Haute Ecole d'ingénierie et d'Architecture de Fribourg

Filière Informatique

Formation Did@cTIC

Travail de diplôme

Amélioration du cours Système d'Informations II

Stefano Carrino & Joël Dumoulin
Août 2015

Table des matières

Introduction

Ce travail de diplôme de la formation Did@cTIC a été articulé autour du cours « Système d'Informations II » donné aux étudiants de Bachelor en 3ème année Informatique de la Haute école d'ingénierie et d'architecture de Fribourg.

Le cours y est décrit en détail, puis analysé à l'aide des différents outils abordés durant la formation Did@cTIC. Les différentes modifications et améliorations apportées au fil des années à ce cours par les auteurs de ce document y sont détaillées et analysées. Le reste du document est organisé comme suit.

Dans le chapitre 2, le cours « Système d'Informations II » est introduit, puis sa structure est décrite en détail. Le rôle des auteurs, le contexte du cours dans son cadre académique, son organisation, les thèmes qui y sont traités, les compétences visées pour les étudiants ainsi qu'une description des modalités d'évaluation sont autant de points détaillés dans ce chapitre.

Ensuite, la partie théorique et la partie pratique sont traitées séparément dans deux chapitres distincts. Dans le chapitre 3, la partie théorique est analysée. Les approches du cours sont présentées, puis, suite aux évaluations réalisées, des modifications sont détaillées et illustrées à l'aide d'un cas concret. Finalement leur impact est analysé, et des propositions pour le futur sont formulées. Dans le chapitre 4 la même démarche est appliquée à la partie pratique.

Finalement, un chapitre de conclusion clôt ce travail.

Le cours « Systèmes d'Information 2 »

Cours obligatoire de Bachelor, 3ème année informatique (HEIA-FR), le cours de Systèmes d'Information 2 (SI2) à une durée d'un semestre. Le cours vise à donner aux apprenants les connaissances nécessaires pour analyser les besoins concrets d'un système d'information et en concevoir l'architecture sur la base de modèles existants ; la présentation théorique des technologies informatiques pertinentes pour la réalisation, configuration et entretien d'un système d'information est combinée avec des activités de laboratoire sous forme de travaux dirigés (TD) et travaux pratiques (TP), d'exercices individuels et collaboratifs.
Le cours s'adresse à des apprenants francophones.

1 Un cours dynamique

Pendant le semestre, les apprenants sont amenés à rencontrer des sujets à la pointe des technologies informatiques en ce qui concerne le développement web. L'informatique et le web en particulier sont caractérisés par les rythmes frénétiques des mises à jour, évolutions et révolutions qui ont un impact formidable sur les technologies, les paradigmes et le savoir-faire nécessaires pour pouvoir les maitriser. Le cours de SI2 a pour but the fournir aux apprenants les outils de bases pour rentrer dans ce monde en évolution continue.

Puisque une technologie informatique a un cycle de vie très rapide, plutôt que de se focaliser sur une technologie spécifique et de perfectionner les étudiants dans l'utilisation de tel ou tel outil, le cours SI2 vise une approche plus holistique dans laquelle plusieurs technologies sont présentées en mettant l'accent sur les raisons qui ont amené de telles technologies à naitre et se développer. En particulier, le cours accentue les éléments communs à différents systèmes d'information et leur raison d'être.

En d'autres termes, à la fin du cours les apprenants ne seront capables de maitriser jusqu'au bout aucunes des différentes technologies mais, plus important pour la figure professionnelle d'un ingénieur, ils seront capables d'avoir une vision d'ensemble de ce qu'est un système d'information, avec ses différentes couches, comment elles communiquent et leur architecture.

Par conséquent, dû à la richesse des contenus et au dynamisme de la matière et des sujets présents, les enseignants et les collaborateurs sont engagés à donner un effort non négligeable dans la mise à jour des contenus du cours.

2 Structure du cours

2.1 Le rôle des auteurs dans le cadre de SI 2

Les auteurs de ce rapport ont des rôles différents mais complémentaires dans le cadre du cours de SI 2.

Stefano Carrino, comme chargé de cours, a la responsabilité principale de donner les cours théoriques ex-cathedra et de supporter Joël Dumoulin pendant les TD et TP.

Inversement, Joël Dumoulin est responsable pour la mise en place des TD et TP mais aussi, de temps en temps, est chargé de donner des cours sur des topiques dont il possède une excellente maitrise.

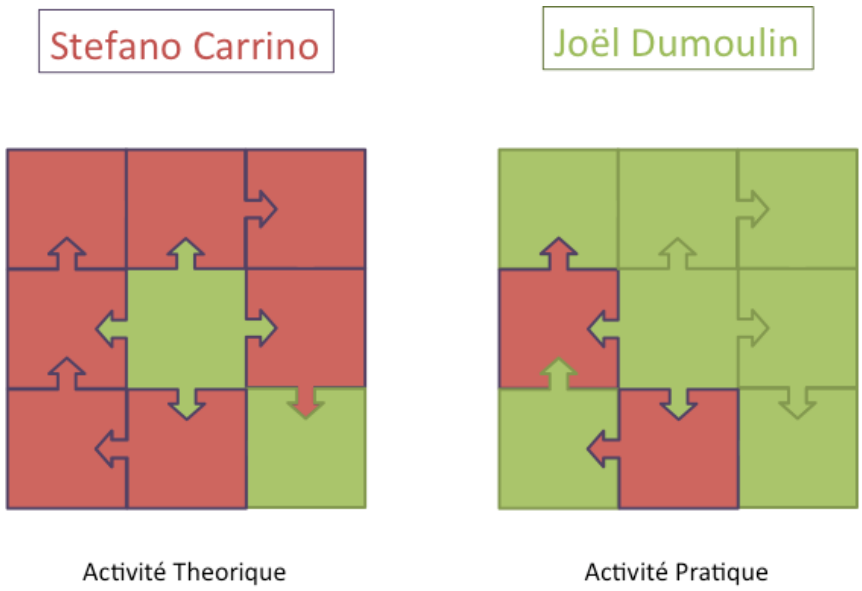


Tableau 1 : Représentation schématique du partage des tâches entre les auteurs

En préparant et donnant ce cours, les auteurs visent outre la formation des étudiants à développer des compétences très importantes pour le futur de leurs activités pédagogiques et leur développement professionnel.

Ceci est caractérisé parfaitement par la définition de Bourdoncle (Bourdoncle, 1991), qui décrit le développement professionnel comme « *un processus d'amélioration des capacités et de rationalisation des savoirs mis en œuvre dans l'exercice de la profession, ce qui entraîne une plus grande maîtrise et une plus grande efficacité individuelle et collective* ».

Comme formation personnelle, puisque ce cours a était donné en parallèle à la formation Did@cTIC, nous avons pu appliquer sur le terrain les méthodes et les approches découvertes pendant la formation. Un facteur capital pour le développent des nos compétences a été aussi la présence et supervision de professeurs avec plusieurs années d'expérience (*mentoring*).

Néanmoins, la possibilité de donner, et en partie concevoir, ce cours, nous le considérons comme une source primordiale d'apprentissage informel (« *learning by doing » ou « on-the-job experience»)*.

2.2 Le contexte du cours dans son cadre académique

Comme mentionné précédemment, SI2 est un cours obligatoire de la 3eme année Bachelor de l'HEIA-FR. Le cours a des prérequis et des corequis.

2.2.1 Prérequis

- Bases de données
- Système d'exploitation 1
- Téléinformatique 1
- Algorithmique et structures de données 2
- Systèmes d'information 1

2.2.2 Corequis

- Cours de base de données
- Cours de génie logiciel

Les connaissances acquises dans le cours SI2 seront par la suite mises en pratique dans le cours *Architecture des systèmes d'information*. Il s'agit d'un projet intégré (PI) réalisé en groupe afin d'acquérir les compétences de gestion de projet de SI et d'entrainer différentes compétences techniques. Ce fil rouge entre les différents cours et modules est crucial pour stimuler la *conscience d'avoir appris* chez les apprenants et leur donner envie de réutiliser les nouvelles capacités acquises.

2.3 Organisation du cours

16 séances en présentiel durant un semestre pour un total de 120 heures (4 ECTS)

La Figure 1 résume les formes d'enseignement (cours magistral et exercices et travaux de laboratoire) et les heures de travail demandées pour le travail en présence (salle de classe et laboratoire) ainsi qu'une estimation des heures de travail personnel correspondant.

Type d'activité	Dotation horaire	Travail personnel
Cours magistral et exercices	32	16
Travaux de laboratoire	32	40
Projets		
Totaux partiels	64	56
Total du volume de travail	**120**	

Figure 1. Forme d'enseignement et volume de travail.

En particulier, les travaux de laboratoire sont divisés en deux types : travaux dirigés (TD) et travaux pratiques (TP). Les **TD** sont une forme d'enseignement qui permet d'appliquer les connaissances apprises pendant les cours théoriques ou d'introduire des notions nouvelles. Les étudiants travaillent individuellement sur des exercices d'application ou de découverte, en présence d'un ou plusieurs assistants parfois accompagnés par le professeur, qui interviennent pour aider et éventuellement corriger les exercices. Dans le cours de SI2, les TD sont réalisés individuellement (les étudiants peuvent, néanmoins, discuter entre eux) et ne sont sujets à aucune forme d'évaluation.

Au contraire les **TP** sont des travaux collectifs réalisés par petits groupes de 2-3 étudiants et sont matière à évaluation, notamment une évaluation **formative**. Pour chaque TP les apprenants devront rendre le travail (souvent composé par un rapport accompagné par du code) qui sera corrigé par l'assistant ou le professeur. Finalement un feedback personnel sera donné aux étudiants via la plateforme Moodle et individuellement (pour chaque groupe) si nécessaire ; au contraire, les difficultés principales communes à tous les groupes seront discutées en classe. Cette évaluation est utilisée principalement comme moyen d'augmenter l'engagement des apprenants, et aura donc un impact, comme facteur correctif, sur la note finale du cours.

La combinaison de cours ex-cathedra et TD/TP a été conçue pour favoriser un apprentissage en profondeur des étudiants. Les travaux pratiques ont souvent une composante créative et de liberté pour les étudiants. Par exemple, les étudiants sont parfois invités à utiliser une technologie spécifique, qui est le topique de la séance, pour la création d'une application définie par eux-mêmes. Souvent, cette liberté permet aux apprenants de travailler sur un sujet d'intérêt ou de développer

un projet personnel. Ceci permet d'intervenir sur plusieurs facteurs favorisants le transfert de connaissances :

- Avoir conscience d'avoir appris
- Avoir envie de réutiliser ce qui est appris
- Connaître des domaines d'application de ce qui est acquis

2.3.1 Analyse du scénario d'enseignement hybride

		Activité 1	Activité 2	Activité 3	Activité 4	Activité 5	Evaluation d'apprentissage
En présence	**Intentions :**	- Aborder la théorie du thème que l'on aborde - Explorer les notions liées à ce thème	- Exercice pratique sous forme de travail dirigé		- Travail pratique sur le thème abordé, sous forme de petit projet (ex : création d'un site web utilisant la technologie présentée)		- Discussion au sujet de l'évaluation du projet
	Médias :	- Présentation PowerPoint au Beamer - Ressources documentaires (liens, documents PDF)	- Ordinateurs personnels - Visualisation de solutions via beamer		- Ordinateurs personnels - Utilisation de Moodle pour la mise à disposition de ressources		
	Acteurs :	**Professeur,** Etudiants	**Assistant,** Etudiants (Travail personnel)		Assistant, **Etudiants (Travail en groupe)**		Professeur, Assistant, Etudiants
A distance	**Intentions :**			- Synthétiser les notions mises en pratiques lors de l'activité précédente - Mettre en relation les notions théoriques et le cas pratique		- Finalisation du travail pratique à distance	- Evaluation du projet
	Médias :			Dossier de suivi d'auto-formation		- Utilisation de moyens de communications (Skype, Mails, Moodle, etc.)	- Utilisation de Moodle pour le rendu du projet
	Acteurs :			Etudiants (Travail personnel)		Etudiants (Travail en groupe)	Professeur, Assistant

Tableau 2: Scénario d'enseignement hybride selon l'outil ActIMA (Référence: Platteaux Hervé et Daele Amaury. ActIMA : un outil pour représenter des scénarios de cours hybrides)

Le tableau ci-dessus illustre le scénario hybride du cours. Chaque matière (thème, comme par exemple Microsoft .Net, Java Enterprise Edition, etc.) qui sera abordée durant le cours pourra être découpée et préparée selon ce scénario. Il n'y a pas d'obligation temporelle, c'est-à-dire que chaque activité peut être échelonnée sur plusieurs périodes de cours, en fonction de l'importance de cette dernière (même si une indication de temps standard est néanmoins indiquée).

Dans sa construction, il est largement inspiré du modèle de Lebrun (voir Figure 2). L'idée étant de ne pas seulement dispenser une base théorique, mais de la mettre en pratique de manière réfléchie, afin d'avoir des boucles de rétroaction visant à motiver les étudiants, et également permettre une plus grande interaction entre leurs activités et les productions issues des travaux pratiques.

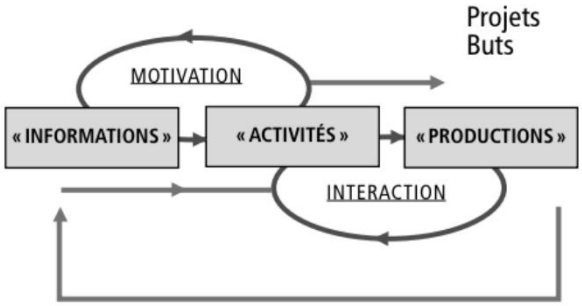


Figure 2: Modèle d'apprentissage de Lebrun (2002)

Activité 1, Informer, ~4 périodes

Cette première activité se concentre sur la partie théorique du thème abordé. Le professeur étant l'acteur principal, les élèves étant les acteurs secondaires car ils se contentent généralement de suivre le cours. La théorie est généralement présentée par l'utilisation de présentations PowerPoint, et projetée au beamer. Différentes ressources documentaires (liens sur des sites web intéressants, documents PDF, etc.) sont également mises à disposition des élèves afin qu'ils approfondissent les points abordés lors du cours.

Activité 2, Motiver/Activer, ~2 périodes

Afin de mettre en pratique les éléments théoriques vus lors de l'activité 1, un exercice pratique sous forme de travail dirigé est effectué. Le travail dirigé diffère du travail pratique dans le sens où l'assistant (et non les élèves) est l'acteur principal. En effet, l'idée est ici de mettre en pratique les notions théoriques, mais tout en accompagnant du mieux possible les étudiants, afin qu'ils prennent doucement leurs marques avec la technologie utilisée. Les étapes à réaliser sont projetées en direct au beamer par l'assistant. Cette façon de faire est motivante pour les étudiants car ils peuvent réaliser des choses sans se retrouver bloqués, et prennent ainsi rapidement connaissance des possibilités qu'ils ont avec telle ou telle technologie. C'est de ce point de vue une excellente préparation au travail pratique.

