévision pour celui de son PC. La confiscation du Web par cette multiplicité d'ego, choisissant d'en faire cette excroissance narcissique qu'il est devenu, fut considérée à ce point révolutionnaire que le célèbre hebdomadaire américain « Time » choisit d'élire *You* la personne de l'année 2006.

L'affichage d'un site web selon le bon vouloir de son créateur nécessite la maîtrise d'un langage de présentation de contenu appelé HTML (XHTML ou HTML5, vous comprendrez ces nuances par la suite), décrivant et paramétrant l'organisation de la page, la taille et le style des différents éléments de celle-ci. C'est le seul langage compris par les navigateurs, qui affichent la page de la manière précisée et transmise dans les instructions HTML.

Rendre la page « dynamique » requiert, de façon un peu plus exigeante, la maîtrise d'un langage de programmation comme il en existe malheureusement trop : JavaScript, C++, Java, .Net, Python, PHP, Ruby, SmallTalk, Eiffel, OCaml et tant d'autres. Ainsi, la mise en place d'un site web dynamique capable de se comporter comme un programme, réagissant comme il se doit aux sollicitations de son utilisateur, exige de mixer ces deux savoirs dans une proportion dépendant de l'attention accordée soit à son interactivité, soit à son apparence.

Heureusement, depuis l'apport pionnier des applets et des servlets Java, de multiples technologies logicielles sont apparues afin de combiner efficacement programmation et physionomie d'un site. Qui, dans le monde informatique, n'a entendu parler de PHP, JSP (dans la mouvance Java) ou ASP.Net (offre de Microsoft), qui structurent les milliards de sites interactifs disponibles aujourd'hui sur le Web ? Au-dessus de ces langages apparaissent de multiples boîtes à outils logicielles facilitant la conception des plates-formes web : Symfony et Zend pour PHP, Spring pour Java et, sujet de ce livre, Django pour Python.

Elles font partie de ces multiples solutions, relativement équivalentes (même si bien entendu leurs concepteurs s'en défendent avec vigueur), facilitant la prise en charge conjointe de la présentation de la page et de son interactivité.

Bien que ces différentes technologies puissent également s'implémenter côté client, cet ouvrage portera principalement sur celles qui s'exécutent côté serveur, les plus sollicitées. Le code exécuté côté serveur a pour mission de produire à destination du client une page répondant à ses desiderata. Pour ce faire, il est fréquemment nécessaire d'obtenir et d'exploiter des informations stockées dans une base de données relationnelle et de glisser ces dernières de manière judicieuse entre les instructions HTML5 qui se chargent de clairement les présenter.

## Pourquoi cet ouvrage ?

Programmation, HTML5/CSS3, bases de données, les technologies se multiplient et semblent exiger de la part des développeurs web un CV épais et noirci d'acronymes incompréhensibles pour la plupart des directeurs du personnel. La tentation est grande de les laisser sur le bas-côté du cursus universitaire, malgré leur omniprésence et une importance stratégique grandissante pour les entreprises.

C'est pour combler ce fossé entre la réalité mouvante des technologies web et le programme plus installé et stable du monde universitaire – d'aucuns diraient un peu poussiéreux et envahi de toiles... d'araignées – que Python et Django sont apparus et ont changé la donne.

## Le choix de Python et de Django

Depuis quelques années, grâce à Python et Django, il est devenu possible d'aborder en douceur le cocktail de technologies informatiques à la base de la programmation web. Ils ont pour eux la simplicité d'utilisation. Ainsi, le langage Python est très sou-

vent plébiscité pour sa facilité d'usage, sa rapidité d'acquisition, ce qui en fait un langage de prédilection pour l'apprentissage de la programmation. Son utilisation est élémentaire à démarrer et très interactive. Les instructions étant plus simples et plus intuitives, on parvient plus rapidement au résultat escompté. Bien que l'exécution de ces codes conduise au même message à l'écran, `print ("Hello World")` en Python est incontestablement plus simple à écrire mais aussi à comprendre que le fameux `public static void main (String[] args) {System.out.println "Hello World"}` rédigé avec stupeur par tous les débutants en programmation Java.

Langage de programmation disponible pour plusieurs plates-formes, Python est donc simple d'emploi : pas de typage explicite, instructions concises, structures de données d'un usage élémentaire (listes, dictionnaires et chaînes de caractères). Pour autant, il n'en préserve pas moins les fonctions essentielles de tout langage réputé puissant. Il semble, un peu mieux que les autres, réussir à concilier efficacité et simplicité, à se positionner entre ces deux acteurs exigeants que sont, d'une part, le processeur qui cherche l'efficacité et l'économie de mémoire et, d'autre part, le programmeur qui cherche la simplicité et la clarté d'utilisation.

#### EN PRATIQUE Des goûts et des couleurs en programmation...

Bien sûr, en matière de langage de programmation, ce genre de préférence tient pour beaucoup de l'acte de foi et il en va des guerres de langages comme de celles de religions ou de goûts culinaires. Toutefois, suffisamment d'enseignants ont fait de Python le premier langage à enseigner aux étudiants pour nous conforter dans notre choix.

Django, entièrement développé et basé sur le langage Python, fut à l'origine développé pour faciliter et accélérer la programmation de sites web dans un milieu journalistique (le journal Lawrence Journal-World publié quotidiennement dans la ville de Lawrence au Kansas), donc pas vraiment versé dans les technologies de développement logiciel. Les sites web en question exigeaient d'être développés et mis à jour à un rythme très élevé ; les outils devaient pour cela être faciles à prendre en main et les logiciels produits aisément réutilisables. Il était de surcroît préférable de séparer le métier d'ergonome de sites web, préoccupé pour l'essentiel par l'apparence et la physionomie du site, du métier de développeur, préoccupé par le fonctionnement optimal et sécurisé d'un code dont l'écriture devait rester très abordable et facilement modifiable.

Django est devenu cette boîte à outils logicielle (*framework* dans le jargon professionnel), ce canif suisse du développement web. Aujourd'hui, cette aventure se prolonge selon le modèle collaboratif de l'open source, impliquant des dizaines de milliers d'utilisateurs et de développeurs passionnés tout autour de la planète.

Ainsi, à l'origine de la programmation web, la technologie CGI à exécuter côté serveur donnait naissance à d'horribles codes comme celui ci-dessous (pour afficher une liste d'employés d'un service stockée dans une base de données relationnelle) :

### EXEMPLE 0.1 À l'ancienne mode CGI, tout est mélangé !

```
# -*- coding: utf-8 -*-

import sqlite3
import cgi

print "Content-Type: application/xhtml+xml; charset=utf-8\n"

print '<?xml version="1.0" encoding="UTF-8" ?>'
print '<!DOCTYPE html>'
print '<html xmlns="http://www.w3.org/1999/xhtml" lang="fr">'
print ' <head>'
print '   <title>eCarnet - Employés d'un service</title>'
print ' </head>'
print ' <body>'

form = cgi.FieldStorage()
service_id = str(form["service"].value)
db_connection = sqlite3.connect('database.sqlite3')
db_connection.row_factory = sqlite3.Row
cursor = db_connection.cursor()
cursor.execute("SELECT nom FROM service WHERE id=" + service_id)
row = cursor.fetchone()
service_nom = str(row['nom'])

print '   <h1>Employés du service « ' + service_nom + ' »</h1>'

cursor.execute("SELECT prenom, nom, tel_fixe FROM employe
                WHERE id_service=" + service_id)
rows = cursor.fetchall()

print '   <ol>'
for row in rows:
    print '       <li>' + row['prenom'] + ', ' + row['nom'] + ', '
        + row['tel_fixe'] + '</li>'
print '   </ol>'
print ' </body>'
print '</html>'

db_connection.close()
```

Sans qu'il soit nécessaire d'en saisir toutes les lignes, tout informaticien, même débutant, découvrira un mélange plutôt indigeste de trois réalités informatiques fonda-

mentales pourtant assez faciles à tenir distinctes, trois langages dont l'usage correct évite ce charabia pénible. Ces trois langages sont le SQL pour la gestion et l'interrogation des bases de données relationnelles, le Python pour la programmation et le HTML5 pour l'affichage et l'apparence des pages web.

C'est le pari réussi de Django de maintenir ces trois réalités informatiques suffisamment différenciables pour, par exemple, épargner à l'artiste des pages web les subtilités du stockage relationnel et les joies de la programmation. De surcroît, adopter une technologie alternative d'affichage ou de stockage de données devrait laisser suffisamment inaltérés les autres aspects du développement, contribuant ainsi à stabiliser l'intégralité du développement (souci essentiel d'une informatique dont les technologies ne cessent d'évoluer et de projets logiciels dont le cahier des charges se modifie en continu). Cette séparation découle d'une recette de conception bien connue des informaticiens et dénommée MVC (Modèle Vue Contrôleur), que nous aurons l'occasion de présenter largement par la suite, recette qui facilite à la fois la vie du développeur et celle de l'enseignant de cette pratique de développement.

## À qui s'adresse cet ouvrage ?

De par la simplicité de la syntaxe Python, les outils de développement web mis à disposition par Django, ainsi que le souci de ce dernier de maintenir les aspects esthétiques et les besoins du développement assez logiquement séparés, cet ouvrage présentant à la fois Python et Django vise pour l'essentiel trois types de lecteurs.

- Tout d'abord, les **enseignants du monde universitaire et des écoles d'ingénieurs**, tout spécialistes qu'ils soient en programmation, en bases de données relationnelles ou en technologies de présentation de contenu web (HTML5/CSS3, XHTML...) verront un intérêt à cet ouvrage. L'expérience de l'un des auteurs, Hugues Bersini, le montre. Il enseigne la programmation depuis 25 ans dans différentes facultés bruxelloises. Pour la partie plus appliquée et expérimentale de son enseignement, les étudiants sont ravis de mettre en œuvre le langage Python dans le cadre du développement web. Ils se plaisent à faire fonctionner une bibliothèque ou vidéothèque sur le Web, un site bancaire ou de commerce électronique. Ces projets leur donnent l'impression d'être en phase avec la réalité de l'informatique telle qu'elle se pratique aujourd'hui. Nombreux sont ceux qui, ayant découvert Python et Django, leur sont restés fidèles dans la vie professionnelle, dans leur entreprise ou leur start-up.
- Évidemment, les **étudiants** suivant cette même formation trouveront de quoi facilement concevoir une variété infinie d'autres sites web que celui présenté dans cet ouvrage. Au-delà de Django et Python, ils se sensibiliseront au métier de

développeur web, ses recettes, ses exigences, et les multiples solutions logicielles capables de leur mâcher la besogne.

- La troisième cible pouvant tirer profit de la lecture de ce livre comprend les **professionnels impliqués dans le développement web** en quête d'une entrée en douceur dans cet univers technologique somme toute assez exigeant. Même si leur environnement professionnel les contraint à l'une ou l'autre technologie web différente de Django (basée sur ASP.Net, Java/JSP, PHP, Ruby on Rail...), ce qu'ils découvriront à la lecture de ce livre, (les différentes technologies de programmation, stockage relationnel, modèles en HTML ou en CSS, le besoin d'une claire séparation entre elles, la réutilisation de fonctionnalités) leur demeurera d'une grande utilité. Et pourquoi pas, si cette liberté leur est offerte, ils pourront faire le choix de Django pour leur entreprise, plateforme logicielle légère et formidablement réactive ! Pierre Alexis, l'autre auteur de ce livre qui assiste Hugues Bersini dans son enseignement, peut en témoigner dans cet ouvrage commun !

## Le plan du cours

### RESSOURCES EN LIGNE Code source du projet Trombinoscoop

Le code source du site développé dans le livre est librement téléchargeable depuis la fiche du livre sur le site des éditions Eyrolles.

▶ [www.editions-eyrolles.com/livre/9782212134995](http://www.editions-eyrolles.com/livre/9782212134995)

Notre propos est découpé en deux grandes parties : nous présentons d'abord l'essentiel des aspects théoriques pour ensuite décrire en détail un projet concret.

La première partie a comme objectif de vous donner toutes les bases indispensables pour comprendre ce qu'est un site web et comment il fonctionne. Nous y présentons à la fois des notions théoriques (web dynamique, MVC, programmation objet, bases de données relationnelles) et les langages nécessaires à leur mise en œuvre (HTML5, CSS, JavaScript, SQL, Python).

- Le **premier chapitre** aborde les rudiments théoriques de la programmation web : HTTP, URL, notions de Web statique et dynamique, de serveurs web et de serveurs de bases de données.
- Le **chapitre 2** insiste sur la nécessité de séparer les tâches de programmation, de présentation et de structure/stockage des informations. Nous y présentons le modèle MVC et la programmation orientée objet. Nous expliquons en outre la notion de *framework* de développement et montrons en quoi l'utilisation d'un framework tel que Django facilite la tâche du développeur.

- Le **chapitre 3** balaie les notions élémentaires de programmation indispensables à l'utilisation de Python : variables, structures de contrôle, fonctions, rudiments de programmation objet. Ce chapitre suffira à comprendre les exemples de code Python présentés dans l'ouvrage et, de manière générale, la plupart des applications développées sous Django.

#### Aller plus loin

Ce livre n'est pas un manuel approfondi d'utilisation de Python. D'autres ouvrages existent pour cela, dont l'excellent livre de Gérard Swinnen :

 [Apprendre à Programmer avec Python](#), Gérard Swinnen, Eyrolles 2012

- Le **chapitre 4** s'intéresse aux langages de présentation des pages web, HTML5 et CSS, ainsi qu'à la manière de rendre les pages plus interactives et plus dynamiques (côté client cette fois) par l'utilisation du langage JavaScript.
- Le **chapitre 5** décrit le travail préliminaire à toute programmation web : l'élaboration d'éléments d'analyse qui permettent d'avoir une vue claire et réfléchie sur le projet qu'on désire développer. À cet effet, nous verrons ce que sont cas d'utilisation, wireframes et modèle de données. Pour appuyer notre propos, nous analyserons le projet développé dans la seconde partie de l'ouvrage. Nous fournissons également dans ce chapitre les notions essentielles concernant les bases de données relationnelles.
- Même si Django masque les échanges avec les bases de données, il est nécessaire d'acquérir les rudiments du langage SQL qui sert à dialoguer avec ces dernières. C'est l'objet du **chapitre 6**. Nous nous y appuyons sur les scripts CGI. Les limitations de cet ancêtre des technologies de programmation web démontrent la nécessité d'utiliser le MVC.

La deuxième partie met en application de façon concrète toutes ces notions et langages. Nous y décrivons étape par étape la construction d'un projet web complet inspiré de Facebook en nous appuyant sur le framework Django.