Activité 3, Motiver/Activer, ~1/2 période

Il est demandé aux étudiants de réaliser une petite synthèse concernant les notions théoriques vues lors de l'activité 1 et mises en pratique lors de l'activité 2. Cela leur permet de mettre en relation les notions théoriques et les exemples pratiques qu'ils ont réalisés. De ce fait, c'est une motivation pour l'étudiant, qui voit ainsi que les notions théoriques qui lui ont été enseignées sont réellement utilisées, donnant ainsi du sens à ce qu'ils apprennent. Cette synthèse se fait dans le dossier de suivi d'auto-formation, qui accompagne chaque étudiant durant tout le semestre.

Activité 4, Interagir/Produire, ~4 périodes

Afin de mettre à profit les connaissances théoriques ainsi que pratiques assimilées lors des activités 1 et 2, les étudiants doivent réaliser, un travail pratique sur le thème abordé, sous forme de petit projet (ex : création d'un site web utilisant la technologie présentée). Afin d'améliorer les interactions, le travail est à réaliser en binôme (groupe de deux étudiants). Cela leur permet d'organiser leur travail de manière collaborative, définir les rôles de chacun des membres du groupe, etc. L'interaction avec l'assistant est également vivement conseillée, notamment afin que les étudiants ne se retrouvent pas bloqués dans la réalisation de leur travail. Ce travail pratique est très intéressant pour les étudiants, qui se voient donner l'opportunité de produire quelque chose de concret, et d'y apporter leurs idées.

Activité 5, Interagir/Produire, ~4 périodes

Si une partie du travail pratique (activité 4) se fait en classe, une moitié du temps prévu est à réaliser à distance. Les étudiants se voient ainsi obligés de faire preuve d'une plus grande autonomie, car ils n'ont pas l'assistant « sous la main ». Ils doivent donc plus s'organiser, afin de pouvoir continuer à travailler en binôme, utiliser différents moyens de communications (Skype, mails, Moodle, etc.). C'est la partie la plus intéressante pour eux dans l'aspect collaboratif du travail.

Evaluation d'apprentissage, Produire, ~1 période

Finalement, le travail pratique réalisé par chaque groupe et déposé sur Moodle (généralement une archive contenant le code source du travail réalisé ainsi qu'un rapport de synthèse) est évalué par l'assistant (et accessoirement par le professeur). Une discussion est ensuite organisée en classe autour de cette évaluation, afin de donner un retour général aux étudiants. De plus, il est habituel que le professeur prenne quelques minutes avec certains étudiants de manière individuelle afin de clarifier certains points d'incompréhension qui auraient été soulevés dans le dossier de suivi d'auto-formation, ou encore dans le rapport de synthèse.

De plus, pour chaque thème abordé, un document indiquant la liste des objectifs définis pour ce thème est mis à disposition des étudiants. Cette liste leur permet d'évaluer s'ils ont assimilés toutes les connaissances sur lesquelles ils risquent d'être questionnés lors du test d'évaluation final.

2.4 Thèmes

Le semestre est découpé en deux. Durant la première moitié du semestre, les deux technologies principales (Microsoft .Net et Java EE) sont étudiées ensemble à une introduction aux principes fondamentaux des systèmes d'information. A la fin de la première moitié, un test oral permet d'évaluer ces matières principales. Durant la deuxième moitié, des thèmes importants du monde des systèmes informations, généralement indépendants des technologies, sont abordés (comme par exemple les services web et les RIA). Des technologies concurrentes à Microsoft .Net et Java EE orientées développement web y sont également présentées. Le MindMap suivant illustre la répartition des thèmes entre la première (semaines A01 à A08) et la deuxième moitié (semaines A09 à A16) du semestre :

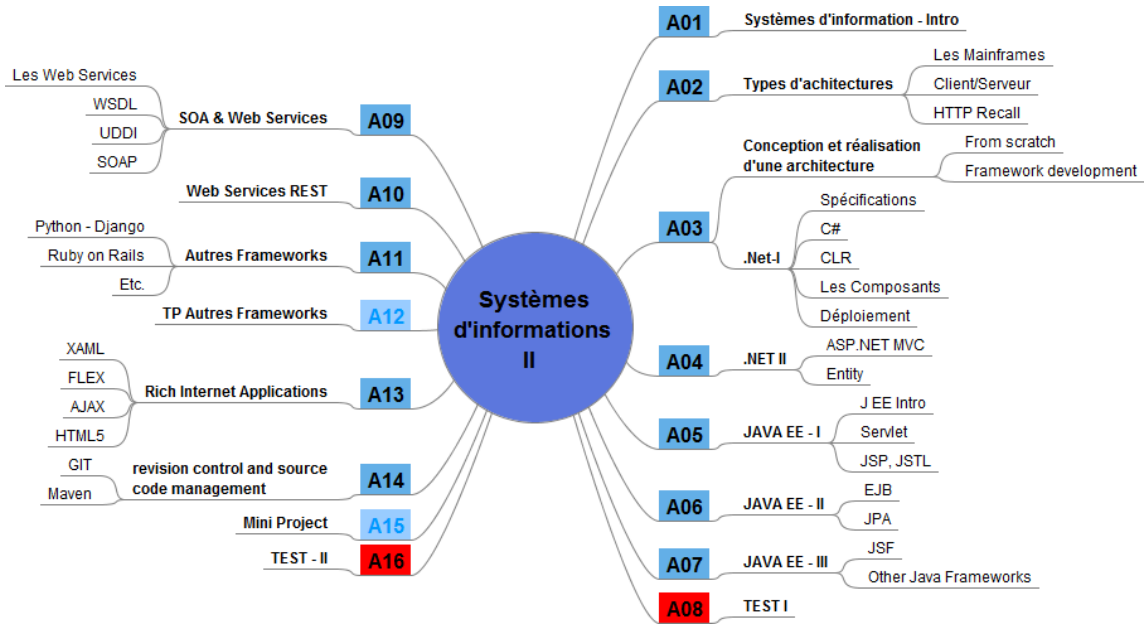


Tableau 3 : Répartition des thèmes sur le semestre (semaines A01 à A16)

2.5 Learning outcomes

A l'issue de ce cours les étudiant-e-s auront acquis les compétences suivantes :

- Installer, configurer et entretenir un serveur d'information.
- Analyser un besoin concret de système d'information et en concevoir l'architecture sur la base de modèles existants (N-tiers, réparti, etc.).
- Décrire, comparer et utiliser les technologies usuelles de présentation de l'information sur des supports fixes et mobiles.
- Appliquer et choisir les frameworks les plus pertinents pour la réalisation d'un portail d'information.
- Assumer une fonction de consultant technique auprès d'un client, dans le domaine des systèmes d'information.
- Déterminer les enjeux des nouvelles technologies, l'importance de la standardisation dans le processus d'échanges des données et dans la chaine éditoriale de l'entreprise

Objectifs [1]	
Savoir-refaire / savoir-redire	L'apprenant(e) pourra : -Décrire et comparer les différents frameworks et les technologies plus utilisés dans le domaine des systèmes d'information. -Décrire les enjeux des nouvelles technologies, l'importance de la standardisation dans le processus d'échanges des données et dans la chaine éditoriale de l'entreprise (basée sur XML, JSON,…).
Savoir-faire convergents	L'apprenant(e) pourra : -Décrire un besoin standard d'un système d'information et en concevoir l'architecture sur la base de modèles existants et rencontrés pendant le cours (N-tiers, réparti, etc.).

[1] Référence : la taxonomie de De Ketele

Savoir-faire divergents	L'apprenant(e) pourra : -Analyser un besoin concret de système d'information et en concevoir l'architecture sur la base de modèles existants non abordés directement et/ou explicitement en classe. -Appliquer et choisir les frameworks les plus pertinents pour la réalisation d'un portail d'information.
Savoir-être / savoir-devenir	L'apprenant(e) pourra : -Assumer une fonction de consultant technique auprès d'un client, dans le domaine des systèmes d'information.

Tableau 4 : Objectifs du cours SI2.

2.6 Evaluation

2.6.1 Organisation

En plus de l'évaluation formative mentionnée ci-dessus et liée aux TP, le cours prévoit deux évaluations orales, la première ayant lieu environ à la moitié du semestre et la deuxième à la fin du cours.

Les évaluations faites lors de ces examens sont sommatives. L'objectif est de mesurer les compétences que les étudiant(e)s ont développées dans le domaine des systèmes d'information. Les évaluations suivent la même approche et sont conçues à l'aide des objectifs du cours qui sont présentés oralement et par écrit aux apprenant(e)s pendant le cours (en particulier pendant la première séance nous présentons les objectifs globaux du cours, tandis que dans chaque séance sont introduits les objectifs particuliers de la séance (les deux sont constamment consultables sur le Moodle du cours).

L'examen final est composé de 2 parties :

- Partie écrite (préparation) : composée par des questions ouvertes et à choix multiple, cette partie vise à tester les connaissances de type savoir-refaire, savoir-redire et savoir-faire convergent dans la taxonomie de Ketele. L'étudiant(e) peut utiliser des documents fournis par le professeur (*quick références*) et consultables pendant tout le semestre. Le document rédigé pendant cette phase est utilisé comme support et point de départ dans la partie orale de l'examen.
- Partie orale : composée par des questions théoriques qui se portent sur tous les transparents, les TDs, les TPs, et les documents supplémentaires cette partie vise à tester les hauts niveaux de la taxonomie de Ketele, notamment : savoir-faire divergent, savoir-être (moins le savoir-devenir). L'oral démarre avec une discussion sur le document rédigé par l'apprenant pendant la partie de préparation, suivie par différentes questions et est conclu par une question pratique, sur un problème concret.

Selon les années, l'examen a une durée comprise entre 20 et 30 minutes.

Une échelle d'évaluation (ordinale) à 6 niveaux basée sur l'échelle de Bloom a été mise au point (Figure 3).

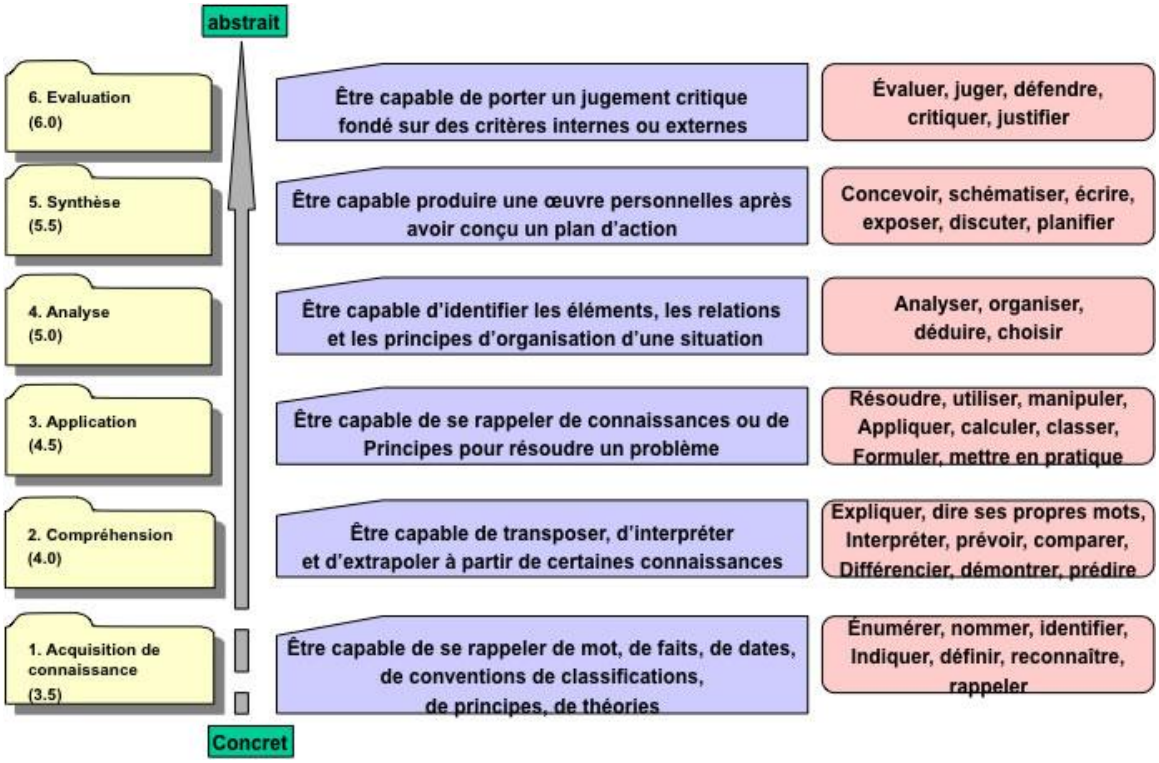


Figure 3. Echelle de Bloom.

Bien que la taxonomie de Ketele intègre le niveau des compétences, l'échelle de Bloom est également utilisable afin de créer une métrique pour l'évaluation d'un examen à réponses ouvertes. Dans le cadre de ce cours, notre choix sur le type d'échelle s'est porté sur la taxonomie de Bloom pour plusieurs raisons (aussi de type pratique). En fait la taxonomie de Bloom :

- est composée d'une hiérarchie de processus du plus simple au plus complexes que les apprenant(e)s devraient acquérir.
- fournit la structure pour définir les «learning outcomes ».
- offre une structure claire (linéaire) sur plusieurs niveaux (et une liste des verbes), que, en pratique, est facilement transposable dans une échelle ordinale à cinq-six valeurs.

2.6.2 Analyse des modalités d'évaluation

Validité

- L'évaluation orale permet de vérifier les connaissances, mais également les compétences acquises par l'étudiant lors du cours, et découlent directement des objectifs définis pour ce dernier.
- L'évaluation comportant une partie théorique mais également une partie orientée vers un cas pratique, les connaissances et compétences acquises lors des séances théoriques et des travaux pratiques sont évaluées.
- Si aucune question sur des points qui n'ont pas été vu durant le semestre n'est posée, il est en revanche courant de demander à l'étudiant de mettre en lien ses connaissances avec des cas qui n'ont pas été abordés durant le cours, afin de vérifier ses savoirs faires divergents.

Pertinence

- L'évaluation se déroule de façon à ne pas uniquement évaluer certaines connaissances ou compétences, mais bien une grande partie de celles acquises lors des cours et séances

pratiques. En effet, les questions théoriques posées à l'étudiant sur le sujet qu'il tire au sort sont un point de départ, et les trois personnes du jury peuvent ensuite réorienter la discussion afin de couvrir la majorité des points abordés en classe.

Fidélité

- La plupart des questions théoriques qui sont posées sont facilement jugeables comme « juste » ou « faux », mais ce n'est pas toujours le cas. Il est donc difficile ensuite d'avoir une fidélité « intra-juge » adéquate, car la façon dont l'étudiant justifie ses choix et réponses peut facilement être évaluée de façon subjective par le jury.
    - Comme expliqué plus en détail dans la partie « Biais », le fait de mettre en commun le choix de la note finale permet d'améliorer ce facteur.
    - Mais d'une manière générale, l'utilisation de la taxonomie de Bloom et de son échelle de notation permet d'avoir un bon point de repère afin de garantir la fidélité de l'évaluation.