- Le **chapitre 7** s'intéresse à l'architecture et à l'orchestration du site. Il présente ce qu'on appelle les « vues » sous Django, ce qui correspond à l'aspect *Contrôle* du MVC (attention à la confusion dans les termes).
- Le **chapitre 8** sépare ce qui concerne la présentation du site. Les rendus HTML deviennent réutilisables. Cela se fait grâce aux « templates » de Django et correspond à l'aspect *Vue* du MVC.
- Le **chapitre 9** traite séparément le transfert de données entre l'utilisateur et le site, par l'intermédiaire des formulaires.

- Dans le **chapitre 10**, nous voyons comment décrire et implémenter les « modèles » avec Django (ce qui correspond à l'aspect *Modèle* du MVC).
- Élément indispensable aux sites web modernes, la gestion des sessions garantit une bonne authentification des utilisateurs et la persistance momentanée de leurs données. Cela fait l'objet du **chapitre 11**.
- Nous terminons l'élaboration de notre site exemple en construisant dans le **chapitre 12** les pages qui manquent encore.
- Le **chapitre 13** propose d'aller plus loin en ajoutant encore plus de dynamisme aux pages web grâce à la technologie Ajax.

Enfin, l'**annexe** décrit les étapes à suivre afin d'installer les outils de développement, dont Python et Django, quel que soit le système d'exploitation que vous utilisez (Windows, Mac OS X ou Linux). Les exemples d'application étant développés à partir de l'environnement Eclipse, le chapitre en décrit également l'installation, la configuration et la prise en main (intégrant l'emploi de Django).

## Remerciements

Pierre et Hugues adressent à Xavier Devos, Gaël Rabier et Jonathan Unikowski leurs remerciements les plus sincères pour leur suivi, leur relecture attentive et les nombreuses corrections et améliorations qui leur sont dues dans les pages qui vont suivre. Un immense merci aussi à l'équipe éditoriale d'Eyrolles : Muriel Shan Sei Fan pour, tout d'abord, y croire dur comme fer, et nous communiquer l'enthousiasme, les repères et les balises essentiels à la bonne conduite d'un tel projet, Laurène Gibaud et Anne Bougnoux pour le formidable travail de remaniement et Gaël Thomas... pour avoir transformé l'essai en une impeccable mise en forme.

# Table des matières

---

<b>Avant-propos .....</b>	<b>V</b>
Pourquoi cet ouvrage ? .....	VII
Le choix de Python et de Django .....	VII
À qui s'adresse cet ouvrage ? .....	X
Le plan du cours .....	XI
Remerciements .....	XIII

## PREMIÈRE PARTIE

### **Les notions essentielles .....**

**1**

#### CHAPITRE 1

#### **Comment fonctionne un site web ?.....**

**3**

Qu'est-ce que le Web ? .....	4
Fonctionnement d'un site web statique .....	5
Le protocole HTTP .....	7
L'URL, adresse d'une page web .....	8
Le serveur web : à la fois ordinateur et logiciel .....	9
Des sites web qui se mettent à jour tout seuls .....	9
Fonctionnement d'un site web dynamique .....	11
Utilisation d'une base de données .....	12
Passage de paramètres à une page web .....	13
Ai-je bien compris ? .....	14

#### CHAPITRE 2

#### **Programmation orientée objet et framework MVC.....**

**15**

Des programmes modulaires .....	16
Une écriture simplifiée .....	16
Des modules réutilisables .....	16
Un travail d'équipe facilité et plus efficace .....	16
Les langages orientés objet .....	17
Les cas d'utilisation (use cases) et le MVC .....	18

Le principe de la séparation modèle-vue-contrôleur (MVC) .....	18
Le diagramme des cas d'utilisation (use cases) .....	20
Correspondances entre MVC et cas d'utilisation .....	21
Django et le MVC .....	23
<b>Le framework Django</b> .....	23
<b>Ai-je bien compris ?</b> .....	24

## CHAPITRE 3

### **Rappels sur le langage Python ..... 25**

<b>Qualités et défauts de Python</b> .....	26
Qualité : la simplicité .....	26
Défaut : la simplicité ! .....	27
<b>Les bases : variables et mémoire centrale</b> .....	28
Déclaration et initialisation des variables .....	28
Type des variables .....	29
Modification et transtypage des variables .....	29
Copie de variables .....	30
Se renseigner sur une variable .....	30
<b>Quelques opérations sur les types simples</b> .....	31
Le type int .....	31
Le type float .....	32
Le type string .....	33
<b>Les types composites : listes et dictionnaires</b> .....	34
Les listes .....	35
Les dictionnaires .....	36
<b>Les instructions de contrôle</b> .....	37
Les instructions conditionnelles .....	37
<i>Aiguillage à deux directions : instruction if (...) else</i> .....	37
<i>Aiguillage à plus de deux directions : instruction if (...) elif (...) else</i> .....	39
Les boucles .....	40
<i>La boucle while</i> .....	40
<i>La boucle for (...) in</i> .....	41
<b>Les fonctions</b> .....	43
Les variables locales et les variables globales .....	44
<b>La programmation objet</b> .....	45
Les classes et les objets .....	45
L'association entre classes .....	47
<i>Héritage et polymorphisme</i> .....	51
<b>Import et from : accès aux bibliothèques Python</b> .....	54
<b>Ai-je bien compris ?</b> .....	55

## CHAPITRE 4

**Rappels sur HTML5, CSS et JavaScript..... 57**

Structurer les pages web avec HTML5 .....	59
Le concept de « balises » .....	60
Structure d'un document HTML .....	63
<i>L'encodage de la page</i> .....	65
Quelques éléments HTML .....	66
<i>Principaux éléments de structuration HTML</i> .....	66
<i>Éléments de structuration annexes : en-têtes, pieds de page et sections</i> .....	68
<i>Les liens hypertextes</i> .....	70
<i>Les listes</i> .....	71
<i>Les images</i> .....	72
<i>Mise en évidence du texte</i> .....	73
<i>Les formulaires</i> .....	75
<i>Autres éléments HTML</i> .....	78
Mettre en forme avec les feuilles de styles CSS .....	78
Les propriétés CSS .....	79
Les sélecteurs CSS .....	79
<i>Sélectionner toutes les balises de même nom</i> .....	80
<i>Sélectionner un élément particulier : id en HTML et # en CSS</i> .....	80
<i>Sélectionner quelques éléments de même nature : class en HTML et . en CSS</i> ...	81
<i>Appliquer une propriété seulement quand l'élément est dans un état donné</i> ...	81
<i>Combiner les sélecteurs</i> .....	82
<i>Sélectionner tous les éléments</i> .....	83
Lier CSS et HTML .....	83
<i>Placer le code CSS dans les balises HTML</i> .....	83
<i>Placer le code CSS dans l'en-tête du fichier HTML</i> .....	84
<i>Placer le code CSS dans un fichier séparé</i> .....	85
Dimensions des éléments en CSS .....	86
Imbrication des éléments (boîtes) .....	87
Positionnement par défaut des éléments en CSS .....	88
Sortir certains éléments du flux .....	89
Application à un exemple concret plus complet .....	91
Dynamiser les pages web « à la volée » avec JavaScript .....	96
Les événements .....	96
Langage de programmation de scripts .....	96
Un premier exemple de DHTML .....	97
jQuery et les frameworks JavaScript .....	99
Ai-je bien compris ? .....	101

## CHAPITRE 5

**Mise en application : un site web inspiré de Facebook ..... 103**

Notre site web : Trombinoscoop .....	104
Les cas d'utilisation .....	105
Maquette du site : les wireframes .....	106
L'écran d'authentification .....	107
L'écran de création d'un compte .....	108
L'écran d'accueil .....	108
L'écran de modification du profil .....	108
L'écran d'affichage d'un profil .....	108
L'écran d'ajout d'un ami .....	108
Scénario complet du site .....	112
Modèle de données et petit rappel sur les bases de données relationnelles .....	113
Clé primaire et clé étrangère .....	114
<i>Relation 1-n</i> .....	114
<i>Relation 1-1</i> .....	116
<i>Relation n-n</i> .....	117
La problématique de la mise en correspondance relationnel/objet .....	118
<i>Avec Django</i> .....	119
Retour au modèle de données de Trombinoscoop : son diagramme de classes ..	119
<i>Des personnes : étudiants et employés</i> .....	120
<i>... qui travaillent ou étudient à l'université</i> .....	120
<i>... et qui échangent des messages avec des amis</i> .....	122
Ai-je bien compris ? .....	123

## CHAPITRE 6

**Premier contact avec les bases relationnelles et SQL****à partir d'un exemple en CGI..... 125**

Analyse du carnet d'adresses : des cas d'utilisation au modèle de données .....	127
Trois cas d'utilisation simples .....	127
Maquette des écrans (wireframes) .....	127
Le modèle de données du carnet .....	127
Création de la base de données avec SQLite .....	129
Accès à la base de données via SQL .....	131
Syntaxe des requêtes SQL les plus courantes .....	131
Quelques exemples liés à notre base de données .....	132
Réalisation du carnet d'adresses avec SQL et CGI .....	133
Lancement d'un serveur web Python .....	133
L'écran d'accueil de l'ULB .....	135
La page principale du carnet d'adresses .....	136

La liste des employés d'un service .....	140
Ajout d'un employé .....	143
<b>CGI : ce qu'il ne faut plus faire</b> .....	145
<b>Ai-je bien compris ?</b> .....	145

## DEUXIÈME PARTIE

# Mise en application avec Django ..... 147

### CHAPITRE 7

## Les vues Django : orchestration et architecture ..... 149

Utilité des vues .....	151
Le fichier urls.py .....	152
Le fichier views.py .....	153
<b>Enfin ! Notre première page web en Django</b> .....	155
Lancement de l'environnement Eclipse et création du projet .....	155
Le fichier urls.py .....	158
Le fichier views.py .....	158
Importation de la fonction dans urls.py .....	160
Configuration de Django pour qu'il écrive du XHTML 5 .....	160
Test de notre ébauche de site .....	161
<b>Bel effort, mais...</b> .....	163
<b>Ai-je bien compris ?</b> .....	163

### CHAPITRE 8

## Les templates Django : séparation et réutilisation des rendus HTML ..... 165

Principe des templates .....	166
Notre premier template .....	167
Dynamisons ce premier template .....	170
Le langage des templates .....	172
Les variables .....	172
Formatage des variables .....	172
Sauts conditionnels et boucles .....	173
Héritage et réutilisation de templates .....	174
<b>Et si on avançait dans la réalisation de Trombinoscoop ?</b> .....	176
<b>Amélioration visuelle de la page de login</b> .....	180
<b>Ai-je bien compris ?</b> .....	184

## CHAPITRE 9

**Les formulaires Django ..... 185**

Patron courant de gestion des formulaires .....	186
L'objet request .....	187
Traitement du formulaire de login .....	188
Ajout du formulaire dans la vue .....	188
Gestion du message d'erreur .....	189
Présentation du message d'erreur .....	190
La bibliothèque forms de Django .....	190
Création d'un formulaire avec la bibliothèque forms .....	191
Intégration du formulaire dans la vue et le template .....	192
Validation du formulaire .....	194
Présentation des messages d'erreur .....	196
Validation de l'adresse de courriel et du mot de passe .....	197
Faut-il se contenter d'un seul visiteur autorisé ? .....	199
Ai-je bien compris ? .....	199

## CHAPITRE 10

**Les modèles Django ..... 201**

Les modèles Django .....	202
Création d'un premier modèle .....	202
Le modèle Personne .....	203
Configuration .....	204
Création de la base de données et du compte administrateur (superutilisateur) .....	205
Création des autres modèles et de leurs liens .....	207
Le modèle Message : relation 1-n .....	207
La relation « ami » : relation n-n .....	208
Les modèles simples Faculté, Campus, Fonction et Coursus .....	209
Les modèles Employé et Étudiant : héritage .....	209
Le lien entre Faculté et Personne : relation 1-n .....	210
Régénération de la base de données .....	211
Utilisation des modèles .....	211
Création et modification d'un enregistrement .....	211
Récupération de plusieurs enregistrements .....	212
Tri des données .....	212
Récupération d'un enregistrement unique .....	212
Suppression d'enregistrements .....	213
Accès à des objets « liés » .....	213
Remplissage de la base de données .....	214
Configuration de l'interface d'administration des bases de données .....	214

Gestion de la base de données avec l'interface d'administration .....	216
<b>Authentification utilisant la base de données .....</b>	<b>219</b>
<b>Les ModelForm .....</b>	<b>221</b>
Création du formulaire Étudiant dans le fichier forms.py .....	221
Création de l'URL et de la vue de création de compte .....	222
Création du template de création de compte .....	223
Un peu de mise en forme .....	223
<b>Finalisation de la page de création de compte .....</b>	<b>225</b>
Création de deux formulaires : un pour les étudiants et un pour les employés ..	225
Gestion des deux formulaires dans la vue .....	226
Gestion des deux formulaires dans le template .....	227
Un peu de dynamisme .....	229
<b>Ai-je bien compris ? .....</b>	<b>230</b>

## CHAPITRE 11

### **Comprendre et utiliser les sessions .....** **231**

À quoi servent les sessions .....	232
Les sessions selon Django .....	233
<b>Utilisation d'une session .....</b>	<b>234</b>
Configuration .....	234
Maniement d'une variable de session .....	235
Enregistrement de l'utilisateur authentifié .....	235
Vérification que l'utilisateur est bien authentifié .....	236
Utilisation des données de la session .....	237
Que trouve-t-on dans le cookie ? .....	238
Que trouve-t-on dans la session ? .....	238
Protection des pages privées .....	238
<b>Amélioration de notre page d'accueil .....</b>	<b>239</b>
Personnalisation de la bannière .....	240
Division du corps de la page .....	243
Liste des messages .....	245
<i>Récupération des messages de la liste .....</i>	<i>245</i>
<i>Présentation de la liste des messages .....</i>	<i>247</i>
Liste des amis .....	248
Publication d'un message à destination de ses amis .....	250
<b>Ai-je bien compris ? .....</b>	<b>251</b>

## CHAPITRE 12

### **En finir avec Trombinoscoop .....** **253**

La page d'ajout d'un ami .....	254
--------------------------------	-----

Ajout d'un formulaire dans forms.py . . . . .	254
Création de la vue add_friend . . . . .	255
Création du template add_friend.html . . . . .	256
Ajout d'une URL dans urls.py . . . . .	256
Ajout du lien dans la page d'accueil . . . . .	256
<b>La page de visualisation d'un profil . . . . .</b>	<b>257</b>
Création du template show_profile.html . . . . .	258
Création de la vue show_profile . . . . .	259
Ajout de l'URL dans urls.py . . . . .	260
Amélioration de la présentation dans style.css . . . . .	260
Ajout des liens dans la page d'accueil . . . . .	260
<b>La page de modification d'un profil . . . . .</b>	<b>262</b>
Création du template modify_profile.html . . . . .	262
Création de la vue modify_profile . . . . .	263
Ajout de l'URL dans urls.py . . . . .	264
Ajout des liens dans la page d'accueil . . . . .	264
<b>Ai-je bien compris ? . . . . .</b>	<b>265</b>