Biais

- La nature même de l'évaluation (examen oral) constitue le plus grand biais. En effet, l'évaluation des réponses de l'étudiant peut devenir subjective. De plus les questions donnent souvent lieu à des réflexions, et il n'est pas simple de quantifier le degré d'exactitude des réponses (pas possible de tout le temps dire « Juste » ou « Faux »).
    - Mais le fait que le jury soit constitué de trois personnes (2 professeurs et l'assistant chargé de la partie pratique) diminue le caractère subjectif de l'évaluation. En effet, les notes attribuées par chacun sont discutées afin d'obtenir une note finale en commun accord.
- Les sujets et questions posées ne sont pas toutes les mêmes pour les étudiants. Il y a donc théoriquement des risques que des questions soient plus faciles à répondre que d'autres.
    - Mais nous faisons attention à balancer le plus possible la difficulté des questions (par exemple en mettant une question « facile » et une question « difficile » à chaque sujet).
- L'évaluation orale peut, en fonction de la quantité d'étudiants qui compose la classe, durer très longtemps, du fait qu'un seul étudiant soit évalué à la fois. D'une part, le jugement peut être un peu biaisé du fait de la fatigue du jury après plusieurs heures d'évaluation. D'autre part, l'étudiant qui passe en dernier aura eu le temps de plus se préparer, et aura également eu le loisir de discuter avec les étudiants étant déjà passé au sujet des questions qui sont posées.
    - A part boire beaucoup de café, il est difficile de trouver un moyen de limiter le biais lié à la fatigue du jury ! Une solution serait d'étaler l'interrogation sur plusieurs jours, afin d'avoir des pauses, mais cette solution n'est pas envisageable du fait des calendriers « serrés » en période d'examen, et également afin de ne pas favoriser en temps d'étude les étudiants qui passeraient en dernier.
    - Le tournus des questions est optimisé afin qu'un étudiant ayant eu une question X ne puisse pas discuter avec le prochain étudiant qui aura cette même question X.

Feedback

- Il est généralement difficile de donner un feedback aux étudiants concernant leur prestation lors de l'examen oral. Cependant, les étudiants qui le souhaitent peuvent organiser une discussion avec le professeur responsable du cours afin de déterminer les raisons de l'attribution de sa note.
- Afin de garder une trace, chaque membre du jury prend des notes lors des oraux, et ces notes sont conservées afin d'être disponibles si besoin est.
- Les étudiants qui ont obtenu une note inférieure ou égale à 4.0 doivent prendre contact avec le professeur afin d'organiser une réunion de tutorat. Durant cette réunion, l'examen

est passé en revue, et le professeur tente avec l'étudiant de déterminer les points à améliorer afin de réussir l'examen suivant. Le dossier de suivi d'autoformation peut s'avérer être utile pour cet exercice (à condition que l'étudiant l'ait rempli de manière consciencieuse).

2.6.3 Exemple concret

Chaque sujet (tiré au sort par l'étudiant) est présenté sous une forme similaire à l'exemple suivant :

Metadata

Thème : ASP.NET, ADO.NET					
Acquisition de connaissances	Compréhension	Application	Analyse	Synthèse	Evaluation
	X	x			

Enoncé

Répondez aux questions suivantes et illustrer si besoin.

1. Dans ASP.NET, qu'est-ce que le *postback*? Qu'est-ce que le *viewstate*? Comment sont-ils liés ?
2. Une Application .NET unique peut être créée en utilisant plusieurs langages de programmation. Expliquer les caractéristiques permettant à .NET de fournir cela.
3. Qu'est-ce que le *data provider* dans ADO.NET? Quelles sont les classes clés de chaque « provider »?

Tableau 5 : Exemple concret d'un sujet de l'interrogation orale

Le tableau « Metadata » permet à l'étudiant d'identifier le thème du sujet, ainsi que le type de compétences qui vont être évaluées (Compréhension / Application / Analyse / Synthèse / Evaluation). Ensuite, plusieurs questions sont posées à l'étudiant. Il peut s'agir de questions purement théoriques, mais également orientées pratique. En fonction des questions, l'étudiant peut être invité à schématiser ses idées.

3 Cours théoriques

Cette section présente l'organisation des cours théoriques en se focalisant en particulier sur les modifications que les auteurs ont apportées au cours suite à l'évaluation qu'ils ont réalisé. En fait, des changements dans ces types de cours sont nécessaires afin de fournir aux apprenants les nouvelles connaissances dont ils ont besoin pour s'intégrer rapidement dans le monde du travail. Ceci représente une tâche très intéressante pour les auteurs de ce rapport et dans le cadre du travail pour de diplôme pour Did@cTIC. Pour cette raison cette section inclut un exemple concret fait par les auteurs.

3.1 Approches

Les cours magistraux correspondent à 32 heures sur le totale de 120 heures du cours. C'est à travers les cours magistraux que la plupart des notions sont communiquées aux étudiants. Chaque cours théorique est complété par des exercices de laboratoire complémentaires.

Il est opportun de distinguer 2 typologies de cours : les cours « frameworks » (*multi-séance)* et les cours « technologie » (souvent *one shot*).

Les cours frameworks occupent principalement la première moitié du semestre et visent à traiter avec plus de détails les deux frameworks les plus importants dans le contexte des systèmes d'information : le framework Java Enterprise Edition et le framework Microsoft .NET.
Ce deux frameworks sont particulièrement intéressants car :

- Ils font partie des framework les plus utilisés aujourd'hui dans le développement de système d'informations, c'est donc intéressant pour les apprenants de connaître leur spécificités.
- Ils se basent sur des principes qui sont largement (ré)utilisés aussi dans d'autre frameworks et peuvent donc être utilisés comme base pour enseigner des solutions technologiques généralement valides et pas liées strictement à une implémentation particulière (comme exemple, voir le principe de l'Object Relational Mapping décrit dans les paragraphes ci-dessous).

Les cours technologie permettent de focaliser une séance sur un élément bien spécifique qui caractérise un système d'information. Typiquement, ce principe affecte plusieurs technologies. Par exemple, les services web sont des technologies (permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués) utilisées pas plusieurs (sinon la totalité des) frameworks pour le développement web (JEE, .NET, Django, RoR, etc.).

3.1.1 Structure d'une séance

Chaque séance est structurée en plusieurs phases qui, grosso modo, se répètent chaque semaine :

- Sommaire de la séance avec les éléments qui seront traités.
- Rappel des concepts présentés pendant les séances précédentes (seulement dans le cas de cours multi-séance).
- Entrée en matière avec une description du « problème » existant dans le monde réel lié au topic motivant l'émergence ou l'adoption d'une technologie.
- Présentation du topic de la séance.
    - Présentation de la nouvelle technologie
    - Détails technologiques
    - Discussion de comment la solution technologique présentée fait face aux points précédemment introduits
    - Discussion sur les avantages et les limites de la solution présentée.
- Eventuel deuxième topic de la séance…
- …
- Conclusion

3.1.2 Supports au cours

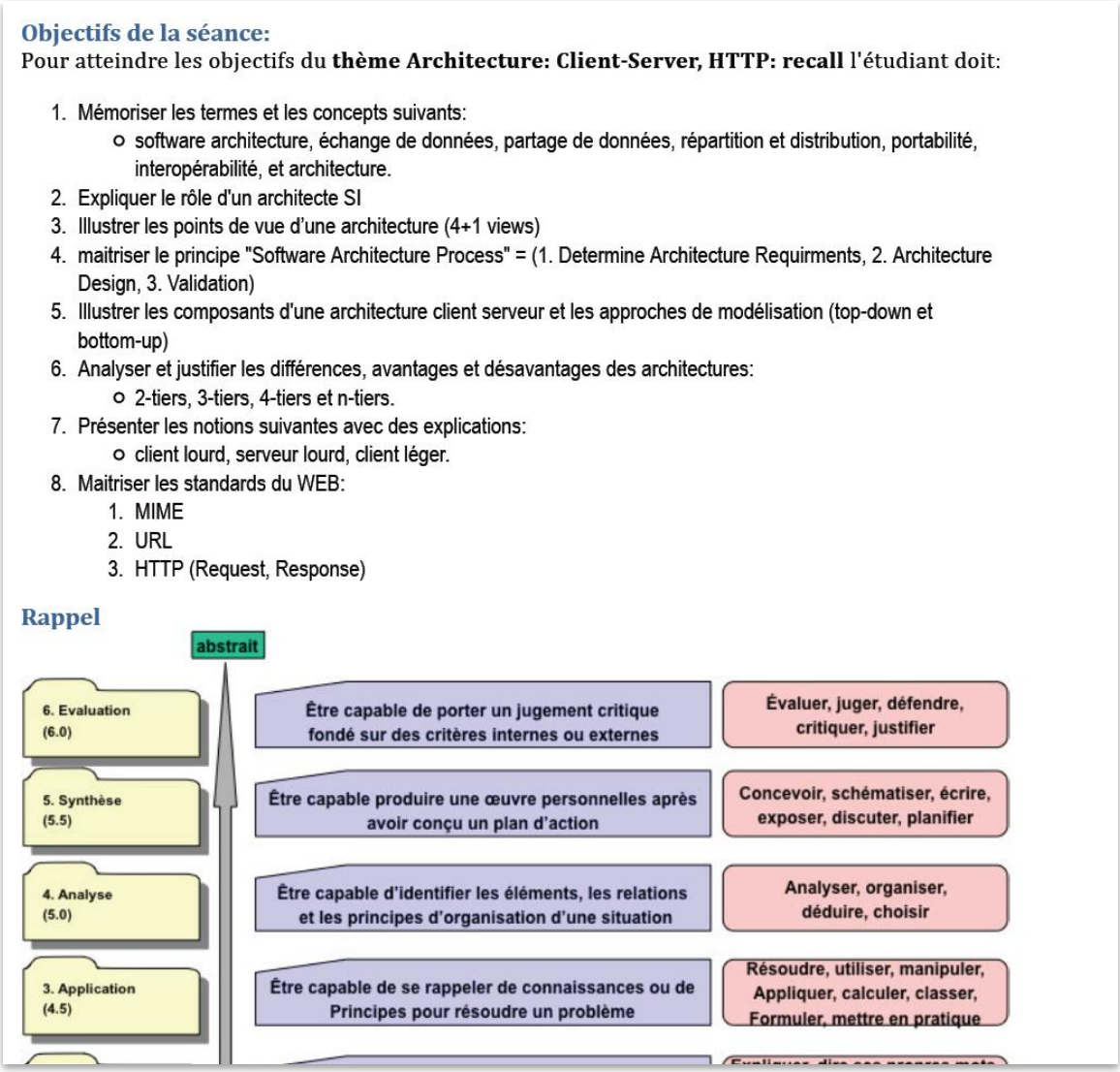
Le cours est présenté à l'aide de transparents Power Points. Vidéos, logiciels et autres ressources disponibles online sont aussi fréquemment utilisés pour montrer des cas d'utilisation ou comme démos.

Le cours est présenté en français tandis que le matériel écrit (transparents, documents supplémentaires, etc.) sont en anglais. La présence de l'anglais vise à rapprocher les étudiants à cette langue qui est nécessaire dans un domaine tel que l'informatique où les logiciels, la plupart de la documentation et des informations peuvent difficilement être restreintes entre des frontières nationales et où l'anglais est la langue utilisée par default pour le partage des connaissances.

Les apprenants utilisent la plateforme Moodle (http://cyberlearn.hes-so.ch/) pour récupérer les transparents et le matériel additionnel.

Pour chaque cours un fichier « Student should » indique clairement quel sont les objectifs de la séance et ce que les professeurs s'attendent de l'étudiant.

Comme le sont également les documents présentant les TD et TP, ce fichier est en français. La figure ci-dessous montre un extrait du fichier « student should » associé au cours *Architecture: Client - Server, http (recall)*.



Objectifs de la séance:

Pour atteindre les objectifs du **thème Architecture: Client-Server, HTTP: recall** l'étudiant doit:

1. Mémoriser les termes et les concepts suivants:
   - software architecture, échange de données, partage de données, répartition et distribution, portabilité, interopérabilité, et architecture.
2. Expliquer le rôle d'un architecte SI
3. Illustrer les points de vue d'une architecture (4+1 views)
4. maitriser le principe "Software Architecture Process" = (1. Determine Architecture Requirments, 2. Architecture Design, 3. Validation)
5. Illustrer les composants d'une architecture client serveur et les approches de modélisation (top-down et bottom-up)
6. Analyser et justifier les différences, avantages et désavantages des architectures:
   - 2-tiers, 3-tiers, 4-tiers et n-tiers.
7. Présenter les notions suivantes avec des explications:
   - client lourd, serveur lourd, client léger.
8. Maitriser les standards du WEB:
   1. MIME
   2. URL
   3. HTTP (Request, Response)

Rappel

Figure 4: Extrait du fichier « Student should » lié à la séance Architecture : Client-Serveur, http: recall.

3.2 Apport au cours : Mise à jours des contenus

Comme mentionné auparavant, un défi associé à ce type de cours est la nécessité de mises à jour continues des contenus du cours.

La figure 5 schématise le processus qui guide le changement. Ce type de schéma est typique d'une HES qui est très proche des besoins et des attentes du marché.

Figure 5. Motivation aux changements.

Afin d'expliquer l'approche dans la conception d'une nouvelle séance, un exemple concret est détaillé dans la section suivante.

3.3 Un cas concret

Cette section présente un exemple concret de mises à jour d'un cours en se focalisant sur les motivations qui ont mené à ce changement et les méthodes employées pour la création du cours. Dans la limite du possible, le document ne rentrera pas dans les détails technologiques et, où cela semble nécessaire, les auteurs introduirons une brève et souvent générique explication des concepts mentionnés.

3.3.1 D'ADO.NET classique aux solutions ORM et l'ADO.NET Entity Framework

Sans s'attarder trop sur les détails technologiques, il faut savoir qu'ADO.NET est un framework (un ensemble de classes) dans le monde Microsoft .NET. ADO.NET a pour but de faciliter les développeurs dans la création d'une interface entre la logique d'une application et les données que l'application nécessite et qui sont souvent stockées dans une base de données relationnelle (note : *relationnelle* signifie que les données sont sauvegardées sous forme de tableaux à deux dimensions, appellés tables, et que les relations entre les données sont maintenues avec des liens entre les tables dans la base de données).

Le cours ADO.NET faisait donc partie du cours « frameworks » traitant le topic Microsoft .NET.

Pour un ingénieur en informatique, ce framework est intéressant d'un point de vu pédagogique pour 3 raisons principales :

- ADO.NET permet d'abstraire le développeur du type et de la technologie de base de données utilisés. Avec le grand avantage que le développeur ne doit pas trop se soucier de quel type de technologie de base de données est utilisée dans l'application.
- ADO.NET fournit un ensemble de fonctionnalités qui permettent de calibrer l'accès à une base de données selon l'utilisation dont le logiciel a besoin (lecture rapide, beaucoup d'utilisateurs, etc.).
- ADO.NET permet aussi d'accéder à des données qui ne sont pas dans des bases de données relationnelles (fichier XML, objets, …).
- Pendant plusieurs années celui-ci a été l'approche standard pour les développeurs utilisant la technologie Microsoft.