## CHAPITRE 13

### **Des sites web encore plus dynamiques avec Ajax..... 267**

Exemple de l'interactivité attendue entre client et serveur . . . . .	268
Validation de la création d'un compte avec Ajax . . . . .	271
Ajout d'une URL pour la requête Ajax . . . . .	272
Vue traitant la requête Ajax . . . . .	272
Ajout du code JavaScript . . . . .	274
<i>Détecter que l'utilisateur a bien terminé de remplir le champ courriel . . . . .</i>	<i>274</i>
<i>Validation du courriel saisi . . . . .</i>	<i>276</i>
<i>Insérer la liste des erreurs au-dessus du champ « fautif » . . . . .</i>	<i>277</i>
Ajout d'un ami via Ajax . . . . .	279
Ajout d'un champ texte et d'un lien . . . . .	279
Ajout de l'URL et de la vue . . . . .	280
Création du JavaScript d'ajout d'un ami . . . . .	281
Insertion du HTML dans la page web . . . . .	283
<b>Conclusions . . . . .</b>	<b>284</b>
<b>Ai-je bien compris ? . . . . .</b>	<b>285</b>

## ANNEXE A

### **Installation de l'environnement de développement ..... 287**

Que faut-il installer ? . . . . .	288
Python . . . . .	288

Django .....	289
Eclipse .....	289
En résumé .....	290
<b>Installation de Python</b> .....	292
Pour Windows .....	292
Pour Mac OS X .....	295
Vérification de l'installation .....	296
<b>Installation de Django</b> .....	297
Pour Windows .....	297
Pour Mac OS X .....	299
Pour Ubuntu .....	300
Vérification de l'installation .....	301
<b>Installation de Java</b> .....	302
Pour Windows et Mac OS X .....	302
Pour Ubuntu .....	302
<b>Installation d'Eclipse</b> .....	303
<b>Installation du plug-in Eclipse PyDev</b> .....	305
<b>Installation de Web Developer Tools pour Eclipse</b> .....	310
<b>Premier projet de test</b> .....	311
<b>Ai-je bien compris ?</b> .....	314

<b>Index</b> .....	<b>315</b>
--------------------	------------

# PREMIÈRE PARTIE

## Les notions essentielles

Cette première partie vise à vous donner toutes les clés pour comprendre ce qu'est un site web, comment il se construit et comment il fonctionne :

- protocole HTTP, URL, notions de web statique et dynamique, serveurs ;
- programmation orientée objet, modèle MVC, notion de framework ;
- éléments de base du langage de programmation Python ;
- structuration des pages web avec HTML5, présentation avec les feuilles de styles CSS, interactivité côté client avec JavaScript ;
- élaboration du modèle de données : cas d'utilisation (use cases), schémas de présentation (wireframes), bases de données relationnelles ;
- éléments de base du langage d'interrogation de bases de données SQL, utilisation par le biais de l'ancienne technologie CGI, discussion des limitations de cette dernière.

# 1

## Comment fonctionne un site web ?

---

*Ce chapitre rappelle les bases du Web, ses origines statiques et son passage graduel vers plus de dynamisme avec, notamment, l'exploitation d'une base de données côté serveur, dont les mises à jour se répercutent côté client. Sont également expliquées les notions d'URL, de protocole HTTP, de serveur web et le passage de paramètres du client vers le serveur.*

### SOMMAIRE

- ▶ Petite histoire du Web
- ▶ Du Web statique au Web dynamique
- ▶ URL, protocole HTTP et serveur web
- ▶ Passage de paramètres vers le serveur

# 2

## Programmation orientée objet et framework MVC

---

*Ce chapitre a pour mission d'expliquer la conception logicielle respectueuse du découpage MVC et sa version sous Django. Nous exposons d'abord les principes de la programmation orientée objet. Ensuite, nous verrons comment la mise à disposition d'un framework s'inscrit dans cette logique.*

### SOMMAIRE

- ▶ Explication de la conception respectant le découpage MVC
- ▶ Mise en place du MVC sous Django
- ▶ Explication de la notion de framework

# 3

## Rappels sur le langage Python

---

*Ce chapitre a pour mission de présenter le langage de programmation Python, sur lequel repose le projet Django. Les règles syntaxiques brièvement décrites ici devraient vous suffire pour comprendre les différentes applications de Django que nous verrons par la suite, ainsi que pour maîtriser les aspects « contrôle » de votre projet.*

### SOMMAIRE

- ▶ Introduction au langage de programmation Python
- ▶ Les aspects procéduraux du langage
- ▶ Python et l'orienté objet

# 4

## Rappels sur HTML5, CSS et JavaScript

---

*Ce chapitre va s'intéresser aux langages de présentation des pages web, HTML5 et CSS, ainsi qu'à la manière de rendre les pages plus interactives et plus dynamiques côté client, par l'utilisation du JavaScript. Ces trois technologies sont devenues incontournables, quelle que soit la plate-forme de développement web choisie.*

### SOMMAIRE

- ▶ Synthèse rapide des technologies HTML5, CSS et JavaScript
- ▶ HTML5, CSS, JavaScript : pièces maîtresses du Web communes à tous les environnements de développement web

# 5

## Mise en application : un site web inspiré de Facebook

---

*Ce chapitre présente le projet de développement qui servira de base pour tous les autres chapitres : Trombinoscoop. Tout projet web démarre par la rédaction de ses cas d'utilisation (use cases) et la conception de ses wireframes, et d'une base de données relationnelle côté serveur. Quelques rappels théoriques sur le modèle relationnel se révèlent indispensables. Django exige que le modèle de données soit réalisé via un diagramme de classes, donc dans une perspective purement orientée objet. Par conséquent, quelques lignes sont consacrées à l'épineux problème de la mise en correspondance entre le modèle relationnel et le modèle objet.*

### SOMMAIRE

- ▶ Trombinoscoop : le projet auquel ce livre va se consacrer
- ▶ Cas d'utilisation et maquettage (*wireframes*)
- ▶ Rappels théoriques sur les bases de données relationnelles et différence avec la modélisation orientée objet
- ▶ Diagramme de classes de Trombinoscoop

# 6

## Premier contact avec les bases relationnelles et SQL à partir d'un exemple en CGI

---

*Le site web doit pouvoir interroger et modifier la base relationnelle où sont stockées toutes les données du modèle. Il lui faut pour cela un langage particulier : le SQL. Les notions de SQL dont nous aurons besoin par la suite, mais qui sont le plus souvent cachées par le framework Django, seront exposées dans un contexte web réalisé en CGI. Cette technologie n'est plus de mise aujourd'hui pour le développement web côté serveur, mais à titre pédagogique, ce chapitre nous permettra de l'évoquer comme un parfait exemple d'un mélange de genres à éviter.*

L'installation des outils de développement est expliquée en annexe A

### SOMMAIRE

- ▶ Aperçu des requêtes SQL côté serveur
- ▶ Introduction à CGI : un condensé des mauvaises pratiques du Web

## DEUXIÈME PARTIE

# Mise en application avec Django

Dans cette deuxième partie, nous allons construire notre site Trombinoscoop en nous appuyant sur le framework Django. Nous détaillerons chacune des étapes d'élaboration d'un site web :

- « vues » Django (correspondant à l'aspect Contrôle du MVC) ;
- « templates » Django (correspondant à l'aspect Vue du MVC) ;
- formulaires ;
- « modèles » Django (correspondant à l'aspect Modèle du MVC) ;
- gestion des sessions ;
- suppléments en Ajax.

# 7

## Les vues Django : orchestration et architecture

---

*Ce chapitre va nous lancer dans l'utilisation du framework Django et nous permettre d'en découvrir les premiers aspects : URL et « Views ». MVC, la recette de conception fondamentale de tout développement web mise en place par Django, commencera à être mieux comprise. Un premier projet Django sera réalisé se bornant à renvoyer une page HTML très élémentaire, un peu comme on le ferait pour une application statique toute basique.*

### SOMMAIRE

- ▶ Le modèle MVC selon Django
- ▶ Utilité des vues
- ▶ Définition des URL
- ▶ Définition des méthodes « vues »
- ▶ Création d'un projet Django dans Eclipse
- ▶ Réalisation de notre première page web
- ▶ Démarrage et test d'un projet dans Eclipse

# 8

## Les templates Django : séparation et réutilisation des rendus HTML

---

*Tout en continuant le développement de notre projet « Trombinoscoop », ce chapitre aide à comprendre comment la mise en place des « templates » Django tient les instructions HTML (dédiées à la présentation du site web) séparées des instructions Python (plus centrées sur son fonctionnement). Le rôle et l'utilisation des templates constituent l'essentiel de ce qui va suivre.*

### SOMMAIRE

- ▶ Introduction aux templates de Django qui séparent au mieux le HTML du code Python
- ▶ Description de tout ce que l'on peut faire et produire à partir de ces templates
- ▶ La suite de notre projet Trombinoscoop

# 9

## Les formulaires Django

---

*L'objectif de ce chapitre est d'étudier comment gérer les formulaires HTML avec Django. Nous nous poserons les questions suivantes :*

- *Qu'allons-nous mettre comme URL dans l'attribut `action` ?*
- *Comment allons-nous gérer dans Django la réception des données du formulaire ?*
- *Comment gérer des données erronées du formulaire ?*
- *Comment réagir lorsque toutes les données du formulaire sont correctes ?*

### SOMMAIRE

- ▶ Comment réaliser les formulaires en Django ?
- ▶ Installation de nos premiers formulaires dans Trombinoscoop

# 10

## Les modèles Django

---

*L'objectif de ce chapitre est d'étudier comment mettre en place une base de données à l'aide des modèles Django et comment interagir avec elle. Plus précisément, nous allons apprendre à :*

- *décrire les données à enregistrer dans la base de données ;*
- *créer la base de données ;*
- *insérer des données dans la base de données ;*
- *accéder aux données pour ensuite les utiliser dans notre code.*

### SOMMAIRE

- ▶ Réalisation de la base de données à la manière de Django
- ▶ Introduction aux modèles Django, version objet de la base de données sous-jacente
- ▶ Suite du projet Trombinoscoop

# Comprendre et utiliser les sessions

---

*Ce chapitre va nous familiariser avec les sessions, qui servent à sauvegarder un contexte global d'interaction entre deux visites sur le serveur web. Les sessions fonctionnent grâce aux célèbres et tant décriés cookies qui mémorisent les accès au serveur. Django a sa propre manière de répartir cette mémoire entre le client et la base de données côté serveur. Le projet Trombinoscoop sera enrichi par exemple par la mémorisation de l'identité du visiteur du site.*

## **EN PRATIQUE** Les applications Django travaillent pour vous

Notez que l'application `django.contrib.auth` peut faire pour vous la grande partie de ce que nous expliquons dans ce chapitre, mais nous choisissons bien sûr de nous en passer, pour l'intérêt de l'exercice.

## **SOMMAIRE**

- ▶ Explication de la notion de session
- ▶ Explication et utilisation des cookies avec Django
- ▶ Prise en charge des sessions dans Trombinoscoop

# 12

## En finir avec Trombinoscoop

---

*Finalisons notre projet de développement web en réalisant les derniers wireframes. Nous appliquerons pour cela tout ce que nous avons appris dans les précédents chapitres.*

### SOMMAIRE

- ▶ Finition de Trombinoscoop

# 13

## Des sites web encore plus dynamiques avec Ajax

---

*Ce chapitre va se pencher sur la technologie Ajax qui permet au client de solliciter le serveur sans quitter la page actuellement active de son côté. Le résultat du serveur viendra juste s'insérer dans la page client courante sans qu'il faille entièrement la remplacer par une autre. C'est du JavaScript s'exécutant côté client qui va se charger de réaliser la connexion avec le serveur, la récupération des informations et leur insertion dans la page client courante. L'interaction sera dite asynchrone car, lorsque le client envoie la requête, il n'est pas contraint d'attendre la réponse du serveur pour poursuivre sa tâche.*

### SOMMAIRE

- ▶ Présentation de la technologie Ajax
- ▶ Discussion sur la répartition de la partie « programme » du site web entre le client et le serveur
- ▶ Mise en place de solutions Ajax pour enrichir le dynamisme de Trombinoscoop
- ▶ Point final de Trombinoscoop



# Installation de l'environnement de développement

---

*Cette annexe décrit étape par étape l'installation des différents éléments logiciels nécessaires à la réalisation du projet Trombinoscope.*

## **SOMMAIRE**

- ▶ Annexe décrivant l'installation des différents outils nécessaires au développement web sous Django : Python, Java, Django, Eclipse et plug-ins permettant l'exécution de code Python dans l'environnement Eclipse.
- ▶ Cette installation est décrite pour les trois systèmes d'exploitation les plus répandus : Windows, Mac OS et Linux.

Avant de se lancer dans la programmation Python/Django, il est nécessaire de configurer son environnement de travail. L'objectif de ce chapitre est d'expliquer chacune des étapes qui permettront d'installer et de configurer les divers outils nécessaires pour se constituer un environnement de développement optimal. Les explications sont déclinées pour trois systèmes d'exploitation :

- Microsoft Windows 7 ;
- Mac OS X Mountain Lion ;
- Ubuntu 12.

Nous espérons couvrir la grande majorité des systèmes d'exploitation utilisés aujourd'hui, sachant que les explications sont facilement adaptables pour leurs variantes. Par exemple, la procédure décrite pour Mac OS X Mountain Lion reste valable pour les versions inférieures Lion et Snow Leopard.

## Que faut-il installer ?

### Python

Le premier outil à installer, et dont on ne pourra pas se passer, est le langage de programmation Python qui nous permettra de compiler et d'exécuter des programmes écrits dans ce langage.

Python est accompagné d'une importante *bibliothèque standard* offrant de nombreux modules pré-écrits (types de données complexes, cryptographie, traitement d'images, traitement audio, envoi de courriels, etc.). C'est par l'instruction `import` qu'on appelle ces différents utilitaires dans les codes qui en font usage.

Il existe plusieurs implémentations du langage Python et de sa bibliothèque standard. Nous utiliserons l'implémentation « traditionnelle » de référence, proposée sur le site web [python.org](http://python.org) et disponible pour de nombreuses plates-formes (Windows, Linux, etc.).

Python, comme tout langage de programmation qui se respecte, est en constante évolution ; il en existe plusieurs versions. La dernière version majeure est la 3 et mérite qu'on s'y attarde un peu. Elle a eu notamment pour objectif de simplifier le langage en lui retirant certaines constructions jugées redondantes. La compatibilité ascendante a été brisée. En d'autres mots, et à la grande déconvenue de nombreux programmeurs, le compilateur Python 3 n'est plus capable de compiler tous les programmes écrits en Python 2. Or, Django a été écrit à la base pour Python 2 et sa migration vers Python 3 n'a pas encore abouti. Nous installerons donc la version 2.7 de Python, qui est la dernière précédant la version 3. Toutefois rassurez-vous, l'adaptation de Django à la der-

nière version de Python, qui se fera bien un jour, ne devrait pas modifier pour l'essentiel notre enseignement et le projet que nous vous proposons.

## Django

Une fois Python installé, il faut l'enrichir avec le framework Django, qui n'est pas prévu de base dans la bibliothèque standard Python. Ce framework se télécharge sur le site officiel de Django.

Un des avantages de Django est l'intégration d'un serveur web léger. On n'est donc pas obligé d'installer un serveur web tiers, tel Apache. Bien entendu, si le serveur web léger inclus dans Django suffit amplement pour tester son site durant la phase de développement, il n'en sera pas de même lorsqu'on ouvrira son site au public. Un serveur web tiers plus robuste sera nécessaire, car il permettra de mieux supporter un nombre élevé de visites sur le site et permettra une configuration plus fine des utilitaires web.

## Eclipse

Python et Django suffiraient à débiter le développement de notre premier site web. À l'aide d'un simple éditeur de texte et d'une interface en ligne de commande, nous pourrions écrire notre premier code et le lancer. Ce serait néanmoins laborieux et peu convivial. C'est pourquoi nous allons installer un *environnement de développement intégré* ou IDE (*Integrated Development Environment*).

### DÉFINITION IDE

Un environnement de développement intégré est un ensemble d'outils facilitant et rendant plus convivial le développement d'applications.

Généralement, les IDE offrent au moins les utilitaires suivants, plutôt précieux :

- un éditeur de texte capable de colorer le code, de détecter les erreurs de syntaxe en ligne ou d'aider à la saisie de code en affichant toutes les constructions possibles ;
- une interface graphique conviviale pour simplifier la compilation d'un programme ou le lancement de l'application ;
- un débogueur graphique permettant d'exécuter pas à pas un programme et d'observer son état à tout instant (valeurs des variables, position dans le code, etc.).

Notre choix d'environnement de développement intégré s'est porté sur Eclipse car il s'agit d'un environnement très populaire et complet, capable de gérer plusieurs langages de programmation, dont Python et Django.

Eclipse est un IDE écrit en Java. C'est aussi à ce jour l'environnement de développement Java le plus utilisé, sachant que Java est actuellement le langage de programmation le plus répandu et le plus enseigné. Le moteur d'exécution Java, qui permet de lancer des programmes écrits dans ce langage, devra donc faire partie de la panoplie d'outils à installer, sans quoi Eclipse ne pourra pas être lancé.

L'installation de base d'Eclipse ne contient pas les modules permettant de gérer le langage Python et le framework Django. Ces modules, regroupés dans le plug-in nommé PyDev, doivent être ajoutés manuellement par la suite, ce que nous ferons également.

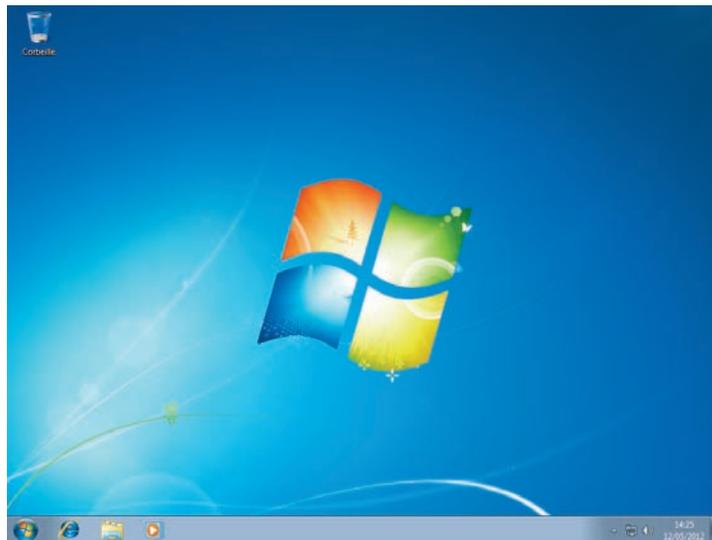
## En résumé

La mise en place de notre environnement de développement passera par l'installation successive de ces cinq composants :

- le langage Python, dans sa version 2.7 ;
- le framework Django ;
- le moteur d'exécution Java (*Java Runtime Environment*) ;
- l'IDE Eclipse ;
- le plug-in Eclipse PyDev.

Les sections suivantes sont consacrées à l'installation de ces éléments sous différents systèmes d'exploitation. Vous pouvez bien entendu ne lire que les sections qui correspondent à votre environnement.

**Figure A-1**  
Windows 7  
professionnel 32 bits





## Installation de Python

Commençons par une bonne nouvelle : sous Ubuntu, Python est déjà pré-installé. Pour Windows et Mac OS X, la première étape consiste à installer l'implémentation officielle *CPython*.

### EN PRATIQUE « Vieilles » versions de Python sous Mac OS X

Mac OS X est livré en standard avec Python. Malheureusement, celui-ci n'est mis à jour qu'à chaque sortie d'une nouvelle version du système, soit environ tous les deux ans. On se retrouve souvent avec une version de Python largement dépassée. Il est donc indispensable d'installer la version qui nous intéresse à côté de celle existante et qui cohabitera en très bon voisinage avec cette dernière.

### EN PRATIQUE Version de Python

La version à installer est la 2.7 et non la 3, cette dernière n'étant pas compatible avec Django. Nous recommandons d'installer la dernière version de la lignée 2.7. À l'heure où ces lignes sont écrites, il s'agit de la version 2.7.3.

Pour la télécharger, il faut se rendre dans la section [Download](#) de [www.python.org](http://www.python.org). Différents paquets sont disponibles. Nous avons choisi :

- pour Windows, le paquet [Windows x86 MSI Installer](#), comprenant un installateur destiné aux Windows 32 bits ;
- pour Mac OS X, le paquet [Mac OS X 64-bit/32-bit x86-64/i386 Installer](#). C'est en réalité une image disque [DMG](#) qu'il suffit de monter en l'ouvrant.

► [www.python.org](http://www.python.org)

## Pour Windows

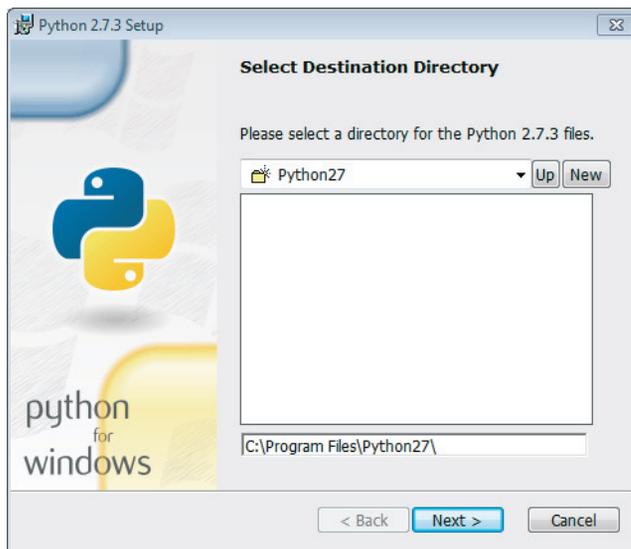
Une fois le paquet téléchargé, il suffit de l'exécuter et de suivre les étapes de l'assistant d'installation. À la première étape, l'assistant vous demande si vous désirez installer Python pour tous les utilisateurs de l'ordinateur ou juste pour l'utilisateur courant ; il est recommandé de choisir [Install for all users](#).

L'assistant demande ensuite de choisir un emplacement pour l'installation. Afin de respecter les standards Windows, nous recommandons d'installer Python dans [Program Files](#) et non à la racine du disque système, comme proposé par défaut par l'installateur.

**Figure A-4**  
Première étape de l'assistant  
d'installation

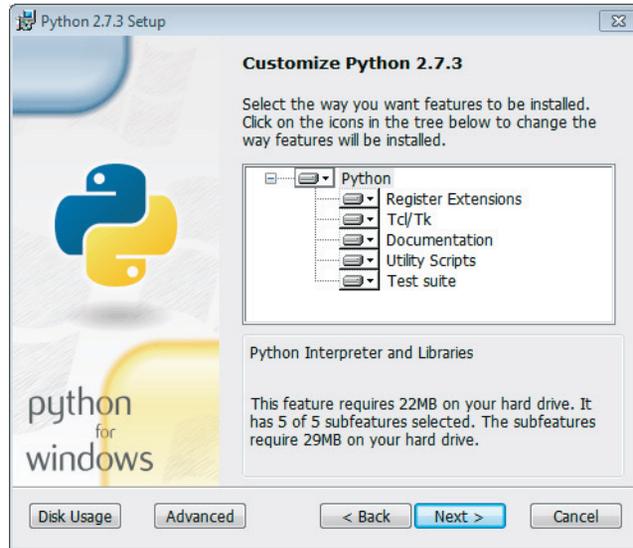


**Figure A-5**  
Deuxième étape de l'assistant  
d'installation



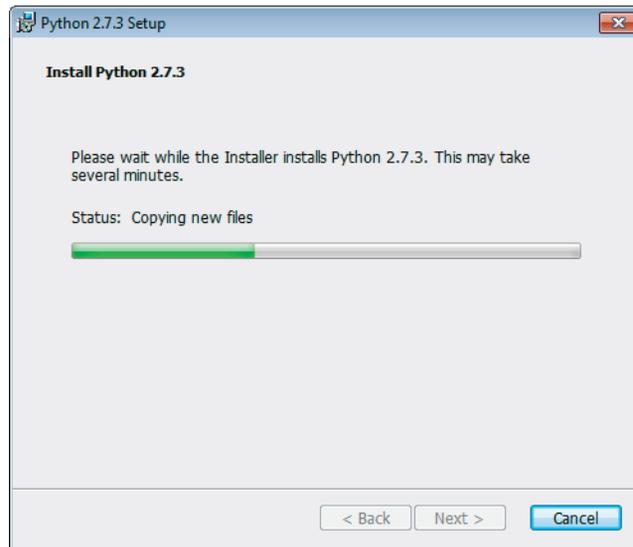
À la troisième étape, l'installateur demande quels composants de Python installer. Nous recommandons de tout sélectionner, afin de tout installer.

**Figure A-6**  
Troisième étape de l'assistant  
d'installation



À l'étape suivante, l'installation commence.

**Figure A-7**  
Quatrième étape de l'assistant  
d'installation



Python demeurant un langage et un environnement somme toute très léger, l'installation est terminée après quelques minutes. Il ne reste plus qu'à cliquer sur le bouton *Finish* pour fermer l'assistant.

**Figure A-8**  
Dernière étape de l'assistant  
d'installation



## Pour Mac OS X

Une fois l'image montée, il suffit d'exécuter l'application `Python.mpkg` et de suivre les étapes de l'assistant d'installation. À la fin du processus, on obtient cette fenêtre :

**Figure A-9**  
Dernière étape de l'assistant  
d'installation



Cet installateur met Python dans le dossier `Applications` de votre Mac.

## Vérification de l'installation

Il reste à valider la bonne installation du langage en lançant par exemple la ligne de commande graphique Python :

- Démarrer > Tous les programmes > Python > IDLE (Python GUI) sous Windows ;
- Applications > Python 2.7 > IDLE sous Mac OS X.

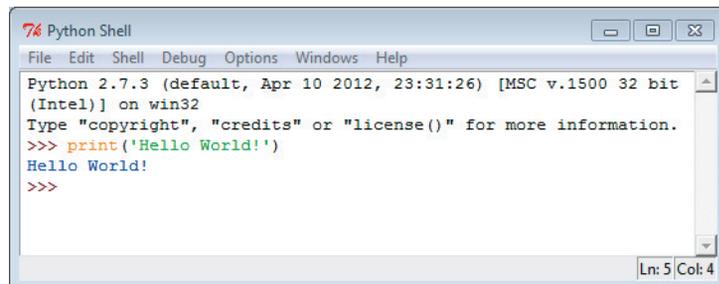
Ensuite, il suffit de saisir un peu de code, par exemple le sempiternel « hello world », bien connu des programmeurs :

### EXEMPLE A.1 Petit programme de test

```
print('Hello World !')
```

Ce code devrait afficher le texte « Hello World ! » dans la console, comme illustré sur la figure suivante. Vous pourriez refaire l'exercice avec « Bonjour Monde ! », mais sachez que cela fait tout de suite beaucoup moins pro.

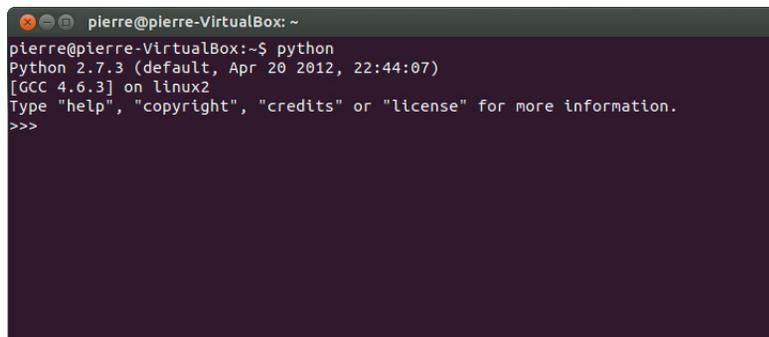
**Figure A-10**  
Test de l'installation



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print('Hello World!')
Hello World!
>>>
```

Sous Ubuntu, il suffit de lancer un terminal (via le *Tableau de bord*, on lance l'application *Terminal*) et de taper la commande `python`. Apparaît alors le numéro de version du langage :

**Figure A-11**  
Version de Python  
include dans Ubuntu



```
pierre@pierre-VirtualBox: ~
pierre@pierre-VirtualBox:~$ python
Python 2.7.3 (default, Apr 20 2012, 22:44:07)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

## Installation de Django

Django, écrit en Python, n'est pas livré en standard avec ce dernier. Il s'agit d'un framework tiers à installer manuellement.