Comme mentionné auparavant, une base de données relationnelle permet d'organiser des données sous forme de tableaux et relations entre différentes tables. Par contre aujourd'hui, la plupart de langages de programmation ne travaillent pas avec des tables mais plutôt avec des « objets ». Un objet représente un concept, une idée ou toute entité du monde physique, comme une personne,

une voiture, une facture ou encore une page d'un livre. Il possède une structure interne et un comportement, et il sait interagir avec ses pairs. La programmation orientée objets fonctionne via l'utilisation de ces objets et leurs relations ; l'interaction entre les objets via leurs relations permet de concevoir et réaliser les fonctionnalités attendues d'une façon conceptuellement simplifiée. On peut comprendre que cette représentation par objets, qui facilite énormément le travail du développeur, est en quelque sorte en friction avec une représentation de l'information par tables qui est une forme moins riche de représentation de l'information.

Surgit donc le problème de coupler la logique d'une application (orienté objets) et les données d'une application (sous forme de tables).

La solution à ce *mismatch* a été pendant plusieurs années de laisser au développeur le soin d'adapter les deux mondes en convertissant les objets en tables pour les sauvegarder dans une base de données et, vice-versa, transformer tables en objets au moment de les récupérer.

Une deuxième solution conçue par les développeurs a été de créer des bases de données orientées objets. Cependant, cette solution n'a pas (pour le moment) eu beaucoup de succès principalement pour 2 raisons :

- la plupart des données qui aujourd'hui existent sont stockées dans des bases de données relationnelles (et ce serait donc très couteux de les restructurer pour un autre type de base de données)
- les systèmes de gestion des bases de données relationnelles sont extrêmement performants (grâce aux travaux d'amélioration qu'il y a eu pendant plusieurs années).

C'est pour ces raisons que, dans les dernières années, une nouvelle approche s'est peu à peu répandue jusqu'à devenir la solution la plus utilisée : l'*Object Relational Mapping* (ou plus brièvement ORM).

Sans décrire la technologie en arrière-plan, un ORM automatise la conversion d'un objet et ses relations en un ou plusieurs tableaux dans une base de données relationnelle (et vice-versa de la base de donnée relationnelle à objet).

Ceci signifie que le développeur est soulagé de cette tâche et il ne doit pas lui-même définir comment un objet est transformé mais la transformation est faite automatiquement. Tâche qui est maintenant faite automatiquement sans impacter les performances des bases de données relationnelles qui sont maintenues.

L'avènement de cette technologie signifiait pour le cours de système d'information de décider si il fallait continuer à proposer l'approche classique ADO.NET (encore très utilisée) ou plutôt passer à la nouvelle approche ORM.

3.3.2 Evaluation de l'impact des changements sur le cours

Pour évaluer l'impact d'un éventuel changement sur le cours de SI, nous avons comparé plusieurs facteurs qui sont représentés dans le tableau ci-dessous.

Paramètres de choix	ADO.NET (Classique)	ORM
Multiple technologies	✗	✓
Diffusion sur le marché	✓	✓ ✓
Probable impact dans l'avenir des systèmes d'information	✗	✓
Types différents de données	✓	✗
Intégration dans le flux du cours	✓	✗

Tableau 6: Comparatif entre ADO.NET classique et les solutions ORM.

Comme on peut remarquer en observant le tableau, les deux technologies sont intéressantes dans le cadre du cours de SI.

- **Multiple technologies :** plusieurs frameworks de développement proposent une solution ORM. Tandis que la solution d'ADO.NET classique est strictement liée à la plateforme .NET avec peu de rechutes sur les autres technologies.
- **Diffusion sur le marché :** la technologies ADO.NET est très utilisée mais reste limitée aux applications développée avec la technologie .NET. D'autre coté les solutions ORM sont de plus en plus répandues.
- **Probable impact dans l'avenir des systèmes d'information :** puisque de plus en plus le nouveau framework de développement propose par défaut une solution ORM est très probable que dans le futur les applications seront développées en suivant cette approche.
- **Types différents de données :** ADO.NET fourni une solution très intéressante pour permettre aux développeurs de travailler avec de diffèrent types de données (objets, fichier XML et BD relationnelles) d'une façon très simplifié, cette solution s'appelle ***LINQ***.
- **Intégration dans le flux du cours :** le cours sur ADO.NET est présenté aux apprenants dans la série de séances traitant le framework Microsoft .NET et ses composantes et il était donc bien intégré dans le fil rouge du cours.

En d'autres termes, intégrer le cours ORM implique l'introduction de nouvelles connaissances mais engendre la perte d'autres que nous estimons encore intéressantes.

Donc, pour intégrer correctement ces nouveaux contenus dans le cours de systèmes d'information il est avantageux sinon nécessaire de mitiger les « croix rouges ». Une solution existe et ceci est l'*ADO.NET Entity framework*.

3.3.3 ADO.NET Entity Framework

L'ADO.NET Entity Framework ou Entity Framework ou encore EF est la solution ORM proposée par Microsoft. Comme on peut deviner du nom, l'ADO.NET Entity Framework a l'avantage de se baser sur ADO.NET (et LINQ).

Cela signifie qu'en présentant l'Entity Framework aux apprenants cela nous permet d'introduire le concept d'ORM sans négliger complétement ADO.NET et prendre, en quelque sorte, le meilleur des deux mondes.

Le tableau ci-dessous ajoute au comparatif la colonne ORM – Entity framework.

	ADO.NET (Classique)	ORM	ORM - Entity Framework
Multiple technologies	✗	✓	✓
Diffusion sur le marché	✓	✓ ✓	✓ ✓
Probable impacte dans l'avenir des systèmes d'information	✗	✓	✓
Types différents de données	✓	✗	✓
Intégration dans le flux du cours	✓	✗	✓

En présentant d'abord le concept d'ORM, il est possible d'introduire d'abord des principes généraux et ensuite se focaliser sur une technologie spécifique (l'Entity Framework). Ceci permet d'introduire l'ORM comme une solution valide pour différentes technologies et répandues dans le marché et avec un grand impact sur les SI du futur. En décrivant la solution spécifique, l'Entity Framework a l'avantage de se baser sur ADO.NET pour certains mécanismes et d'intégrer LINQ (il fournit donc la possibilité de travailler avec différents types de données). Finalement, l'EF étant une solution Microsoft, ceci permet de l'intégrer dans le cours sans devoir restructurer l'organisation des différentes séances.

3.3.4 Structure de la séance "Object Relational Mapping and the Entity Framework"

Grace aux éléments discutés auparavant l'organisation de la séance a été extrêmement simplifiée et très naturellement nous avons organisé la séance selon la structure qui suit :

- Introduction – rappel de la séance précédente
- Motivation – *mismatch* entre objets et DB relationnels
- Solution – Object Relational Mapping
- Implémentation technologique – le ADO.NET Entity Framework
- Discussion – Avantages et désavantages des ORM
- Motivation (2) – Nécessité de travailler avec différents types d'entités
- Solution – EF et LINQ
- Discussion – Avantages et désavantages
- Conclusion – Rappel des éléments clés de la séance

3.4 Impact

Bien évidemment, le temps d'une séance étant limité ce n'est pas possible d'introduire de l'information sans devoir enlever du contenu.

Dans l'exemple proposé, nous avons décidé de négliger les aspects techniques les plus détaillés d'ADO.NET. Cela veut dire que les apprenants ont une connaissance moins pointue de la technologie .NET mais une vision plus large d'un concept très important comme l'Object Relational Mapping.

Ceci respecte pleinement les objectifs de ce cours : fournir de connaissances fondamentales sans trop rentrer dans les spécificités technologiques.

4 Travaux Pratiques

La partie pratique de ce cours est un challenge permanent. Les technologies présentées dans la partie théorique doivent être prises en main très rapidement dans la partie pratique. En effet, chacune de ces technologies pourrait faire l'objet d'un cours complet, mais seulement quelques heures sont à disposition ! De plus, les librairies ainsi que les environnements de développement qui sont utilisés sont mis à jour régulièrement, ce qui demande une grande réactivité de la part de l'assistant qui prépare les travaux pratiques pour constamment adapter les exercices ainsi que les supports de cours.

Ce chapitre détaille la partie pratique. Les approches et modalités d'enseignement qui sont utilisés y sont détaillées et discutées. Une partie des modifications qui ont été apportée afin d'améliorer cette partie du cours y sont présentées. Un cas concret est également présenté afin d'illustrer les modifications apportées à un thème et en analyser l'impact. Finalement, quelques propositions d'amélioration pour le futur sont formulées.

4.1 Modalités d'enseignement

4.1.1 Approche en surface : « Hello World »

L'enseignement pratique est donné avec une approche en surface, c'est à dire que les technologies abordées durant la partie théorique ne font l'objet que d'une prise en main lors de la session pratique. En effet, la plupart du temps, les exercices réalisés par les étudiants sont de type « Hello World »[2]. Un programme de type « Hello World » consiste à faire la démonstration rapide d'une technologie ou d'un langage de programmation à but pédagogique, afin de facilement mettre en place la base d'un programme et vérifier son bon fonctionnement.

Il serait évidemment très intéressant de pouvoir pousser l'apprentissage de chaque technologie en profondeur, mais malheureusement le temps à disposition ne le permet pas. En effet, chaque technologie pourrait à elle seule faire l'objet d'un cours d'un semestre !

Cependant, les deux technologies principales qui sont vues durant ce cours que sont Microsoft .Net et Java EE donnent lieu à plusieurs séances, qui occupent toute la première moitié du semestre.

4.1.2 TD et TP

La dotation horaire de la partie pratique est de 32 heures, accompagné d'un travail personnel à hauteur de 40 heures. Elle est organisée en deux types de travaux : les travaux dirigés (TD) et les travaux pratiques (TP).

Le but des TD est la prise en main des outils et des techniques abordées durant la partie théorique. Durant les TD, l'assistant montre une partie de la résolution des exercices en live au projecteur, en général pose des questions ou donne des instructions, et laisse ensuite le temps aux étudiants d'essayer de résoudre l'exercice par eux-mêmes. Finalement, les solutions sont montrées par l'assistant au projecteur.

[2] https://fr.wikipedia.org/wiki/Hello_world

Le but des TP est de permettre aux étudiants de mettre en pratique les concepts et techniques abordés durant le cours théorique, tout en allant plus en profondeur dans le thème que lors des TD. L'assistant met à disposition la donnée du travail sur Moodle, et fait une brève introduction du travail afin de clarifier ce qui est attendu de la part des étudiants. Ensuite, les étudiants ont les heures allouées en classe (normalement 4heures) plus une semaine pour la réalisation. Les TP se font en binôme, ce qui permet aux étudiants de s'habituer au travail d'équipe et ils doivent ainsi se responsabiliser pour la répartition des tâches.

4.1.3 Evaluation

Les TD ne sont pas directement évalués et ne donnent lieu à aucun facteur correctif contrairement aux TP. Une partie des solutions sont montrées directement au projecteur par l'assistant durant le travail. Les solutions complètes de chaque TD sont mises à disposition des étudiants sur la plateforme Moodle une semaine après la fin de la session. Cela permet aux étudiants n'ayant pas terminé un ou plusieurs exercices de le faire en dehors des heures de cours, sans être tentés d'immédiatement accéder aux solutions. Même s'il n'y a pas de contrôle direct du travail effectué, il est rappelé aux étudiants qu'il est nécessaire de réaliser le travail, car ils pourront être interrogés sur cette matière lors des interrogations orales. De plus, nous les encourageons fortement à nous solliciter durant les heures allouées aux travaux pratiques ainsi qu'en dehors, afin de pallier à toute incompréhension.

Une évaluation formative est cependant effectuée par le biais des travaux pratiques. A la fin des séances de travaux pratiques, l'étudiant doit rendre son travail, ainsi qu'une synthèse. L'enseignant corrige ensuite chaque rendu, et lors de la séance suivante, un feedback est donné à l'étudiant, ce qui engage généralement une discussion. Cela permet à l'étudiant de rester au clair avec ce qu'il doit apprendre tout au long du semestre, et de corriger les incompréhensions si nécessaire. L'évaluation de chaque TP donne lieu à un facteur correctif, aussi appelé « ajustement », selon la tabelle suivante :

Absence non justifiée, tricherie de toute nature	-1.5
Prestation insuffisante	-0.5
Prestation correcte	0
Prestation exceptionnelle	+0.5

Tableau 7: Tabelle de facteurs correctifs pour les travaux pratiques

Les facteurs correctifs sont pris en compte dans la formule de calcul de la note finale du module de la manière suivante :

$$Note\ cours = \frac{(test\ oral\ 1 + test\ oral\ 2)}{2} + moyenne(facteurs\ correctifs)$$

Exemple : Un étudiant ayant une moyenne de 5 (par exemple 4.5 au test oral 1 et 5.5 au test oral 2) et un facteur correctif de +0.1 aura donc 5.1 comme note finale de module.

L'intégration de ce facteur correctif dans la formule de calcul de la note du module est prévu de manière à ce qu'il n'ait qu'un impact modéré sur le résultat final. C'est un moyen de motiver l'étudiant à bien travailler, sans qu'il soit trop pénalisé ou au contraire trop avantagé par rapport aux autres étudiants. Le tableau suivant reporte les résultats des TP d'une classe, comprenant quatre TP et un mini-projet (MP) :

TP 01	TP 02	TP 03	TP 04	MP	Facteur correctif
TB	AB	S	I	S	**-0.1**
AB	TB	AB	I	TB	**-0.1**
AB	AB	AB	S	TB	**0**
TB	S	TB	AB	E	**0.1**
E	E	S	TB	TB	**0.2**
AB	AB	AB	TB	TB	**0**
TB	AB	AB	TB	TB	**0**
AB	AB	AB	S	TB	**0**
TB	AB	AB	TB	TB	**0**
E	E	TB	TB	E	**0.3**
AB	AB	AB	TB	E	**0.1**
TB	S	TB	AB	E	**0.1**
AB	AB	AB	TB	TB	**0**
AB	AB	AB	TB	TB	**0**
TB	AB	AB	S	S	**0**
TB	AB	AB	S	S	**0**
TB	AB	S	I	S	**-0.1**
AB	AB	AB	AB	TB	**0**
E	E	TB	TB	E	**0.3**
E	E	S	TB	E	**0.3**

Tableau 8 : Report de la table de notation des TP d'une classe

On peut noter que les meilleurs étudiant ont eu un facteur correctif de +0.3, les moins bons -0.1, et la moyenne des facteurs correctifs est de 0.055 sur un total de 20 étudiants.

4.2 Modifications apportées

4.2.1 Motivations et défis

Des modifications sont nécessaires afin d'assurer une amélioration continue de la partie pratique. Il faut notamment apporter plus de continuité dans l'enchainement des sujets.

Le défi principal dans ce processus de modification consiste à garder une approche en surface, tout en rendant le contenu suffisamment intéressant pour les étudiants, leur donnant envie de s'intéresser de plus près (en dehors des heures de cours) à certaines des technologies présentées. En effet, le problème principal de ce cours réside dans le fait qu'une multitude de technologies différentes soient présentées, et que chacune de ces technologies pourrait faire à elle seule l'objet d'un cours complet ! Or, seules 4 heures uniquement sont allouées par session pratique.

Finalement, il est primordial de toujours bien observer les commentaires et remarques des étudiants et les prendre en compte dans la mesure du possible afin d'avoir une partie pratique qui réponde à leurs attentes.

4.2.2 Modifications annuelles

Chaque année, de nouvelles versions des technologies utilisées sont disponibles, de même que de nouvelles versions des IDE (environnements de développement). Il faut alors faire le choix de d'adopter ces nouvelles versions ou non, et dans le cas d'une adoption, il faut également décider si le support de cours doit être adapté ou non (adapter les textes et les captures d'écran). Souvent, les modifications sont minimes, et une modification du support de cours n'est pas obligatoire. Cependant, lorsqu'il faut le mettre à jour, cela demande un certain investissement, notamment pour refaire les captures d'écran.

De plus, d'années en années, il faut décider du sort de certaines technologies qui commencent sérieusement à dater. Si continuer à les enseigner dans la partie théorique du cours est tout à fait

indiqué, il n'en est pas forcément le cas pour la partie pratique. Certaines technologies sont donc soit retirées soit remplacées. Il s'ensuit en général une discussion avec le responsable du cours pour évaluer le maintien ou non de sa partie théorique. Par exemple, nous pouvons citer ici les technologies suivantes, qui ont été retirées de la partie pratique au fil des années car jugées vieillissantes ou inadéquates avec la ligne directrice du cours : InfoPath, XQuery, ADO.Net, ASP.Net, Rich Faces, Silverlight.

4.2.3 Modifications spécifiques

Other frameworks

Le thème « PHP » a été remplacé par le thème « Other Frameworks ». Plutôt que d'introduire uniquement le langage PHP, plusieurs langages de développement web sont abordés, en s'appuyant sur leur Framework respectif le plus populaire. Ce changement particulier est détaillé dans le chapitre *4.3-Un cas concret : « Other* frameworks *»*.

RIA

Le thème « Rich Internet Application » a été passablement remanié. Tout d'abord, un tutoriel très intéressant sur la création d'une application web de gestion de vidéothèque a été utilisée, afin de mettre en avant les capacités incroyables proposées par les différents outils et assistants intégrés à Microsoft Visio pour le développement Silverlight en utilisant des services WCF (Windows Communication Foundation) et l'ORM EF (Entity Framework). Si ce tutoriel a été très apprécié des étudiants, tout n'a pas été idéal. En effet, chaque année certains étudiants rencontraient des problèmes liés aux versions des outils utilisés, empêchant ainsi leur apprentissage de la technologie ce qui n'était pas du tout optimal. De plus, Microsoft ayant annoncé un abandon quasi total de Silverlight pour un futur proche, nous avons dû nous montrer réactif et remplacer ce travail.

Notre choix s'est porté sur le Framework Javascript AngularJS. En effet, nous avons noté une tendance à l'éradication des technologies basées sur des plugins (Java applets, Microsoft Silverlight, Adobe Flash, etc.), et un recentrement sur HTML5 qui se base sur HTML, CSS3 et Javascript. AngularJS va donc dans ce sens car il est un framework Javascript totalement indiqué pour le développement d'applications HTML5. De plus, le projet étant supporté par Google, cela donne est gage de pérennité et d'une maintenance active. Cette nouvelle matière a été enseignée cette année pour la première fois, et elle a reçu un accueil extrêmement positif de la part des étudiants. En effet, non seulement ils étaient élogieux dans leur commentaires durant les travaux pratiques, mais également nombre de groupes ont décidé d'utiliser cette technologie dans la réalisation du min-projet.

Mini-projet

Suite à la suppression du travail pratique sur le sujet ADO.Net, il nous restait une séance de libre (voir même deux séances de libre en fonction de notre organisation). Nous avons eu l'idée de consacrer deux séances à un mini-projet. L'idée de ce mini-projet a été de donner aux étudiants la possibilité d'utiliser une ou plusieurs technologies vues durant le semestre afin de réaliser une application de leur choix. Cela permet aux étudiants d'aller plus en profondeur sur une ou plusieurs technologies de leur choix. L'idée était également de leur donner plus de liberté, car en règle générale les TP sont assez directifs.

Même si les étudiants avaient une grande liberté, nous avons cependant posé quelques conditions afin de garantir une quantité de travail cohérente entre les étudiants. Les contraintes suivantes ont ainsi été imposées aux étudiants :

- Utiliser une (ou plusieurs) **technologie/langage** de développement vu au cours SI-2
  - .Net
  - JavaEE
  - HTML5
  - Php Zend / Ruby on Rails / Django
- Respecter le modèle **MVC**
- Stocker les données dans une base de données
- Mettre en place (exposer) et utiliser (consommer) au minimum un **service web** (SOAP ou RESTful)
- L'application devra être **déployée** sur un serveur, et être **accessible** par les enseignants (soit hébergement personnel si à disposition, soit simplement sur une machine de la salle D20.22). Les instructions d'accès devront être indiquées dans le rapport.

Quant au choix de l'application (ou jeu), il devait être soumis pour validation auprès des enseignants.

Pour l'évaluation de ce mini-projet, les étudiants devaient :

- Réaliser une présentation live de leur application aux enseignants (démonstration des fonctionnalités, explication des choix technologiques et architecturaux, etc.)
- Réaliser un rapport présentant le travail (justification des choix d'architecture, technologies, langages, etc.)

Le feedback de la part des étudiants (formulé dans la conclusion du rapport) a été très positif. La plupart des étudiants relevant la possibilité de choisir eux même l'application ainsi que les technologies de développement.

Au final, ces mini-projets permettent de vérifier la capacité des étudiants à architecturer une application, faire les choix judicieux quant aux technologies de développement et les défendre, qui sont les compétences visées dans la description du module.

4.3 Un cas concret : « Other frameworks »

Un cas concret a été choisi afin d'analyser un des nombreux changements effectués : le thème « Other frameworks ». Ce thème aborde les plateformes de développement web les plus utilisées et qui ne sont pas basés sur Microsoft .Net Framework et Java Enterprise Edition.

La partie pratique précédemment dispensée s'articulait autour d'un tutoriel extrêmement directif sur le langage PHP. L'utilisateur devait simplement suivre une série d'étapes, qui consistait simplement à copier-coller des fichiers afin de créer de manière itérative un petit blog. Bien que le résultat fût plutôt intéressant, la participation de l'étudiant était beaucoup trop passive. Il n'avait en effet pas besoin de trop réfléchir, ni coder quoique ce soit, tout était déjà prêt dans des dossiers numérotés en fonction du numéro d'étape du tutoriel.

4.3.1 Evolution itérative

2012-2013

La première modification apportée a été de remplacer la technologie PHP par Python. Afin de rentre le travail plus intéressant, nous avons fait le choix d'utiliser un framework ce qui facilite grandement certaines opérations dans le processus de développement d'une application web. Cela permet aux étudiants de faire plus dans le même temps. Plusieurs frameworks étant disponibles (les trois principaux étant Django, Flask, Pyramid, etc.) pour le langage Python, et près avoir comparé leurs

fonctionnalités[3], nous avons retenu Django, car c'est le plus connu et le plus complet de tous, et c'est le framework conseillé lorsque l'on débute.

	Popularité	Fonctionnalités	Ecosystème	Facilité
Django	+++	+++	+++	++
Flask	+	++	+	+++
Pyramid	+	+++	+	+

Tableau 9: Comparatif des trois principaux frameworks Python

Il est généralement conseillé (mais pas obligatoire) de développer des applications Python sur un système Unix (Linux, Mac OS X, etc.). Cependant, pour faciliter la mise en place de l'environnement de développement, nous avons décidé de réaliser le travail sous Windows (les étudiants ont forcément Windows installé sur leur portable personnel, ce qui n'est pas vrai pour Linux ou Mac OS X). Pour faciliter le déploiement, nous avons opté pour un plugin Visual Studio permettant d'installer facilement une pile de développement pour Django[4].

Finalement, le travail consistait à développer une petite application web permettant de gérer une liste de tâches.

Si les étudiants ont donné un feedback globalement positif quant à ce travail, ils ont relevé le fait qu'ils auraient également aimé aborder les deux autres technologies de développement web qui font partie du cours théorique : PHP et Ruby.

2013-2014

Afin de répondre aux attentes des étudiants, nous avons décidé de remanier le TP, en intégrant également les technologies PHP et Ruby. Nous avons opté pour les frameworks les plus utilisés pour ces deux technologies : PHP Zend et Ruby On Rails.

L'idée du TP était de laisser choisir à chaque binôme le framework qu'il allait utiliser pour réaliser le TP. La seule condition était qu'il fallait au moins un binôme pour chaque framework. Ensuite, le travail était similaire pour chaque binôme, indépendamment de leur framework : suivre le tutoriel de base (présent sur le site web de chaque framework), puis réaliser l'application de gestion de liste de tâches.

Finalement, nous avons demandé à chaque binôme de faire une présentation à la classe d'un résumé du framework qu'ils ont utilisé, ainsi que du résultat de l'application qu'ils ont développé. De cette manière, les binômes n'ayant pas utilisé le framework présenté pouvaient avoir un aperçu de ses caractéristiques ainsi que l'avis des binômes l'ayant utilisé.

Dans cette version, le TP a reçu de très bonnes critiques de la part des étudiants. Cependant, ils ont trouvé que le fait de répéter les présentations sur les mêmes sujets n'était pas une bonne solution. Effectivement, chaque groupe traitant du même framework avaient des présentations très similaires.

[3] https://fr.wikipedia.org/wiki/Comparaison_des_frameworks_d%27applications_web#Python_2

[4] http://www.microsoft.com/Web/webmatrix/v2/

2014-2015

Comme lors de la précédente évolution du TP, nous avons essayé de répondre au mieux aux critiques des étudiants. C'est pour cette raison que nous avons opéré deux principaux changements :

- Tout d'abord, nous avons fait en sorte que le choix des frameworks soit bien balancé, cela permet d'avoir des groupes de travail de tailles similaires pour la partie présentation.
- Pour la partie présentation, nous avons demandé aux étudiants de se regrouper par framework, et de préparer et présenter ce dernier en groupe, de façon à éviter les répétitions.

Au final, le feedback des étudiants a été très positifs concernant ce travail.

Remarque : les données de ces TP dans ses versions 2011-2012, 2012-2013 et 2014-2015 sont en annexe.

4.3.2 Impact

L'impact de ces modifications sont jugées au moyen des facteurs apports et limites.

Apports

- Non pas 1 mais 3 technologies sont abordées dans la partie pratique.
- Plutôt que de réinventer la roue en recodant tout de zéro, on profite des outils offerts par des frameworks, ce qui permet d'être plus efficace et ainsi faire plus en autant de temps.
- La présentation finale qui se fait par groupe de travail permet aux étudiants de d'essayer à la mise en commun en groupe, ce qui change de leur habituel binôme.

Limites

- Même si 3 technologies sont abordées, les étudiants ne peuvent se familiariser par la pratique qu'avec une des 3.
- Il faut s'assurer que la présentation finale des groupes est correcte, afin de ne pas induire en erreur les autres groupes.
- Le fait de demander aux groupes une présentation finale demande de réserver une session de plus (ce qui n'est pas le cas normalement avec les TD ou TP), ce qui nécessite une certaine réorganisation du planning.

4.4 Propositions pour le futur

D'une manière générale, il serait très intéressant d'amener plus de continuité entre les différents travaux pratiques. Par exemple, il serait envisageable de créer des TD et TP qui utiliseraient les applications créées dans les TD et TP précédents. Cette succession de petites applications qui s'utilisent pourraient aboutir à une application de plus grande envergure que les « Hello World » habituels, ce qui serait plus valorisant pour le travail des étudiants. Cependant, mettre en pratique une telle proposition demanderait un réel effort pour l'assistant, et ça ne serait pas une tâche facile.

Une autre proposition serait de demander à des étudiants de présenter leur solution lors des travaux dirigés plutôt que de la montrer tout de suite. Cela permettrait de s'assurer que les étudiants fassent bien leur travail. Car, durant les sessions pratiques, l'assistant n'endosse en aucun cas le rôle du policier, et même si il joue sur un climat de confiance, certains étudiants rechignent toujours à mettre la main à la pâte et attendent patiemment que la solution leur soit livrée sur un plateau ! Cette idée serait également très intéressante dans une idée de partage, car en informatique on a souvent plusieurs approches possibles pour la résolution d'un problème, et il n'est pas rare que l'enseignant ou l'assistant apprenne de leurs étudiants.

Conclusion

Ce travail nous a permis de prendre le temps d'analyser en profondeur le cours « Systèmes d'informations II », dans tous ses aspects, aussi bien sur le plan théorique que pratique. Grâce à notre formation Did@cTIC, nous disposions des outils adéquats pour réaliser cette analyse, et également engager les réflexions nécessaires afin de trouver des idées d'amélioration adéquates.

Il nous a également permis de prendre un peu de recul sur nos activités liées au cours de SI2. Ceci avec le grand avantage de revoir sous une forme bien structurée, pas seulement nos interventions, mais aussi le cours dans sa globalité. En particulier, cela nous a permis de formaliser les étapes nécessaires pour l'apport de modifications dans le cours, en ce qui concerne les cours ex cathedra et les travaux pratiques.

Dans ce processus de rédaction, nous avons également pu remarquer les éventuelles limites et lacunes présentes dans nos approches. Nous avons par exemple observé qu'un feedback structuré de la part des étudiants nous aurait aidé dans une évaluation quantitative des modifications apportées au cours (note : des évaluations génériques sur le cours existent). En fait, nous avons eu des interviews, sous forme de dialogue avec la classe et personnellement avec des étudiants pour avoir un feedback direct, mais nous n'avons pas formalisé ces activités à l'aide d'un questionnaire par exemple.

La réalisation d'un tel travail ouvre également de nouvelles perspectives pour les autres cours et travaux pratiques dont nous sommes en charge. En effet, l'approche générale est similaire pour les différents cours d'informatique auxquels nous participons en qualité de chargé de cours ou d'assistant. Dès lors, les mêmes analyses peuvent être réalisées et les mêmes outils didactiques peuvent être utilisés afin d'opérer des changements constructifs également dans ces autres cours, comme nous l'avons fait pour ce cours « Systèmes d'informations II » durant cette formation Did@cTIC. Il faudra cependant veiller à prendre en compte tous les paramètres afin de sélectionner judicieusement les idées transférables. Par exemple, l'idée du mini-projet serait difficilement applicable en l'état dans un cours en deuxième année, les étudiants n'étant encore pas suffisamment autonomes pour ce type d'exercice. Il faudrait alors revoir la construction du min-projet et l'adapter au mieux, en donnant par exemple moins de liberté dans les choix des étudiants, et en leur apportant plus de support.

Le fait de réaliser ce travail en binôme était une évidence, étant donné la complémentarité de nos rôles dans le cadre du cours traité dans ce travail. De la même manière que notre collaboration est essentielle dans la façon de combiner parties théoriques et pratiques lorsque nous préparons et donnons le cours, cette même collaboration nous a permis de réaliser un travail de diplôme didactique homogène. En effet, tout au long du travail, nous avons pu chacun apporter des idées complémentaires à celles de l'autre autant concernant la structuration du travail mais également au niveau des idées d'améliorations possibles à appliquer au cours.