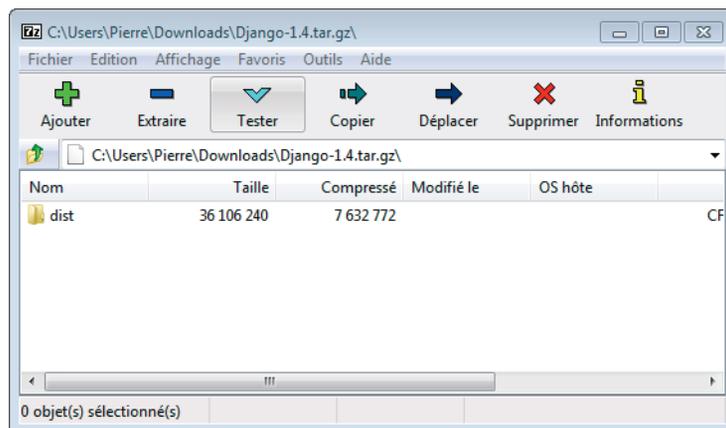
Pour Windows et Mac OS X, le framework est disponible en téléchargement sur le site du projet Django. Il faut se rendre dans la section *Download* et prendre la dernière version officielle. À l'heure où ces lignes sont écrites, il s'agit de la version 1.4. Le fichier à télécharger se présente sous la forme d'un fichier *tar.gz*, en d'autres mots une archive TAR qui a été compressée en format GZIP.

► [www.djangoproject.com](http://www.djangoproject.com)

### Pour Windows

Il faut utiliser une application tierce qui va ouvrir, décompresser et extraire les fichiers d'installation de Django que renferme le fichier *tar.gz*. L'application que nous avons utilisée est *7-Zip*, disponible gratuitement. D'autres décompresseurs et extracteurs peuvent bien entendu être utilisés. Ouvrez le fichier *.gz* à l'aide de l'application *7-Zip*. Un seul dossier se présente dans l'explorateur de l'archive : *dist*.

**Figure A-12**  
Contenu de l'archive TAR



Double-cliquez sur le dossier *dist* pour en afficher le contenu. Un seul fichier s'y trouve : *Django-1.4.tar*. À nouveau, double-cliquez sur ce fichier TAR pour en afficher le contenu. Dans cette archive, se trouve un seul dossier, nommé *Django-1.4* ; cliquez sur le bouton *Extraire* de *7-Zip*. Un emplacement destiné à accueillir le dossier extrait vous est demandé ; choisissez celui qui vous convient le mieux.

L'archive extraite, nous pouvons lancer l'installeur qui s'y trouve. Cet installeur étant écrit en Python, on ne peut pas le lancer comme un exécutable Windows traditionnel en double-cliquant dessus. Il va falloir le lancer en ligne de commande en utilisant Python.

Il faut d'abord lancer une ligne de commande Windows, avec les droits d'administrateur. La ligne de commande Windows se trouve dans *Menu démarrer>Tous les programmes>Accessoires>Invite de commande*. Pour la lancer avec les droits d'administrateur, il faut cliquer droit sur son raccourci et choisir *Exécuter en tant qu'administrateur*.

Une fois l'invite de commande lancée, il faut se rendre, à l'aide de la commande `cd`, dans le dossier d'installation de Django que l'on vient d'extraire. Par exemple, ayant extrait ce dossier dans `C:\Users\Pierre\Downloads\`, nous avons tapé la ligne de commande suivante :

#### EXEMPLE A.2 Positionnement dans le dossier d'installation

```
cd Downloads\Django-1.4
```

Nous n'avons pas spécifié le chemin absolu dans la commande, car en ouvrant la console, celle-ci nous avait déjà positionnés dans le dossier `c:\Users\Pierre`.

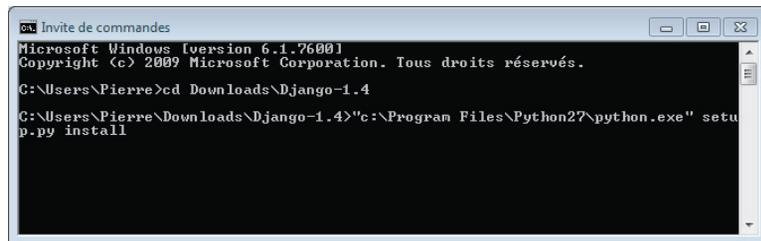
Pour lancer l'installation de Django, il faut entrer la commande suivante :

#### SYNTAXE. Lancement de l'installation

```
"c:\Program Files\Python27\python.exe" setup.py install
```

La figure suivante illustre l'utilisation de ces deux commandes.

**Figure A-13**  
Lancement de l'installation de Django



```
Invite de commandes
Microsoft Windows [version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Tous droits réservés.

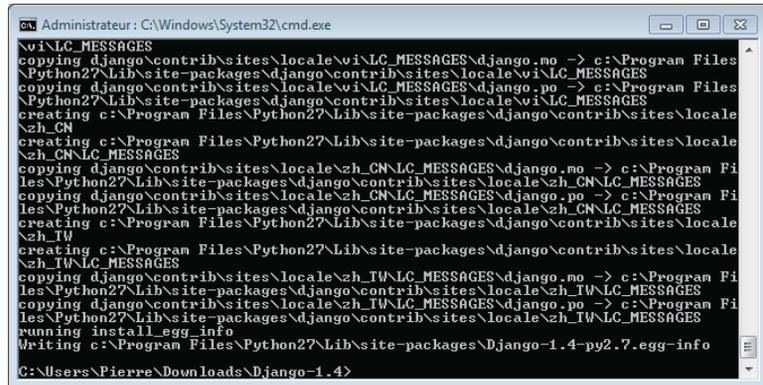
C:\Users\Pierre>cd Downloads\Django-1.4

C:\Users\Pierre\Downloads\Django-1.4>"c:\Program Files\Python27\python.exe" setup.py install
```

Si aucune erreur ne survient, c'est que l'installation s'est bien déroulée.

Figure A-14

Installation terminée de Django



```
Administrateur : C:\Windows\System32\cmd.exe
\n\LC_MESSAGES
copying django\contrib\sites\locale\vi\LC_MESSAGES\django.mo -> c:\Program Files
\Python27\Lib\site-packages\django\contrib\sites\locale\vi\LC_MESSAGES
copying django\contrib\sites\locale\vi\LC_MESSAGES\django.po -> c:\Program Files
\Python27\Lib\site-packages\django\contrib\sites\locale\vi\LC_MESSAGES
creating c:\Program Files\Python27\Lib\site-packages\django\contrib\sites\locale
\zh_CN
creating c:\Program Files\Python27\Lib\site-packages\django\contrib\sites\locale
\zh_CN\LC_MESSAGES
copying django\contrib\sites\locale\zh_CN\LC_MESSAGES\django.mo -> c:\Program Fi
les\Python27\Lib\site-packages\django\contrib\sites\locale\zh_CN\LC_MESSAGES
copying django\contrib\sites\locale\zh_CN\LC_MESSAGES\django.po -> c:\Program Fi
les\Python27\Lib\site-packages\django\contrib\sites\locale\zh_CN\LC_MESSAGES
creating c:\Program Files\Python27\Lib\site-packages\django\contrib\sites\locale
\zh_TW
creating c:\Program Files\Python27\Lib\site-packages\django\contrib\sites\locale
\zh_TW\LC_MESSAGES
copying django\contrib\sites\locale\zh_TW\LC_MESSAGES\django.mo -> c:\Program Fi
les\Python27\Lib\site-packages\django\contrib\sites\locale\zh_TW\LC_MESSAGES
copying django\contrib\sites\locale\zh_TW\LC_MESSAGES\django.po -> c:\Program Fi
les\Python27\Lib\site-packages\django\contrib\sites\locale\zh_TW\LC_MESSAGES
running install_egg_info
Writing c:\Program Files\Python27\Lib\site-packages\Django-1.4-py2.7.egg-info
G:\Users\Pierre\Downloads\Django-1.4>
```

## Pour Mac OS X

Pour l'archive sous Mac, il faut utiliser la ligne de commande. Ouvrez d'abord un *Terminal* (dans *Applications > Utilitaires*). Rendez-vous ensuite dans le dossier où a été téléchargée l'archive, à l'aide de la commande `cd` :

### EXEMPLE A.3 Positionnement dans le dossier de téléchargement

```
cd Downloads
```

Puis lancez la décompression de l'archive et rendez-vous dans le dossier créé :

### SYNTAXE. Décompression de l'archive

```
tar xzvf Django-1.4.tar.gz
cd Django-1.4
```

Ensuite, lancez l'installation de Django à l'aide de l'interpréteur Python que nous venons d'installer (et non de celui fourni de base dans Mac OS X). Il se trouve dans *Library > Frameworks*. Utilisez la commande `sudo` car l'installation doit se faire en tant qu'administrateur.

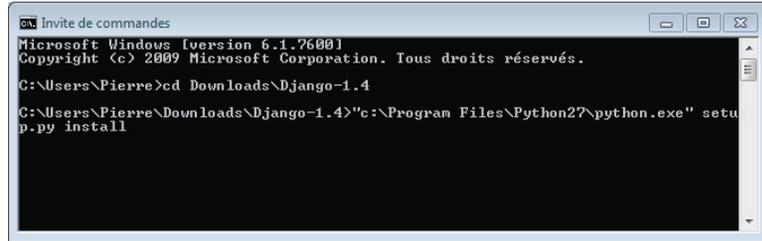
### SYNTAXE. Installation de Django

```
sudo /Library/Frameworks/Python.framework/Versions/2.7/bin/python
setup.py install
```

Attention, il va falloir entrer le mot de passe administrateur.

La figure suivante illustre l'utilisation de cette commande.

**Figure A-15**  
Lancement de l'installation  
de Django



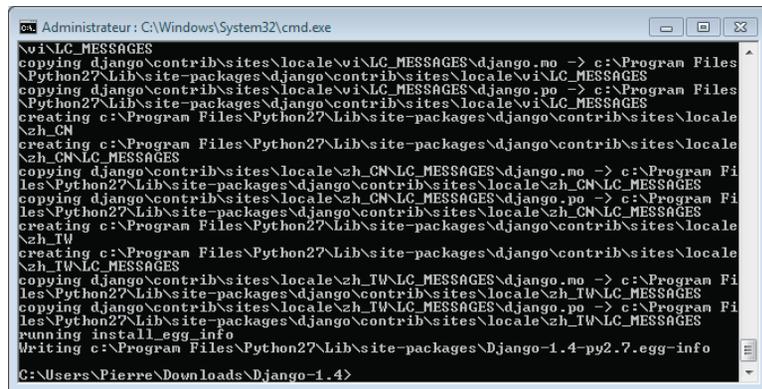
```

Invite de commandes
Microsoft Windows [version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Tous droits réservés.

C:\Users\Pierre>cd Downloads\Django-1.4
C:\Users\Pierre\Downloads\Django-1.4>"c:\Program Files\Python27\python.exe" setup
p.py install
  
```

Si aucune erreur ne survient, l'installation s'est bien déroulée.

**Figure A-16**  
Installation terminée de Django



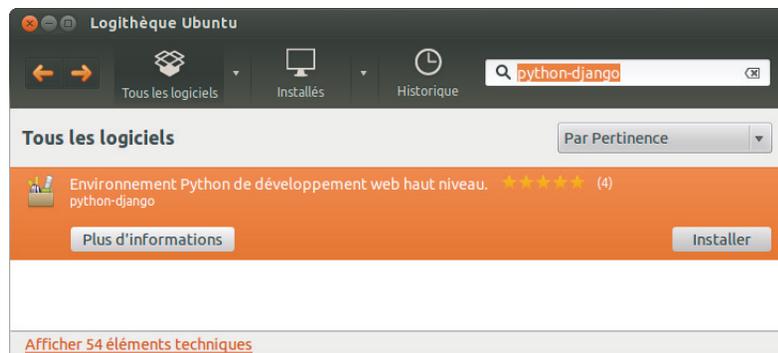
```

Administrateur : C:\Windows\System32\cmd.exe
\n\LC_MESSAGES
copying django\contrib\sites\locale\vi\LC_MESSAGES\django.mo -> c:\Program Files
\Python27\Lib\site-packages\django\contrib\sites\locale\vi\LC_MESSAGES
copying django\contrib\sites\locale\vi\LC_MESSAGES\django.po -> c:\Program Files
\Python27\Lib\site-packages\django\contrib\sites\locale\vi\LC_MESSAGES
creating c:\Program Files\Python27\Lib\site-packages\django\contrib\sites\locale
\zh_CN
creating c:\Program Files\Python27\Lib\site-packages\django\contrib\sites\locale
\zh_CN\LC_MESSAGES
copying django\contrib\sites\locale\zh_CN\LC_MESSAGES\django.mo -> c:\Program Fi
les\Python27\Lib\site-packages\django\contrib\sites\locale\zh_CN\LC_MESSAGES
copying django\contrib\sites\locale\zh_CN\LC_MESSAGES\django.po -> c:\Program Fi
les\Python27\Lib\site-packages\django\contrib\sites\locale\zh_CN\LC_MESSAGES
creating c:\Program Files\Python27\Lib\site-packages\django\contrib\sites\locale
\zh_TW
creating c:\Program Files\Python27\Lib\site-packages\django\contrib\sites\locale
\zh_TW\LC_MESSAGES
copying django\contrib\sites\locale\zh_TW\LC_MESSAGES\django.mo -> c:\Program Fi
les\Python27\Lib\site-packages\django\contrib\sites\locale\zh_TW\LC_MESSAGES
copying django\contrib\sites\locale\zh_TW\LC_MESSAGES\django.po -> c:\Program Fi
les\Python27\Lib\site-packages\django\contrib\sites\locale\zh_TW\LC_MESSAGES
running install_egg_info
writing c:\Program Files\Python27\Lib\site-packages\Django-1.4-py2.7.egg-info
C:\Users\Pierre\Downloads\Django-1.4>
  
```

## Pour Ubuntu

Django est disponible dans la *Logithèque* d'Ubuntu. Il suffit de chercher le logiciel `python-django` et, celui-ci trouvé, de cliquer sur *Installer*.

**Figure A-17**  
Django dans la Logithèque  
Ubuntu



## Vérification de l'installation

Sous Windows et Mac OS X, on peut tester si l'installation s'est exécutée correctement en lançant une console Python et en tapant le code suivant :

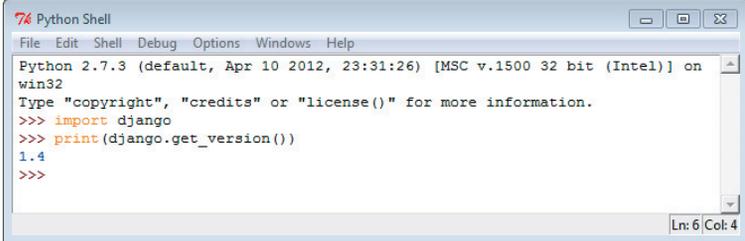
### SYNTAXE. Vérification de la bonne installation de Django sous Windows et Mac OS X

```
>>> import django
>>> print(django.get_version())
1.4
```

Le code doit afficher la version de Django, soit « 1.4 », comme illustré à la figure suivante.