Puisque nos activités dans ce cours se poursuivront ces prochaines années, nous sommes convaincus que ce que nous avons appris au travers de ce travail nous facilitera nos interventions dans le cours.

Haute Ecole d'ingénierie et d'Architecture de Fribourg

Filière Informatique

Formation Did@cTIC

Travail de diplôme

Amélioration du cours Système d'Informations II

Annexes

A. TP Other Frameworks v.2011-2012 *(TP/TD RIA)*
B. TP Other Frameworks v.2012-2013 *(TP 11 – Web framework)*
C. TP Other Frameworks v.2014-2015 *(TP – Other Frameworks)*

Stefano Carrino & Joël Dumoulin
Août 2015

TP/TD RIA

PHP, AJAX

Stefano CARRINO, Joël DUMOULIN, Omar ABOU KHALED

1 Buts du travail

Ce travail pratique a pour but de se familiariser avec le langage PHP pour créer une application riche à l'aide des appels en AJAX et de la présentation améliorée grâce aux librairies JavaScript Prototype et Scriptaculous.

2 Ressources

Les ressources suivantes sont nécessaires pour le bon déroulement du TD:

Composants	Remarques
WampServer	Version 5 ou supérieur installé
PHP	Version 5.2.5 ou supérieur installé, normalement, cette installation est faite lors de l'installation de WampServer.
MySQL	Version 5.1.11 ou supérieur installé, normalement, cette installation est faite lors de l'installation de WampServer
Prototype, Scriptaculous	Disponible dans les sources

3 Rappel des concepts

3.1 PHP :

Le langage PHP est un acronyme pour HyperText PreProcessor et est utilisé principalement en tant que langage de script côté serveur, ce qui veut dire que c'est le serveur qui va interpréter le code PHP et générer du code XHTML, HTML, CSS ou même du JavaScript qui pourra être interprété par un navigateur. Comme sa bibliothèque est très riche, on parle plutôt de plate-forme que d'un simple langage.

Il a été conçu pour les applications Web dynamiques et est majoritairement installé sur un serveur Apache mais peut être installé sur les autres principaux serveurs http comme par exemple IIS. Il permet de récupérer des informations issues d'une DB, d'un système de fichiers comme le

contenu des fichiers et de l'arborescence, ou plus simplement des données envoyées par le navigateur afin d'être interprétées ou stockées pour une utilisation ultérieure.

Le langage n'est pas typé et est donc souple et facile à apprendre. Ceci induit également des failles de sécurité qui apparaissent rapidement dans les applications. A noter que contrairement à RubyOnRails, le nom des fonctions, le passage des arguments et bien d'autres choses encore ne respectent pas toujours une logique uniforme, ce qui peut rallonger la période d'apprentissage. Ce langage nécessite également une bonne compréhension des mécanismes du Web.

Dans une utilisation Web, l'exécution du code PHP se déroule ainsi : lorsqu'un visiteur demande à consulter une page Web, son navigateur envoie une requête au serveur HTTP correspondant. Si la page est identifiée comme un script PHP (généralement grâce à l'extension .php), le serveur appelle l'interprète PHP qui va traiter et générer le code final de la page (constitué généralement d'HTML ou de XHTML, mais aussi souvent de CSS et de JS). Ce contenu est renvoyé au serveur HTTP, qui l'envoie finalement au client.

3.2 RubyOnRails[1] :

En juillet 2004, Ruby a été introduit dans le monde des applications Web par le biais de son framework très complet, Rails. RubyOnRails est donc le langage Ruby dans un framework Rails. Sa première version stable est sortie en décembre 2005. A noter que toutes ces versions sont gratuites, mais qu'elles présentent le défaut de ne pas toujours être compatibles entre elles.

Etant un langage jeune, il est parfaitement adapté au développement actuel alliant une utilisation de l'objet soignée, un code simple à utiliser, une flexibilité hors du commun et un binding automatique des bases de données, peu importe de quel type.

Il est composé en 3 environnements complets qui sont développement, test et production. Ces environnements sont fournis avec des outils différents selon le mode et permettent de cibler le travail et de ne pas influencer le fonctionnement de l'application dans un autre environnement. Cela est très pratique lorsqu'une application est en cours d'utilisation en mode « Production »

[1] Introduit ici comme complément du matériel présenté dans la partie théorique

dans une entreprise et que vous êtes développeurs. Vous aurez donc la possibilité de tout modifier et tester à votre guise sans avoir à vous souciez de vos collaborateurs.

RubyOnRails est basé en intégralité sur le principe MVC et a une philosophie bien strict : « Convention plutôt que configuration » et « Ne pas se répéter ». Ainsi, tout est basé sur les conventions et le fait de créer des choses disponibles pour tous afin de ne pas se répéter. Les conventions vont de la plus simple à d'autre qui sont très complexes. En respectant ces 2 principes, les éléments de l'application ne seront situés qu'à un seul endroit et il sera inutile de préciser des détails car le caractère par défaut remplira sa fonctionnalité dans la plupart des cas. Ainsi, RubyOnRails fournit des fonctions comme le *scaffolding* permettant de créer des applications à l'aide des conventions en 1 ligne de commande ! Finalement, RoR offre plusieurs méthodes afin de simplifier l'utilisation d'AJAX et permet d'intégrer très facilement des autres librairies et plugins diverses permettant d'étendre les possibilités de celui-ci.

3.3 AJAX :

AJAX est l'acronyme pour *Asynchronous JavaScript and XML* et permet le développement de pages dynamiques.

Il ne s'agit pas d'une technologie en elle-même, mais elle utilise conjointement un ensemble de technologies libres comme HTML pour la structure des informations, CSS pour la présentation, DOM et JavaScript pour afficher et interagir dynamiquement avec l'information présentée, XML ou JSON pour remplacer le format des données informatives (JSON) et visuelles (HTML).

L'élément principal utilisé est l'objet XMLHttpRequest. Il est utilisé pour échanger et manipuler les données de manière asynchrone avec le serveur Web.

A l'aide de toutes ces technologies, il est ainsi possible d'appeler le serveur Web de manière asynchrone et surtout invisible pour l'utilisateur et de rafraichir en fonction des résultats une portion de la page uniquement. AJAX permet donc de créer des applications Web laissant penser à l'utilisateur qu'il utilise une application en standalone. Pour améliorer le rendu visuel, il est possible d'utiliser des librairies comme Prototype ou Scriptaculous. A noter que ces librairies sont utilisées dans ce TP.

Les applications AJAX peuvent être utilisées au sein des navigateurs Web qui supportent les technologies décrites précédemment. Parmi eux, on trouve Mozilla Firefox, Internet Explorer, Safari, Opéra, Chrome, etc.

3.4 SilverLight[2] :

SilverLight est un plugin pour navigateur internet multiplateforme basée sur XAML qui améliore les présentations à base de contenu riches comme la 2D et 3D, les animations, les dessins

[2] Pas utilisé dans ce TP. C'est donc juste un rappelle théorique !

vectoriels, ou encore l'audio et la vidéo et toute ceci dans un moteur de rendu vectoriel. Elle fonctionne aussi bien sur Internet explorer, Mozilla Firefox, Safari, Chrome, etc.

De plus, la technologie du serveur d'hébergement n'a plus aucunes importances car SilverLight n'a aucune dépendance pour représenter des contenus enrichis sur internet. Il peut donc être exécuté sur un serveur possédant la technologie Apache pour PHP, TomCat pour Java ou encore IIS pour ASP.Net, la question ne se pose plus.

Au niveau technique, la raison est la suivante, SilverLight représente l'équivalent du CLR (Common Langage Runtime) de Microsoft pour les navigateurs Web. Comme vous l'aviez vu durant les cours et TPs précédents, le CLR crée un moyen de « communiquer » entre les différents langages, il en de même pour SilverLight dans son contexte internet.

La technologie est utilisable dans les navigateurs via l'utilisation d'un plugin dans les OS Windows et Mac OS X. A noter qu'une version pour la communauté Linux et les téléphones portables appelé Moon Light est aussi disponible.

En résumé, qu'est-ce que SilverLight ?

- Nom d'une technologie de présentation Web
- Fonctionne sur un large éventail de plateformes
- Fonctionne sur un large éventail de navigateurs
- Permet la création et l'exécution de contenus multimédias
- Permet d'intégrer de la vidéo Haute Définition
- Permet de réaliser des applications de type RIA
- N'a pas de dépendance avec la technologie du serveur Web utilisé

4 Travail à réaliser

Le travail vise à créer une application permettant de créer un blog géré par des administrateurs et donner la possibilité aux utilisateurs de commenter chaque post. Les administrateurs seront stockés dans un fichier XML alors que les posts et leurs commentaires seront stockés dans la base de données MySQL. Ce blog sera fait en PHP. Le tout utilisant AJAX pour la mise à jour des commentaires des personnes. Les librairies Prototype et Scriptaculous seront utilisés pour une présentation riche.

4.1 PHP

Nous allons créer un blog géré par un administrateur et donner la possibilité aux utilisateurs de commenter chaque post.

Dans un premier temps, nous allons créer la base de données et les tables qui seront utiles à notre application. Puis, nous allons créer une page permettant de gérer toutes ces données. La sécurité de notre site web au niveau de l'authentification et de la mise à jour des posts sera ensuite notre objectif.

Finalement, à l'aide de CSS, AJAX, Prototype et Scriptaculous, nous allons changer notre page bien triste en une page donnant une impression d'application standalone.

Avant tout, assurez-vous d'arrêter Skype ou tout autre programme qui pourrait utiliser le port de votre serveur web installé lors des précédents TPs (normalement, port 80). Puis, **lancez WampServer**.

4.1.1 Mise en place de la base de données

Pour créer la base de données et les tables, nous allons utiliser la ligne de commande. A noter qu'il peut être agréable d'utiliser PhpMyAdmin pour le faire. Cependant, dans notre cas, nous allons utiliser la ligne de commande afin que vous ayez l'opportunité de le voir une fois, la version PhpMyAdmin étant intuitive, il est peu nécessaire de la passer en revue.

Afin d'interagir avec MySQL, il est bon de connaître quelques lignes de commandes qui pourrait vous être utiles, non seulement pour le TP mais pour le futur :

Utilité	Commande
Se connecter au moniteur sur une machine hôte.	**mysql –h** *nomHote* **–u** *userName* **–p** *Aucun paramètre n'est requis.*
Demander un traitement depuis un fichier.	**mysql –h** *nomHote* **–u** *userName* **–p** *nomDB* **<** *nomFichier* *Les paramètres –h –u et –p ne sont pas requis.*
Demander un dump de la base de données. C'est-à-dire, demander à MySQL de créer les commandes SQL pour tout restituer, y compris les **INSERT**)	**mysqldump –h** *nomHote* **–u** *userName* **–p** *nomDB* **>** *nomFichier* *Les paramètres –h –u et –p ne sont pas requis.*
Utilisation/Changement de base de données.	**USE** *nomDB*
Lister les bases de données.	**SHOW databases**
Lister les tables contenues dans la base de données actuellement sélectionnée.	**SHOW TABLES**
Décrire la structure d'une table.	**DESCRIBE** *nomTable*

Ouvrez une ligne de commande et dirigez-vous vers le répertoire MySQL de Wamp contenant les exécutables nécessaires à interagir avec les bases de données. Normalement, le répertoire devrait être

```
C:\wamp\bin\mysql\mysql5.5.16\bin
```

Puis, entrez dans le moniteur MySQL grâce à la commande ci-dessous. Cette commande utilise le paramètre –u pour permettre de spécifier l'utilisateur et –p pour spécifier qu'il y a un mot de passe. Notez qu'un autre paramètre, le paramètre –h permet de spécifier le nom d'une machine hôte avec laquelle vous désireriez interagir.

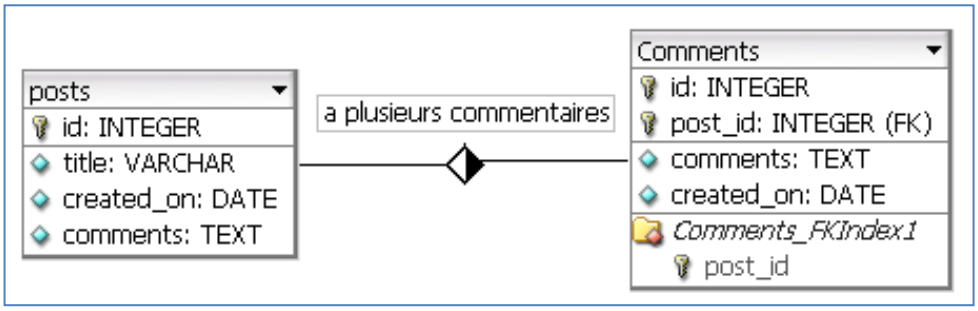
```
mysql –u root -p
```

Après avoir tapez la commande, le moniteur vous demande votre mot de passe, tapez sur « Enter », vous êtes dès lors entrez dans le moniteur MySQL.

Il est maintenant temps de créer la base de données nécessaires à notre site PHP et de créer les tables dans notre base de données. Pour ce faire, tapez la commande permettant de créer la BD nommée « tp_php » et entrez dans la gestion de celle-ci comme le montre l'exemple ci-dessous.

```
mysql> CREATE DATABASE tp_php;
Query OK, 1 row affected (0.00 sec)

mysql> USE tp_php;
Database changed
mysql>
```

Notre application a besoin de 2 tables, une contenant les posts et une autre contenant les commentaires. Notre structure de table est la suivante :



On peut remarquer que la table « *Comments* » possède une clé étrangère permettant de définir à quel post il est relié. Afin de créer ces tables, il vous suffit d'importer le fichier « *phpTables.sql* » disponible dans les sources du TP. Pour ce faire, tapez la commande ci-dessous :

```
source emplacementDuFichier\phpTables.sql ;
```

La base de données est maintenant prête à accueillir vos données.

4.1.2 Création d'une page

Il est maintenant temps de créer notre page qui nous permettra de gérer notre blog. Comme nous allons par la suite changer l'interface dans un concept RIA, il n'est donc pas très utile de pousser ce sujet. Aller dans le répertoire de votre serveur Web, normalement *C:\wamp\www*.

En effet, lorsque votre serveur Web démarre, il va dans le répertoire www et tente de trouver un fichier portant le nom *index.html*, *index.htm* ou *index.php*. Comme nous allons utiliser le langage PHP, il s'agira d'*index.php*. A noter qu'il est possible de modifier ce comportement en allant modifier la configuration dans le répertoire *conf* du serveur web à la ligne

```
DirectoryIndex index.php index.php3 index.html index.htm
```

Copiez donc les fichiers *index.php* qui sera la page principal de notre projet, *viewPost.php* qui représente la vue d'un post et *viewComment.php* qui représente la vue d'un commentaire. Ces fichiers sont disponibles dans les sources du TP sous *php/step0*. Ces pages ne respectent pas les « *Best Practices* » de codage, il s'agit juste d'un exemple. Les « *Best Practices* » seront appliqués dans les chapitres suivants.

Tester la page dans votre navigateur à l'adresse *http://localhost* ou à l'adresse *http://127.0.0.1*.

Cette page se veut très simple. Si vous regardez le code, l'*index* contient uniquement le titre, *viewPost* contient un tableau de 2 lignes de tableau et les lignes provenant de *viewComment* qui contient quant à lui 1 ligne de tableau. Les pages ne sont pour l'instant pas dynamiques mais elles permettent de montrer une ligne PHP très pratique et très utilisée, il s'agit de la commande

```
<?php require ("nomDeLaPage.php"); ?>
```

Vous noterez donc que le code PHP peut être directement insérer dans le code HTML à l'aide de la balise ouvrante *< ?php* et de la balise fermante *?>*. La fonction *require* est utilisée pour importer un fichier dans un autre. Ainsi, l'*index* importe le code de *viewPost* qui importe lui-même le code *viewComment*. Il est maintenant temps d'appeler notre code SQL.

4.1.3 SQL avec PHP

Les appels SQL depuis PHP nécessitent préalablement une connexion vers la base de données. Il est conseillé de créer cette connexion dans un fichier séparé. Copiez les fichiers « *conf.inc.php* » et « *error.php* » contenu dans les sources du TP vers la racine du site (*C:\wamp\www*).

```php
<?php
    /* Configuration afin d'accéder à la base de données */
    $server = 'localhost'; // nom du serveur mysql
    $user = 'root'; // login mysql
    $pass = 'TODO//A COMPLETER'; // mot de passe mysql
    $base = 'tp_php'; // sélection de la base de donnée
    if (!($db = mysql_connect($server, $user, $pass))) {
        header('Location: error.php');
        exit();
    }
    mysql_select_db($base, $db); // choisir une base de données
?>
```

Ce fichier « *conf.inc.php* » permet donc une connexion vers la base de données « *tp_php* » sur le serveur « *localhost* ». En cas de problème lors de la connexion, nous arrêtons le traitement et demandons l'affichage de la page d'erreur. Vous pouvez remarquer que toutes les variables sont non-typées et qu'elles sont précédées par le caractère *$*. Notez que nous utilisons l'utilisateur « *root* » pour nous connecter, cela est dangereux car cet utilisateur a tous les droits et un hacker pourrait l'utiliser afin d'exécuter des commandes hostiles. Dans le cas d'un projet importants, il serait donc important de créer un utilisateur supplémentaire qui aurait des droits limités sur la base de données.

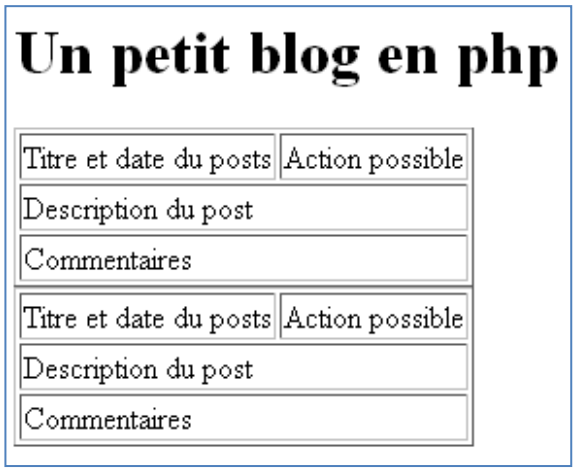
Maintenant que la connexion avec la base de données est établie, nous pouvons appeler les tables afin de les utiliser.

Premièrement, dans la page « *index.php* », nous allons demander la liste de tous les posts et faire une boucle sur chacun d'eux afin de les afficher en utilisant notre page « *viewPost.php* » comme ceci :

```php
<?php
    require_once ("conf.inc.php"); // Connection à la base de données
    $selectPosts = 'SELECT * from posts order by created_on ASC'; // Selection de tous les posts
    $requestPosts = mysql_query($selectPosts); // Execution de la requête
    while ($post = mysql_fetch_array($requestPosts)) {
        // $post contient à chaque fois un nouveau post
        require("viewPost.php"); // Demander la page viewPost qui affiche les valeurs contenus dans $post
    }
?>
```

Ce nouveau fichier d'*index* est disponible dans les sources sous le dossier *php/step1*.

Tester à nouveau votre page sur *localhost*.



Vous pouvez constater que le nombre de tableaux a doublé, notre table *posts* contenant 2 entrées, nous avons donc correctement sélectionné chacun d'eux.

Il est maintenant nécessaire de modifier notre fichier *viewPost*. Il s'agit de récupérer les valeurs contenus dans la variable *$post* provenant de l'*index*, lancer une nouvelle requête permettant de sélectionner tous les commentaires associés au post courant et d'appeler la page *viewComment* afin de les afficher. Copiez donc le fichier « *viewPost.php* » contenus dans les sources sous *php/step1*.

```
<table border="1">
    <tr>
        <td><?php echo $post['title']; ?> <strong><?php echo $post['created_on']; ?></strong></td>
    <td>Actions possible</td>
    </tr>
    <tr>
        <td colspan="2"><?php echo $post['comments']; ?></td>
    </tr>
    <?php
        require_once ("conf.inc.php"); // Connection à la base de données
        $selectComs = 'SELECT * from comments WHERE post_id = '.$post['id'].' order by created_on ASC'; // Selection de tous les commentaires
        $requestComs = mysql_query($selectComs); // Execution de la requête
        while ($comment = mysql_fetch_array($requestComs)) {
            // $comment contient à chaque fois un nouveau commentaire
            require("viewComment.php"); // Demander la page viewComments qui affiche les valeurs contenus dans $comment
        }
    ?>
</table>
```

Remarquez que la variable *$post* est accessible dans le fichier *viewPost*. De plus, il est important de remarquer que les requêtes SQL sont contenues dans un tableau hash.

Tester à nouveau votre page sur *localhost*.

Notre table *Comments* contenant 2 commentaires pour le 1er post et 1 commentaire pour le 2ème post, nous pouvons constater que tout fonctionne correctement. Il ne reste plus qu'à afficher les commentaires. Copiez donc le fichier « *viewComment.php* » contenus dans les sources sous *php/step1*.

```
<tr>
    <td>Date du commentaire : <?php echo $comment['created_on']; ?></td>
    <td><?php echo $comment['comments']; ?></td>
</tr>
```

Le résultat est le suivant et est donc cohérent.

Dans la page « viewPost.php », nous allons ajouter un bouton permettant de supprimer le post en envoyant l'id avec la méthode *get*. Le *get* signifie que la variable est passée dans l'adresse du lien directement. Un autre moyen, nommé *post*, passe les informations de manière transparente au niveau de l'adresse. Cette technique sera revue lorsque nous ajouterons une vue permettant d'ajouter un post. Rajouter donc le code suivant dans la colonne.

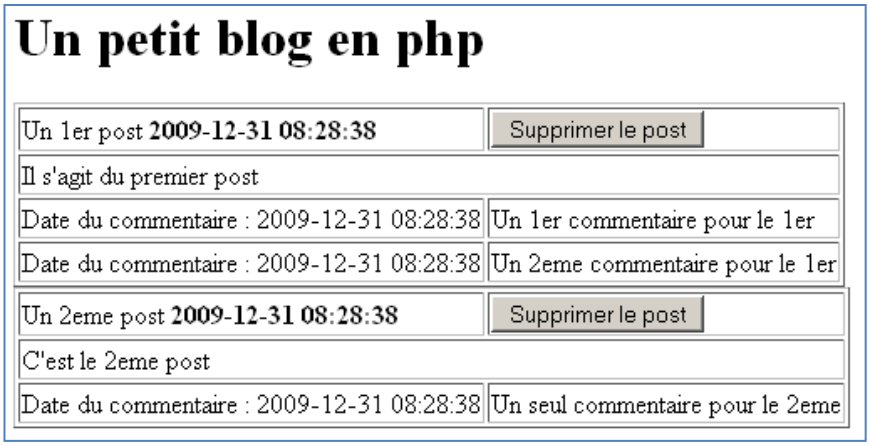
```
<button
        onClick="window.location.href='removePost.php?id=<?php echo $post['id'];?>'">Supprimer le post
</button>
```

Notre bouton fait donc un lien avec la page *removePost.php* en stipulant l'identifiant du post.

Il ne reste maintenant qu'à supprimer le post ayant l'id en paramètre et réafficher la page. Copiez donc le fichier « *removePost.php* » contenus dans les sources sous *php/step1*.

```php
<?php
    $id = $_GET['id']; // l'identifiant en paramètre dans l'adresse

    require_once ("conf.inc.php"); // Connection à la base de données
    $sql = "DELETE FROM posts WHERE id='".$id."';"; // Suppression du post
    $result = mysql_query($sql); // Execution de la requête
    if($result){
        header('Location: index.php'); // Ok, donc recharge l'index
    }else{
        header('Location: error.php'); // Pas ok, afficher l'erreur
    }
    exit();
?>
```

Nous utilisons le domaine *$_GET* pour récupérer notre paramètre envoyé avec la méthode *get* justement. Tester à nouveau à l'adresse *localhost*…Ça fonctionne, il ne manque plus qu'à permettre d'ajouter un post et des commentaires, et on aura terminé.



Afin d'ajouter un post, il nous suffit de créer un petit formulaire contenant uniquement un champ pour rentrer le titre, une zone de texte permettant de décrire son post et un bouton permettant de valider le formulaire. Copiez donc le fichier « *addPostView.php* » contenus dans les sources sous *php/step1*. Demander l'inclusion de cette page en haut de la page d'*index* avec la méthode *require* vu précédemment.

```
<form method="post" action="addPostModel.php">
<table border="1">
    <tr>
        <td colspan="2" align="center"><h2>Ajout d'un nouveau post</h2></td>
    </tr>
    <tr>
        <td>Titre du post</td>
        <td><input type="text" name="titre"></td>
    </tr>
    <tr>
        <td>Description du post</td>
        <td><textarea name="description"></textarea></td>
    </tr>
    <tr>
        <td colspan="2" align="center"><input type="submit" value="Enregistrer"></td>
    </tr>
</table>
</form>
```

Comme nous pouvons le voir ci-dessus, nous utilisons les balises *form* qui permettent de stipuler que tous ce qu'il contient fait partie d'un formulaire, le bouton *submit* permet quant à lui d'envoyer ce formulaire vers le chemin de destination, ici « *addPostModel.php* ». *addPostModel.php* doit donc se charger de récupérer les valeurs du formulaire, d'ajouter le post dans la base de données et recharger la page. Ajoutez donc le fichier « *addPostModel.php* » contenus dans les sources sous *php/step1*.

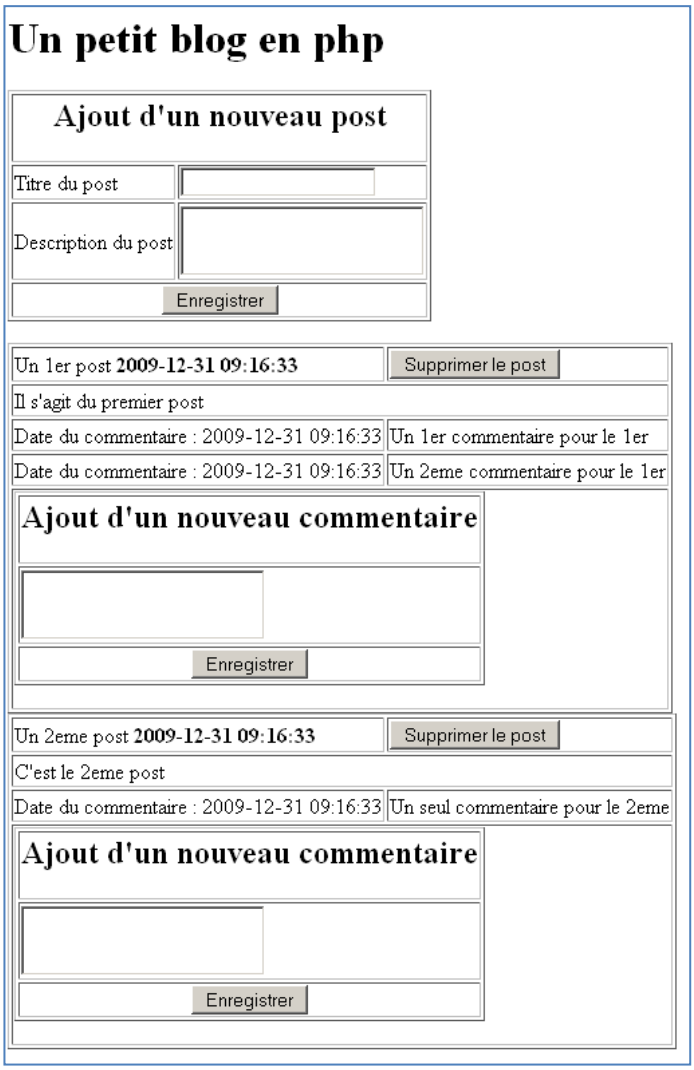
```
<?php
    $titre = $_POST['titre'];
    $description = $_POST['description'];

    require_once ("conf.inc.php"); // Connection à la base de données
    $sql = "INSERT INTO posts (title ,comments) VALUES ('".$titre."', '".$description."');";
    $result = mysql_query($sql);
    if($result){
        header('Location: index.php'); // Ok, donc recharge l'index
    }else{
        header('Location: error.php'); // Pas ok, afficher l'erreur
    }
    exit();
?>
```

Contrairement à la suppression qui était avec la méthode *get*, nous utilisons cette fois-ci le domaine *$_POST* pour récupérer notre paramètre envoyé avec la méthode *post* justement.

Le résultat doit maintenant être le suivant :

La partie concernant l'ajout de commentaire est similaire à celle d'ajouter un post, nous n'allons donc pas la détailler. Il est cependant bon de remarquer que l'*id* du post lors de l'ajout d'un commentaire est passé à l'aide d'un champ caché. Pour arriver à l'état final de cette partie purement axé sur le web, remplacez les fichiers contenus dans votre racine (*C:\\wamp\www*) par les fichiers contenus dans les sources sous *php/step2*.

Le résultat final est le suivant :



Il ne reste qu'à demander aux administrateurs de se connecter avant de pouvoir ajouter un nouveau post.

Injection SQL

Attention aux injections SQL dans les formulaires, les injections vont permettre à des hackers d'afficher vos données ou d'insérer/modifier/supprimer des données de votre base de données. Le but d'une injection SQL est d'ajouter une requête SQL dans un champ d'un formulaire. Voici un exemple :

```
DELETE from posts WHERE id=‘ $id ’ ;
```

Cette requête semble apparemment anodine mais en réalité, elle est très dangereuse. Elle devrait uniquement supprimer de la table « *posts* » une donnée selon la valeur « *$id* ».

Cependant, si une personne mal attentionnée enverrait comme paramètre la ligne « ‘or ‘1 », la requête deviendrait :

```
DELETE from posts WHERE id='' or '1' ;
```

Cette requête supprime tous les éléments de la table « *posts* ». Dans ce cas, ce n'est pas encore très grave, mais il pourrait également, s'approprier les données comme les cartes de crédit, supprimer la base de données, etc. En bref, avoir un contrôle total sur votre base de données.