**Figure A-18**

Vérification de la bonne installation de Django sous Windows et Mac OS X



```
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import django
>>> print(django.get_version())
1.4
>>>
```

Sous Ubuntu, on peut tester la bonne installation de Django en lançant un terminal et en tapant le code suivant :

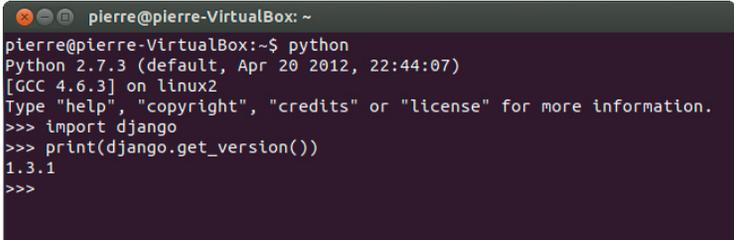
### SYNTAXE. Vérification de la bonne installation de Django sous Ubuntu

```
python
>>> import django
>>> print(django.get_version())
1.3.1
```

Le code doit afficher la version de Django, soit « 1.3.1 », comme illustré à la figure suivante.

**Figure A-19**

Vérification de la bonne installation de Django sous Ubuntu



```
pierre@pierre-VirtualBox: ~
pierre@pierre-VirtualBox:~$ python
Python 2.7.3 (default, Apr 20 2012, 22:44:07)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>> print(django.get_version())
1.3.1
>>>
```

## Installation de Java

Si Java n'est pas déjà installé sur votre ordinateur (ce qui est peu probable), vous devrez y remédier.

### Pour Windows et Mac OS X

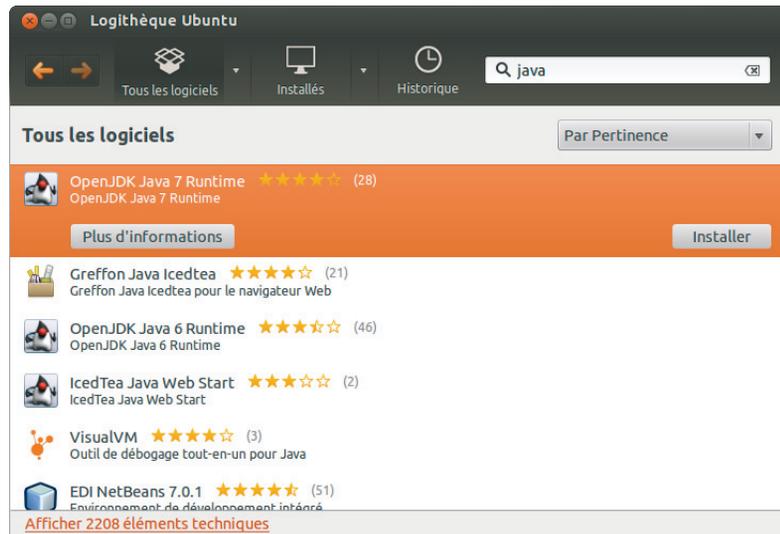
Sur la page d'accueil du site officiel de Java, un lien mis en évidence vous permet de télécharger l'installateur. Choisissez la version de Java correspondant à votre système (32 ou 64 bits). L'installation ne nécessite pas de paramétrage particulier.

► [www.java.com](http://www.java.com)

### Pour Ubuntu

Rendez-vous à nouveau dans la *Logithèque*. Il suffit d'y chercher « Java », de sélectionner *OpenJDK Java 7 Runtime* et de cliquer sur *Installer*.

**Figure A-20**  
Installation de Java  
avec Ubuntu



## Installation d'Eclipse

Nous allons maintenant installer Eclipse, notre environnement de développement intégré.

Le téléchargement d'Eclipse se fait via le site officiel du projet pour Windows et Mac OS X. On retrouve sur cette page plusieurs versions de l'outil, chacune ciblant un langage de programmation ou un usage bien précis. Elles ne sont en fait que la version de base de l'outil sur laquelle ont été pré-installés les plug-ins ad hoc.

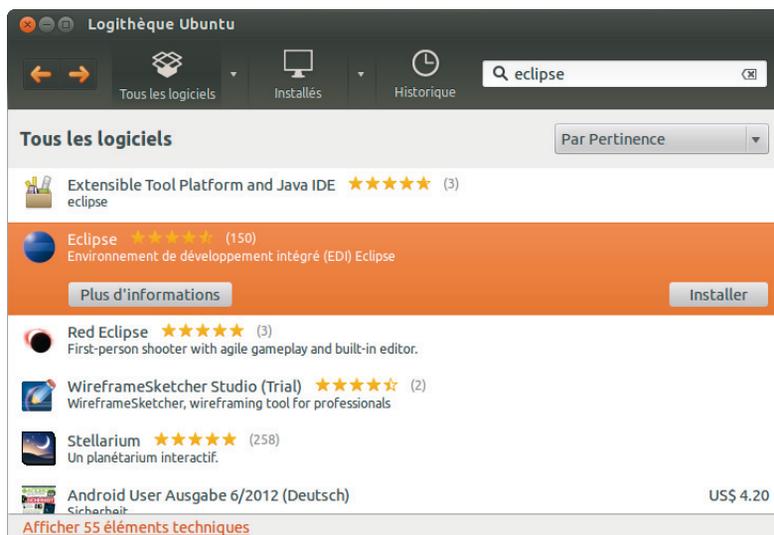
► <http://www.eclipse.org/downloads/>

Comme il n'en existe pas pour Python et Django, nous allons simplement télécharger la version de base, « Eclipse Classic » pour Windows ou Mac (version 3.7.2 au moment où nous écrivons ces lignes), et nous installerons par la suite les plug-ins qui nous intéressent.

Une fois téléchargé, Eclipse se présente sous la forme d'une archive (ZIP pour Windows et tar.gz pour Mac), qu'il suffit de décompresser comme nous l'avons fait pour Django. Elle contient un dossier reprenant tous les fichiers de l'application. Aucun installateur n'est donc à lancer ; il suffit de déplacer le dossier que l'on a décompressé de l'archive à l'emplacement de son choix. Pour notre part, nous avons choisi de déplacer le dossier dans `C:\Program Files` sur Windows et `Applications` pour Mac.

Pour Ubuntu, nous allons faire usage une fois encore de la *Logithèque*. Il suffit de chercher *Eclipse*, de le sélectionner et de cliquer sur *Installer*.

**Figure A-21**  
Installation d'Eclipse  
sous Ubuntu



Nous pouvons maintenant lancer Eclipse :

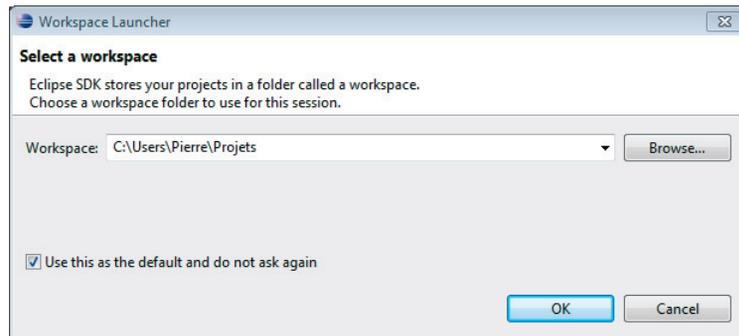
- Sous Windows, double-cliquez sur `eclipse.exe`.
- Sous Mac, si Java n'est pas installé à ce stade, *Mise à jour de logiciels* permet de le faire en affichant un dialogue le proposant.
- Sous Ubuntu, démarrez Eclipse à l'aide du tableau de bord.

Au démarrage, Eclipse nous demande de choisir un *workspace* (espace de travail) et nous propose, par défaut, d'utiliser `C:\Users\XXXXX\workspace` pour Windows et Ubuntu ou `/Users/XXXXX/Documents/workspace` pour Mac. Un workspace regroupe différents projets, spécifie leur emplacement sur le disque et enregistre un certain nombre de préférences propres à Eclipse. Définir plusieurs *workspaces* permet donc d'avoir des environnements de travail différents (paramétrés différemment) contenant des projets différents. Tout au long de ce livre, nous n'utilisons qu'un seul workspace.

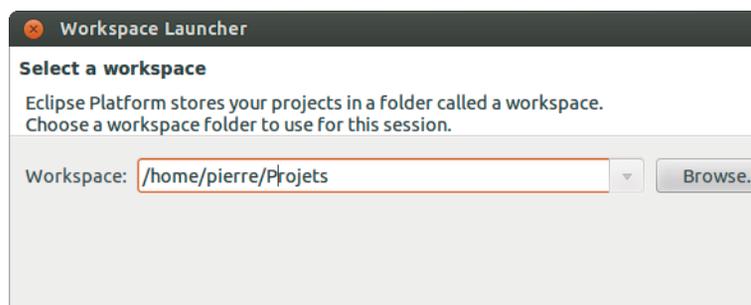
Vous pouvez très bien utiliser le workspace proposé par défaut. Néanmoins, nous avons préféré créer et utiliser un nom plus convivial : `C:\Users\Pierre\Projets` pour Windows et Ubuntu ou `/Users/Pierre/Documents/Projets` pour Mac. Pour créer un nouveau workspace, il suffit d'entrer son nom dans la boîte de dialogue de choix de workspace.

Vous pouvez également cocher la case *Use this as the default and do not ask again* afin qu'au prochain démarrage, Eclipse ne vous demande plus quel workspace utiliser.

**Figure A-22**  
Création et utilisation  
du workspace  
`C:\Users\Pierre\Projets`  
(Windows)

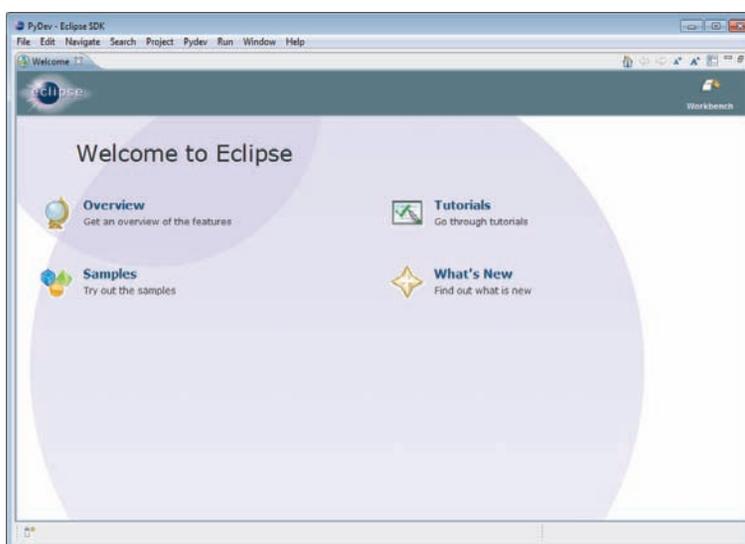


**Figure A-23**  
Création et utilisation  
du workspace  
C:\Users\Pierre\Projets  
(Ubuntu)



Il reste à cliquer sur *OK*. Eclipse s'ouvre alors et affiche sa page d'accueil.

**Figure A-24**  
Écran d'accueil d'Eclipse



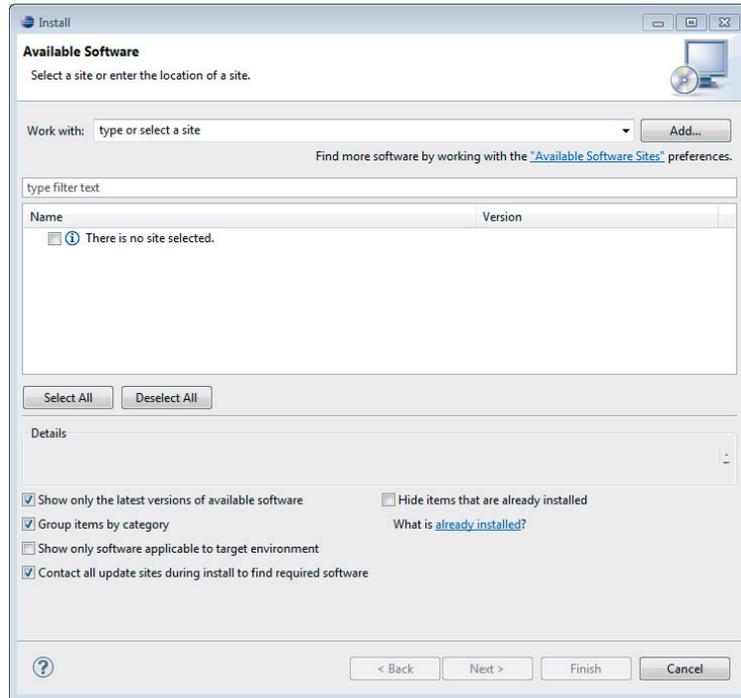
Et voilà ! Eclipse est installé, Nous n'avons plus à nous occuper que du plug-in PyDev.

## Installation du plug-in Eclipse PyDev

Avant de pouvoir commencer à développer en Python et Django dans Eclipse, il nous faut installer le plug-in PyDev.

Les plug-ins s'installent via le menu *Help > Install New Software...* La fenêtre suivante apparaît :

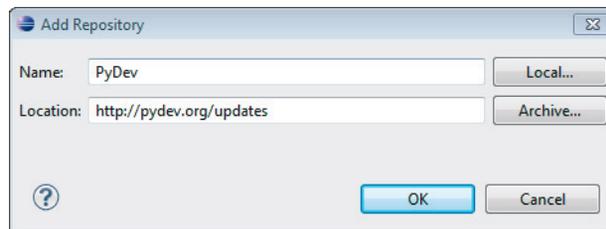
**Figure A-25**  
Écran d'installation de plug-ins



Tout d'abord, nous devons renseigner le *site* sur lequel se trouve le plug-in PyDev. Cliquez sur le bouton *Add...* et, dans la fenêtre qui apparaît, inscrivez *PyDev* et <http://pydev.org/updates>.

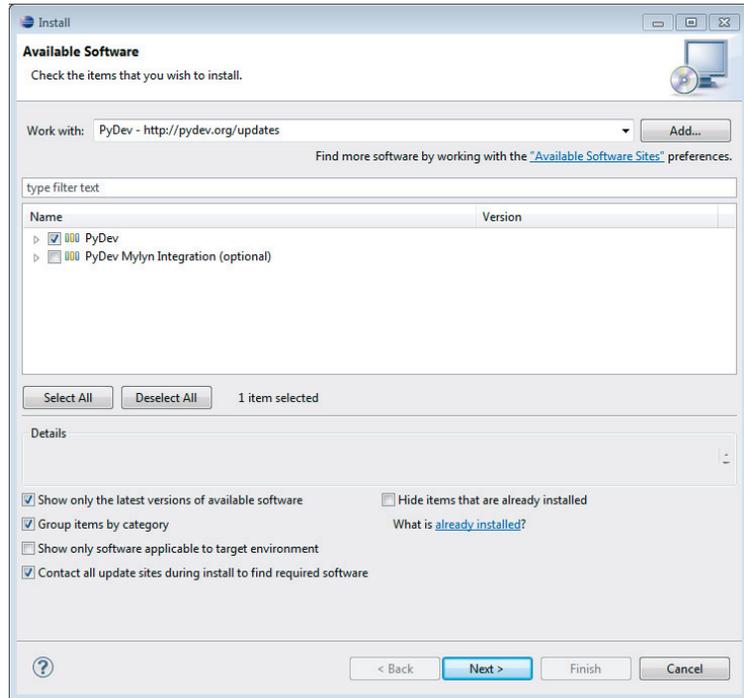
► [pydev.org/updates](http://pydev.org/updates)

**Figure A-26**  
Ajout du site pour  
le plug-in PyDev



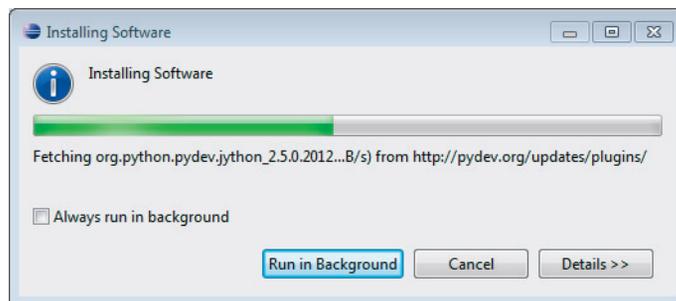
Cliquez sur *OK*. Choisissez ensuite *PyDev* dans la liste des plug-ins disponibles à installer.

**Figure A-27**  
Choix de PyDev



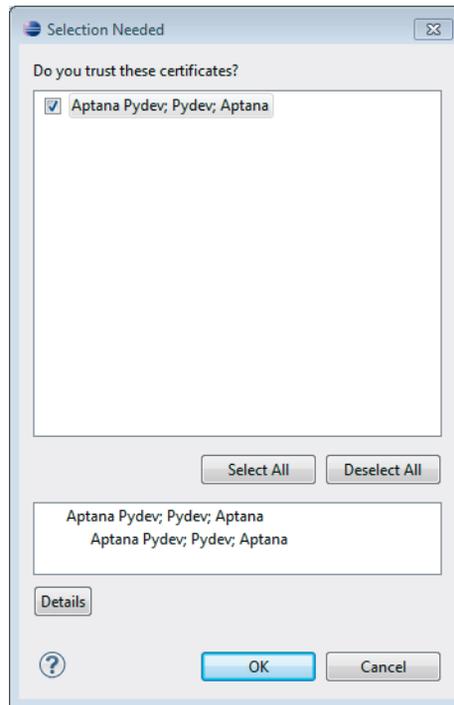
On clique sur *Next* deux fois, on accepte la licence, on clique sur *Finish* et l'installation débute.

**Figure A-28**  
Installation de PyDev



Au cours de l'installation, Eclipse vous demandera de valider un certificat. Il suffit de s'assurer qu'ils sont tous sélectionnés et de cliquer sur *OK*.

**Figure A-29**  
Installation de PyDev



Il est possible qu'à ce stade, on vous demande de redémarrer votre ordinateur, pour que les plug-ins s'installent vraiment. Vous pouvez vous contenter de réexécuter Eclipse.

Il reste maintenant à configurer PyDev pour lui indiquer où se trouve le compilateur Python et ses bibliothèques (que nous avons installées au tout début de ce chapitre).

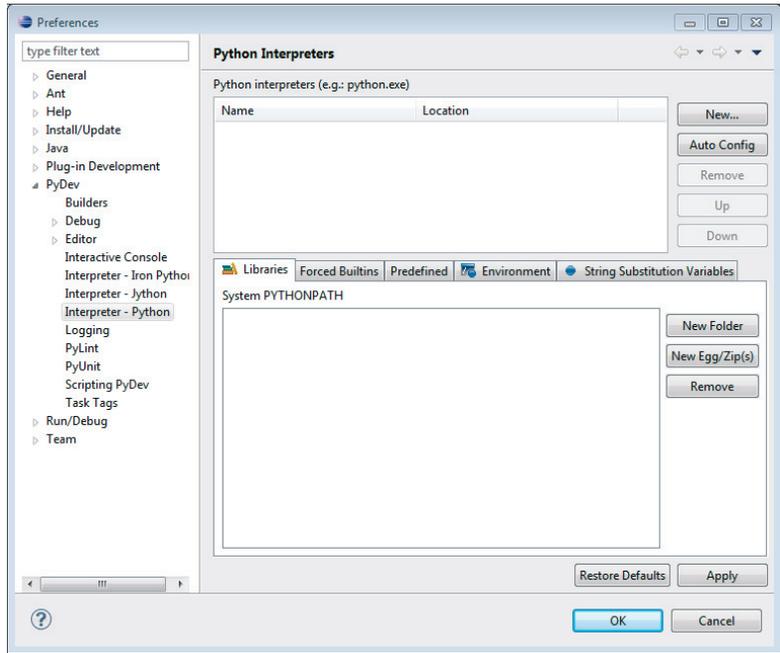
Pour ce faire, il faut aller dans le menu *Window > Preference* (ou *Eclipse > Preference*). Dans l'arborescence de paramètres, rendez-vous à la section *PyDev > Interpreter - Python*.

Cliquez sur le bouton *Auto Config*. Dans la fenêtre qui apparaît, choisissez tous les éléments du dossier d'installation de Python. Cliquez sur *OK* pour valider.

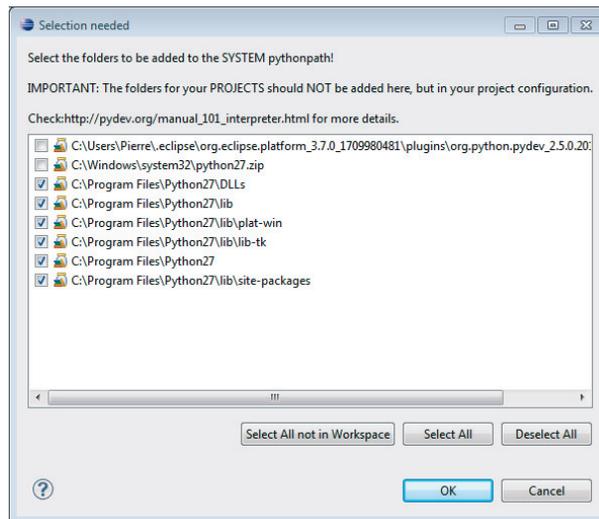
Si *Auto Config* ne marche pas, il suffit alors de faire *New* et de spécifier comme emplacement l'endroit où se trouve l'exécutable de l'interpréteur Python que l'on désire utiliser. Par exemple, pour Mac OS X, il s'agira de `/Library/Frameworks/Python.framework/Versions/2.7/bin/python`.

La configuration est maintenant terminée, cliquez sur *OK* pour fermer la fenêtre de préférences.

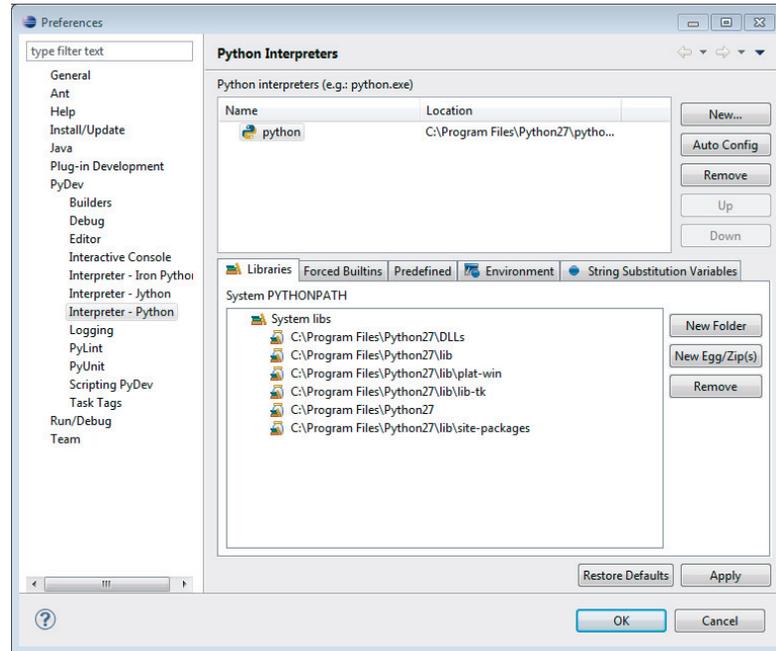
**Figure A-30**  
Configuration de PyDev



**Figure A-31**  
Configuration de PyDev



**Figure A-32**  
Configuration de PyDev



## Installation de Web Developer Tools pour Eclipse

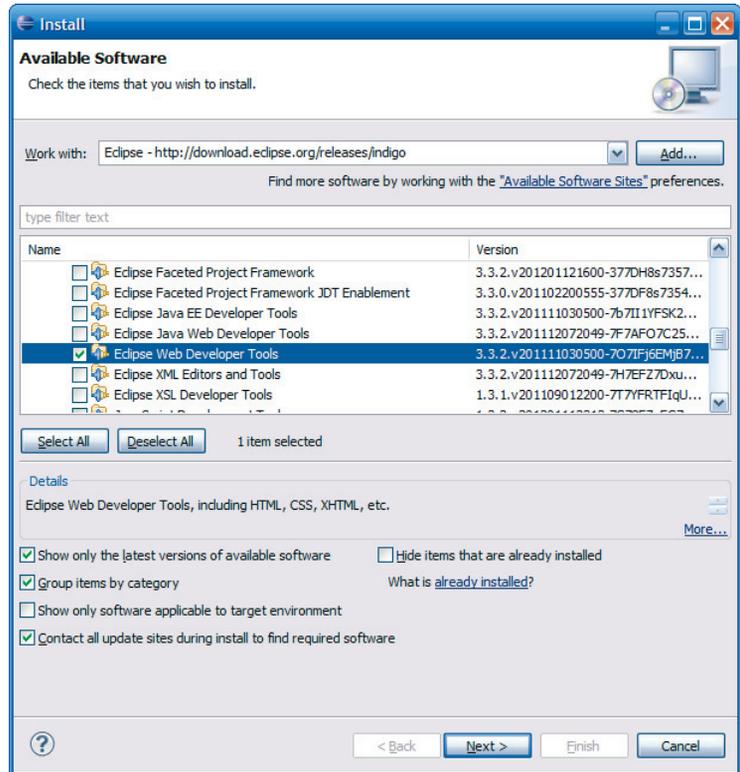
Ce paquet supplémentaire dans Eclipse nous sert dans le chapitre 8 pour éditer directement nos HTML. Il suffit de procéder de la même manière que pour PyDev.

Rendez-vous dans le menu *Help > Install New Software* et cliquez sur le bouton *Add*. Comme nom de site, indiquez ce que vous voulez, par exemple *Eclipse*. Comme emplacement (*location*), mettez <http://download.eclipse.org/releases/indigo> (du moins, si vous travaillez avec la version Indigo d'Eclipse). Si vous utilisez une autre version, il faut remplacer *indigo* dans l'adresse par le nom de celle-ci (regardez bien l'écran de démarrage d'Eclipse, il y est indiqué).

Le site ajouté, sélectionnez-le et patientez pendant que la liste des paquets disponibles se met à jour. Rendez-vous ensuite dans *Web, XML, Java EE and OSGi Enterprise Development* et sélectionnez le paquet *Eclipse Web Developer Tools*.

Suivez les instructions de l'assistant d'installation et répondez aux questions qu'il vous pose. Après un redémarrage d'Eclipse, vous voilà en mesure d'éditer vos HTML directement dans Eclipse.

**Figure A-33**  
Installation du plugin « Web Developer Tools »



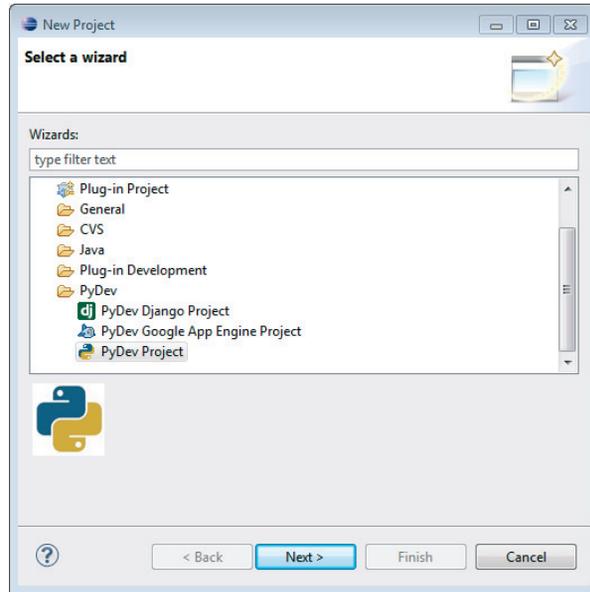
## Premier projet de test

Les différentes installations terminées, nous pouvons réaliser un premier petit projet de test. Cela permet de valider que tout fonctionne à la perfection. Ce petit projet n'a d'autre but que de réaliser un programme Python affichant toujours notre sempiternel texte « Hello World ! » à la console. Pour créer un nouveau projet, cliquez sur le menu *File > New > Project...* puis choisissez *PyDev Project*.

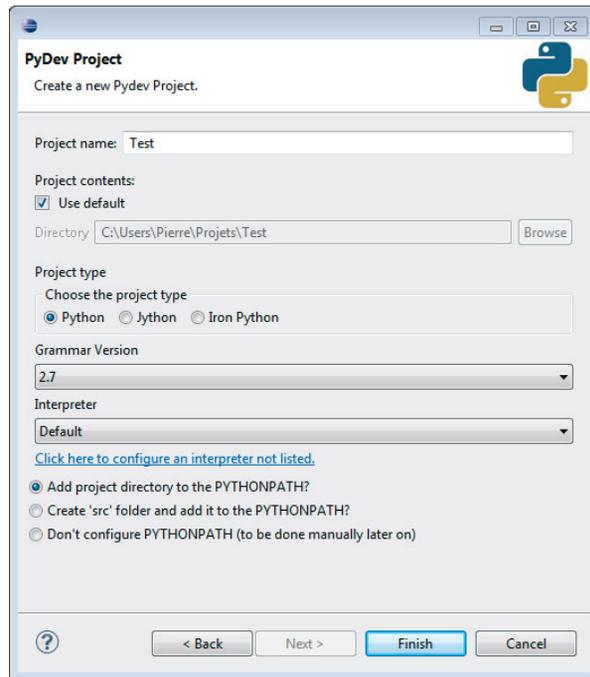
Après avoir cliqué sur *Next*, un assistant de création de projet se lance. À la première étape, il suffit de donner un nom au projet.

Cliquez alors sur *Finish*. Notre projet est créé et apparaît dans l'écran principal d'Eclipse.

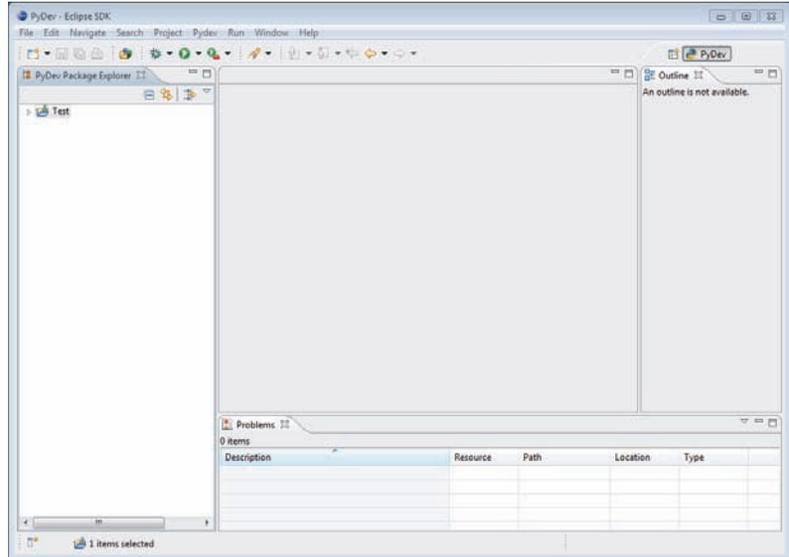
**Figure A-34**  
Création d'un projet de test



**Figure A-35**  
Création d'un projet de test

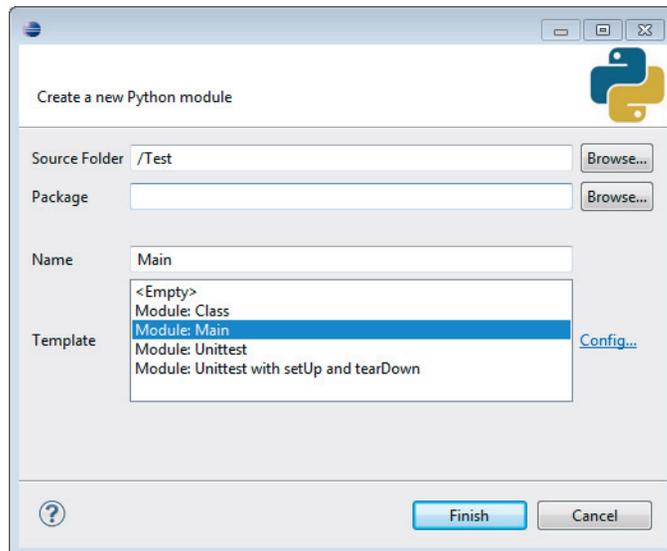


**Figure A-36**  
Création d'un projet de test



Il reste à ajouter à notre projet un fichier `py` qui va contenir nos sources. Cliquez-droit sur le projet `Test` et choisissez `New > PyDev Module`. Une fenêtre apparaît dans laquelle on précise un nom pour le projet (`Main`) et un *template* de base ; dans notre cas, nous allons choisir le template `Module: Main`, car nous allons écrire notre code au niveau du point d'entrée du programme.

**Figure A-37**  
Création d'un projet de test



Cliquez sur *Finish*, le fichier est alors créé. Supprimez-en tout le contenu, en particulier les commentaires ajoutés par défaut par Eclipse, car pour peu qu'ils contiennent des accents, Python plantera. Nous verrons comment gérer correctement les accents dans le chapitre 8. Cela nous permettra de garder le code proposé en standard dans ces fichiers.

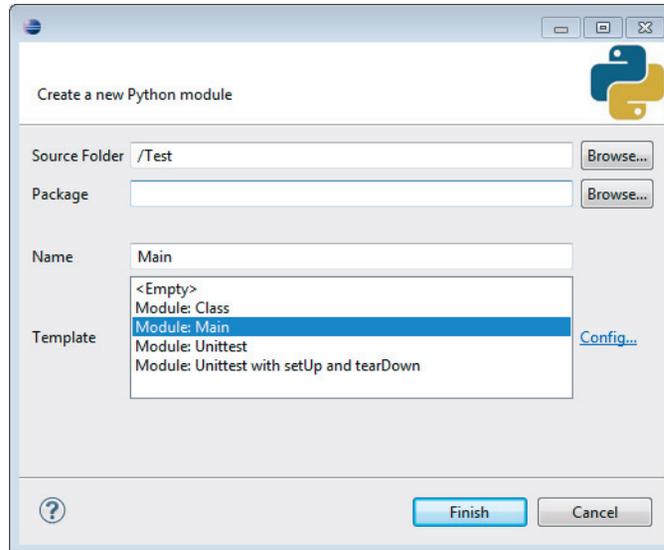
Il reste à insérer le code suivant dans le fichier fraîchement créé :

#### EXEMPLE A.4 HelloWorld ! en Python

```
if __name__ == '__main__':  
    print('Hello World !')
```

On peut maintenant exécuter le programme en utilisant l'icône *Run* représentée par un triangle blanc sur un rond vert. Une fois exécuté, le programme affiche « HelloWorld ! » à la console.

**Figure A-38**  
Création d'un projet de test



Et voilà ! Nous avons créé un programme Python en utilisant Eclipse et son plug-in PyDev. Nous sommes prêts pour des projets plus complexes ayant recours au framework Django.

## Ai-je bien compris ?

- Pourquoi faut-il installer Java alors que l'on va programmer en Python ?
- Eclipse est-il obligatoire pour développer un site en Django ?
- À quoi sert le plug-in Eclipse « PyDev » ?

# Index

---

## A

Access 131  
Ajax (Asynchronous JavaScript and XML) 270

## B

base de données 12  
  nettoyer 234  
base de données relationnelle 214  
  clé primaire 114, 119  
  lien 208  
  relation 1-1 116  
  relation 1-n 114  
  relation n-n 117

## C

cardinalité d'un lien 122  
cas d'utilisation 104  
CGI (Common Gateway Interface) 126, 131,  
  133, 145  
cgi-bin 133, 139  
classe 45  
  attribut 45  
  attribut statique 48  
  constructeur 47  
clé  
  primaire 115, 129  
cookie 233, 238  
  durée de vie 235  
CSS (Cascaded Style Sheets) 58, 60  
  \* (sélecteur) 83  
  absolute 90  
  active 82  
  background-color 79  
  block 88  
  border 86  
  border-width 79

  dimension des éléments 86  
  display 98  
  font-family 79  
  height 79, 86  
  hover 82  
  inline 88  
  link 82  
  marges par défaut 95  
  margin 86  
  padding 86  
  position 89  
  positionnement par défaut 88  
  propriété 79  
  relative 89  
  sélecteur 79  
  sortir un élément du flux 89  
  static 89  
  top 89  
  visited 82  
  width 86

## D

DHTML (Dynamic HTML) 96–97  
diagramme de classes 202  
Django 23, 289  
  \_\_startswith 212  
  \_\_str\_\_ 218  
  \_\_unicode\_\_ 218  
  \_set 213  
  action 186, 188  
  admin.py 215  
  ajax/ 272  
  all 248  
  as\_p 193  
  as\_table 193  
  autodiscover() 215

- blank 208, 211
  - BooleanField 192
  - clean 198, 273
  - cleaned\_data 198
  - commit 222
  - configuration 157, 160, 168, 234–235
  - datetime 170
  - DEFAULT\_CONTENT\_TYPE 160
  - DEFAULT\_CHARSET 157
  - delete 213
  - django-admin.py cleanup 234
  - EmailField 191
  - emplacement des fichiers projet 180
  - errorList 196
  - Field 273
  - filter 212
  - focusout 274–275
  - Form 191, 221
  - forms 190, 254
  - forms.py 221
  - formulaire 186, 188, 221, 225–227, 280
  - GET 187
  - get 212
  - hidden 229
  - HttpRequest 187
  - HttpResponse 154
  - HttpResponseRedirect 188
  - installation 297
  - is\_bound 228
  - is\_valid 195
  - message d'erreur 189
  - Model 203
  - model 150
  - modèle 202
  - ModelForm 220, 226, 263
  - models 204
  - models.ForeignKey 207
  - models.ManyToManyField 207
  - models.OneToOneField 207
  - models.py 203
  - none 239
  - PasswordInput 192
  - patterns 152
  - raise 198
  - ready 274–275
  - redémarrer l'application 169
  - redirection 186
  - render\_to\_response 167, 186
  - request 154, 187, 195
  - save 221
  - session 233
  - settings.py 157, 160, 180, 204, 215–216
  - super 198
  - Sync DB 205, 211, 241
  - template 150, 166
  - TEMPLATE\_DIRS 168
  - try...except 273
  - tuple 152
  - urlpatterns 152
  - urls.py 152, 158, 171, 178, 215, 222, 250
  - valider un courriel 197
  - valider un formulaire 194
  - view 150
  - views.py 153, 158, 168, 172, 179
  - vue 151, 153, 192
  - XHTML 5 160
  - document HTML
    - encodage 65
    - structure 63
- E**
- Eclipse 155, 289
    - installation 303
    - plug-in PyDev 305, 310
  - encodage
    - ISO 8859-1 66
    - UTF-8 66, 138, 157, 167
  - événement 96
  - exception 273
  - expression régulière 153
- F**
- faible de sécurité 232
  - feuille de styles 78
  - fonction anonyme 275
  - formulaire 270
  - framework
    - Django 150

**H**

- héritage 17, 120
- HTML (HyperText Markup Language) 58–59, 126, 138, 145
  - a 70
  - action 76
  - alt 72
  - article 66
  - aside 68
  - attribut 62
  - balise 60
  - body 63
  - change 96
  - class 81
  - click 96
  - dblclick 96
  - dl 259
  - em 73
  - encoding 65
  - figcaption 72
  - figure 72
  - footer 68
  - form 75
  - formulaire 75
  - h1, h2, h3 67
  - head 63
  - header 68
  - height 72
  - html 63
  - id 80
  - image 72
  - img 72
  - input 75
  - lang 62
  - li 71
  - link 85
  - liste 71
  - method 76
  - mise en évidence de texte 73
  - mouseover 96
  - name 76
  - ol 71
  - onclick 99
  - p 66
  - script 99
  - section 66
  - src 72
  - strong 73
  - style 83–84
  - submit 75–76
  - text 75
  - title 61
  - type 75
  - ul 71
  - value 76
  - width 72
  - XHTML 60
  - xmlns 63
- HTTP (HyperText Transfer Protocol)
  - redirection 186
  - requête 75, 187
- I**
- Informix 131
- instruction Python
  - \_\_init\_\_ 45
  - \_\_str\_\_ 47
  - append 35
  - class 45
  - def...return 43
  - del 36
  - for...in 41
  - if...elif...else 39
  - if...else 37
  - import 288
  - indentation 38
  - insert 35
  - len 35
  - print 31, 138
  - self 47
  - type 31
  - while 40
- instruction SQL
  - DELETE FROM...WHERE 132
  - INSERT INTO...VALUES 132
  - SELECT...FROM...WHERE 131
  - UPDATE...SET...WHERE 132
- intégrité des données 233

interactivité client-serveur 268–269

Internet 4

instruction SQL

ORDER BY 132

## J

Java

installation 302

JavaScript 58, 96, 270, 275, 280

getElementById 98

jQuery 99, 229, 272, 274–276

ajax 276

manipulation d'élément 101

prepend 283

sélection d'élément 101

## L

langage

SQL (Structured Query Language) 131

langage de programmation

Python 145, 288

script 96

lien hypertexte 4, 60, 70

## M

modèle de données 104, 113, 119, 122, 127

modèle Django

accéder à des objets liés 213

création 202

créer un enregistrement 211

récupérer plusieurs enregistrements 212

récupérer un enregistrement 212

superuser 206

supprimer des enregistrements 213

trier des données 212

utilisation 211

MTV (Model Templates Views) 150, 166

MVC (Modèle Vue Contrôleur) 58, 150

MVC (Modèle-Vue-Contrôleur) 18

## N

navigateur web 6, 11

## O

Oracle 131

## P

page web 4

dynamique 10, 135

passage de paramètres 13

statique 134

PK (Primary Key) 116

port 162

port de communication 133

programmation

orientée Objet 17

procédurale 17

programmation Objet 45

association entre classes 47

héritage 51

polymorphisme 53

programme modulaire 16

protection des pages 238

protocole HTTP (HyperText Transfer Protocol) 4, 7

Python 17, 26–27

bibliothèque 54

installation 292

interpréteur 138

ordre des classes 210

script 126, 133

## S

scénario d'un site 112

sécurité des données 233

serveur de bases de données 12

serveur web 6, 9

serveur web Python 133

session 231–233, 238

configuration 234

données 237

utilisation 235

SGBD (Système de gestion de bases de données) 12

site web

dynamique 9, 11, 13

page de déconnexion 234

page par défaut 250

statique 5

SQL (Structured Query Language) 126, 131, 133, 145, 202

SQLite 129, 139, 214

  sqlite.db 206, 211, 217

Sybase 131

## T

template

`{%block%}{%endblock%}` 175

`{%extends%}` 176

`{%for%}{%endfor%}` 174

`{%if%}{%else%}{%endif%}` 173

`{{current_date_time}}` 170

`{{logged_user_name}}` 166

  attribut de variable 172

  bodyId 177

  condition 173

  content 177

  élément d'une liste 172

  first 173

  genericPage 177

  headerContent 177

  héritage 174

  langage 172

  length 173

  lower 173

  principe 166

  title 177

  variable 172

## U

UML (Unified Modeling Language) 20

URL (Uniform Resource Locator) 4, 8

  use case 20–21, 105, 127

## V

variable

  chaîne de caractères 33

  de classe 48

  déclaration 28

  dictionnaire 36

  entier 31

  liste 35

  locale/globale 44

  réel 32

  transtypage 29

  type 29

Von Neumann 29

## W

Web 4

  Web Developer Tools 167

  wireframe 104, 106, 127

  World Wide Web 4

## X

XHTML 138