Pour se protéger, il est possible d'utiliser par exemple, la fonction *mysql_real_escape* qui enlève le « *magic quotes* » ou encore pour le cas de chiffre, vérifier que la valeur est réellement un chiffre. Il existe de nombreuses manières, mais il est à savoir qu'aucune n'est sans faille…

4.1.4 Session

Dans l'*index*, nous faisons l'inclusion de la page « *addPostView.php* » qui donne la possibilité d'ajouter des posts. Cependant, nous désirons que seuls les administrateurs puissent en ajouter. Pour ce faire, nous allons créer une condition (*if*). Dans celle-ci, nous allons tester si l'administrateur est logué. S'il l'est, nous affichons la partie d'ajout d'un nouveau post, s'il ne l'est pas, nous affichons la partie permettant de se loguer.

Tout d'abord, la question est « Comment se rappeler du log de notre administrateur ? ». La réponse est avec les variables de session.

Les sessions en PHP permettent de préserver certaines données à travers la visite du site. Pour chaque visiteur du site, un identifiant unique, appelé Session ID, est assigné. Ce Session ID peut être stocké dans un cookie du côté client ou passé à travers l'URL. Une session PHP permet donc l'enregistrement d'un nombre arbitraire de données qu'il faut conserver tout au long de la visite du site. Afin de créer cette session, le moteur PHP assigne aux visiteurs le Session ID, l'enregistre dans le dossier spécifié dans le fichier *php.ini* et passent cette variables de session à chaque page. A noter que si l'utilisateur a la possibilité de stocker les cookies, le moteur PHP passe la session à l'utilisateur, la raison étant de ne pas surcharger le serveur inutilement.

Maintenant que vous avez compris ce qu'est une session, nous pouvons donc modifier notre site. Afin de savoir si une session existe, il suffit d'utiliser la fonction ***session_start*** qui va vérifier si la session a été démarrée. Si elle existe, il la récupère, sinon il en crée une nouvelle. Il est conseillé d'exécuter cette fonction au début de chaque page de l'application. A noter que si l'on veut supprimer la variable de session, dans le cas d'un « délogue » de l'administrateur par exemple, il suffit d'exécuter la fonction ***session_destroy***. Les variables de session sous forme de hash sont disponibles en tapant le nom ***$_SESSION***

Nous allons donc tester l'inclusion ou non de la page permettant d'ajouter un post. Copiez le fichier « *index.php* », « *login.php* », « *unlogin.php* » et « *checkLogin.php* » contenus dans le répertoire *php/step3* des sources vers la racine (*C:\\wamp\www*).

Comme nous avons déjà vu la connexion vers la base de données, les noms d'utilisateurs et les mots de passe ont été enregistrés dans le fichier XML « admins.xml » afin que nous les parsions. Il faut également copier ce fichier à la racine.

Vous pouvez maintenant remarquer dans la page « *index.php* » la 1ère ligne « ***session_start() ;*** » qui démarre la session et le code suivant qui teste si l'administrateur est connecté.

```php
<?php
    // Est-ce que l'administrateur est connecté ?
    if (!isset($_SESSION['connect'])) {
        // NON, alors affiche la partie de login
        require("login.php");
    }else{
        // OUI, alors affiche la partie de délogin et d'ajout de post
        require("unlogin.php");
        require("addPostView.php");
    }
```

Nous avons pu observer auparavant les balises « *form* » pour les formulaires, la règle s'applique également pour le login qui est un formulaire à part entière. Il n'est donc pas nécessaire de le détailler. Par contre, c'est dans la partie de contrôle du login « *checkLogin.php* » que nous allons consacrer notre attention.

```php
<?php
    // La page a été appelé depuis le bouton de login
    // => donc il faut essayer de se connecter
    if(isset($_POST['login']) && isset($_POST['pass'])){
        //Il faut tester si les valeurs sont corrects
        $login = $_POST['login'];
        $pass = $_POST['pass'];

        // Utilisation de DOM pour parser le XML
        $objDOM = new DOMDocument();
        $objDOM->load("admins.xml"); // Chargement du fichier
        $admins = $objDOM->getElementsByTagName("admin");
        foreach( $admins as $admin ){
            $crtLogin = $admin->getElementsByTagName("username");
            $crtLogin = $crtLogin->item(0)->nodeValue;
            $crtPass = $admin->getElementsByTagName("password");
            $crtPass = $crtPass->item(0)->nodeValue;
            if($crtLogin == $login && $crtPass == $pass){
                session_start();
                $_SESSION['connect'] = "il est connecté";
                break;
            }
        }
    }else{
        // La page a été appelé depuis le bouton de déconnexion
        // => donc il faut déconnecter
        session_start();
        unset($_SESSION['connect']);
    }
    header('Location: index.php'); // Revenir dans tous les cas dans l'index
?>
```

Vous pouvez constater que le code est divisé en 2 parties. La 1ère est lorsque la page a été appelée depuis le bouton login, et l'autre depuis le bouton de déconnexion.

Comme discuté précédemment, afin de se déconnecter, il aurait été possible d'appeler la fonction ***session_destroy***, mais comme dans notre cas, il n'y a qu'une variable assigné nommé « *connect* », nous avons préféré vous montrer comment désallouer la variable avec la fonction ***unset***, ce qui est parfaitement suffisamment pour ce petit exemple, la destruction de la session

étant plus utile lorsque les variables sont nombreuses comme dans le cas d'un panier d'achat par exemple.

Vous pouvez constater également que le login et le mot de passe, repris depuis le hash *$_POST* sont comparés aux valeurs contenus dans le fichier XML. Ces valeurs sont lues à l'aide du parseur DOM qui est par défaut dans PHP. Lorsque les données ont été vérifiés, la variables de session « *connect* » est assigné, notre administrateur est connecté et la page « *index.php* » est rechargé.

Remarque : il est conseillé de vérifier que l'administrateur est connecté pour toutes les pages le nécessitant. Dans notre exemple, il s'agit des pages « *removePost.php* », « *addPostView.php* » et « *addPostModel.php* ». Il faudrait donc commencer chacune de ces pages avec les lignes :

```php
<?php
    session_start();
    if (!isset($_SESSION['connect'])) {
        header('Location: index.php'); // Ok, donc recharge l'index
    }
?>
```

Tester à nouveau à l'adresse *localhost* .

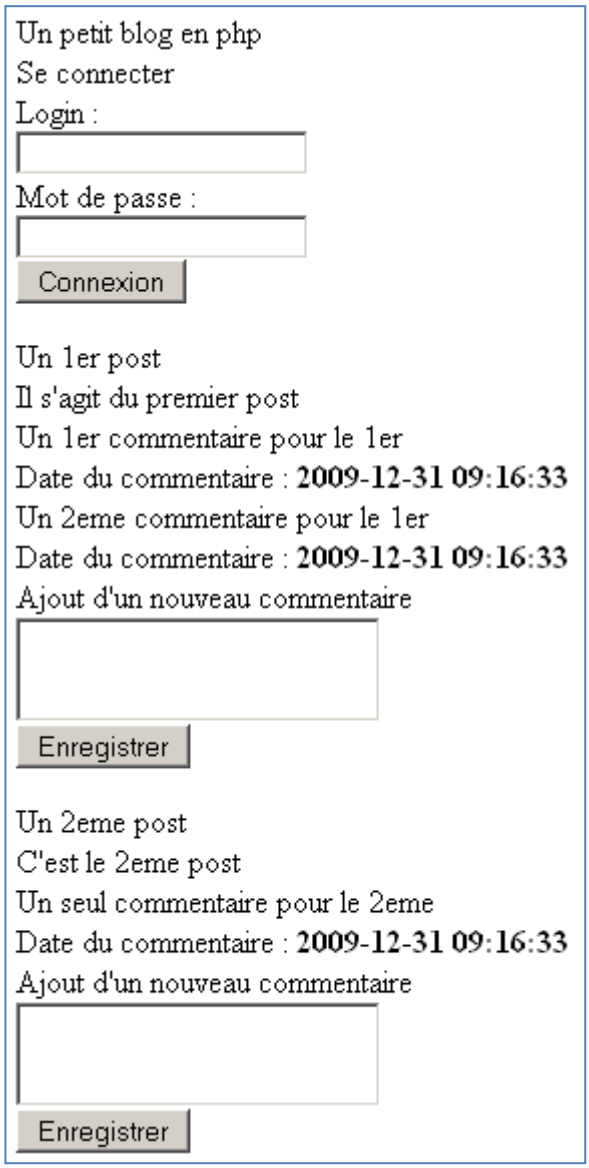
4.1.5 CSS

Nous avons terminé la dernière partie avec une page web très désagréable à utiliser. De plus, en utilisant les tableaux très peu flexibles, il n'est pas réellement possible de créer une interface esthétiquement attirante. L'utilisation des « DIV » est maintenant conseillée. Elles définissent une division/section dans un document HTML.

Le meilleur couplage à faire est d'utiliser des fichiers CSS. Les CSS (Cascading Style Sheet) contiennent des données permettant un mécanisme simple afin d'ajouter du style comme les polices, les couleurs, les emplacements, etc. dans les documents web en cascade. Les styles sont dits en « Cascade » car chaque élément enfant va suivre les styles de ces parents, puis appliquer les styles qui lui sont propres. Cela est particulièrement apprécié et cela peut s'avérer indispensable dans le cas d'une mise en page d'élément de type DIV comme c'est le cas dans cet exercice.

Copiez tous les documents contenus dans le répertoire *php/step4* des sources vers la racine *C:\\wamp\www*. Ces fichiers sont exactement les mêmes que ce que nous avions fait précédemment, à l'exception du fait que chaque ligne de tableau a été remplacé par des éléments de type DIV.

Le résultat est le suivant, vous pouvez constater que les DIV qui ne sont pas paramétrés à l'aide de CSS par exemple s'empilent ligne par ligne:



Par exemple, si nous regardons de plus près le fichier « *viewPost.php* », nous pouvons constater qu'en plus de s'être simplifié, toutes les données sont contenues dans des DIVS.

```php
<div class="postContainer">
    <div class="postInstance">
        <div class="title"><?php echo $post['title']; ?></div>
    <?php
    // Est-ce que l'administrateur est connecté ?
    if (isset($_SESSION['connect'])) {
        // Oui, alors affiche la partie de login
    ?>
        <div class="remove">
            <button class="removeB" onClick="window.location.href='removePost.php?id=<?php echo $post['id'];?>'">Supprimer le post</button>
        </div>
    <?php
    }
    ?>
        <div class="date"><?php echo $post['created_on']; ?></div>
    </div>
    <div class="postComments"><?php echo $post['comments']; ?></div>
    <?php
        require_once ("conf.inc.php"); // Connection à la base de données
        $selectComs = 'SELECT * from comments WHERE post_id = '.$post['id'].' order by created_on ASC'; // Selection de tous les commentaires
        $requestComs = mysql_query($selectComs); // Execution de la requête
        $i = 0;
        while ($comment = mysql_fetch_array($requestComs)) {
            if($i == 0){
                echo "<div class=\"titleComments\">Commentaires</div>";
            }
            // $comment contient à chaque fois un nouveau commentaire
            require("viewComment.php"); // Demander la page viewComments qui affiche les valeurs contenus dans $comment
            $i++;
        }
        $post_id = $post['id']; // $post_id est utilisé dans la vue permettant d'ajouter un commentaire
        require("addCommentView.php");
    ?>
</div>
```

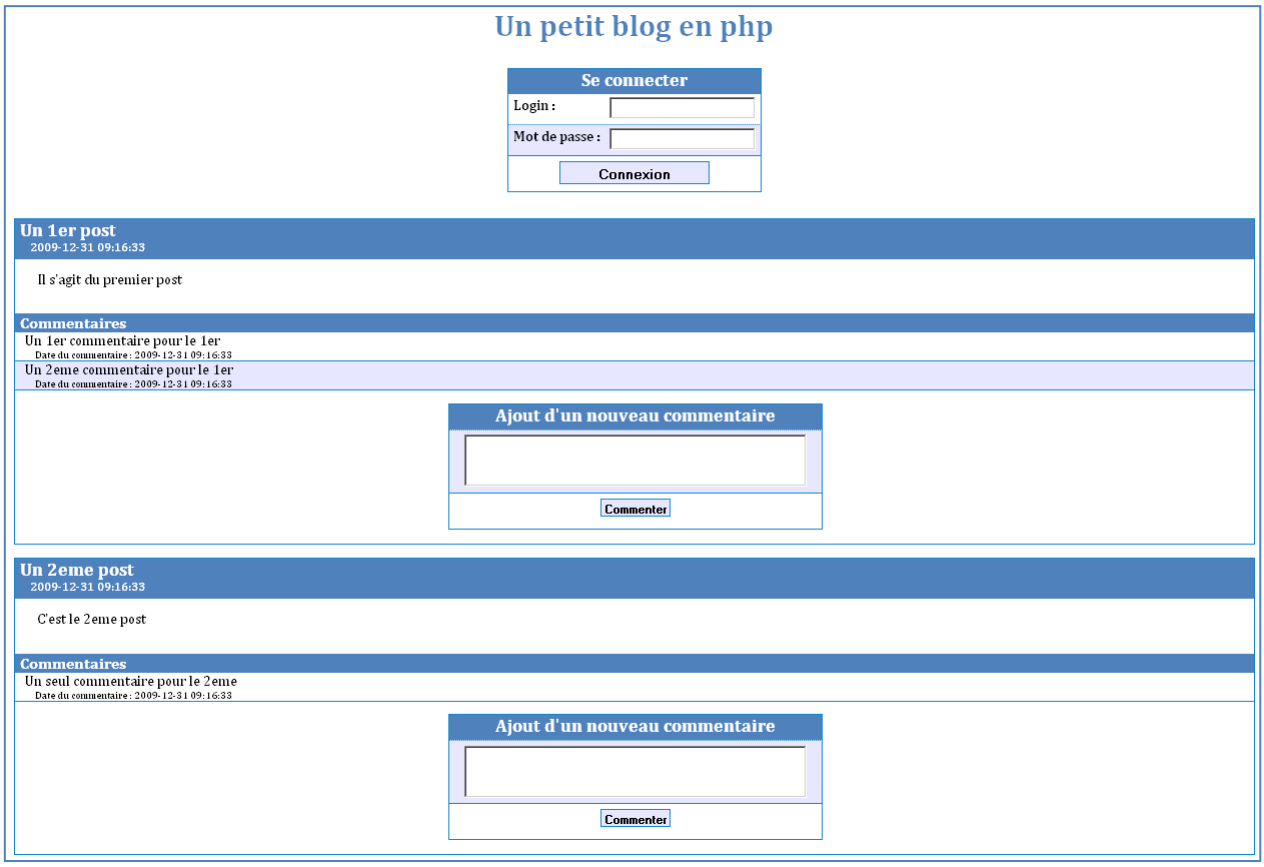
Vous pouvez identifier les DIVS en regardant leur identifiant. L'identifiant est utilisé dans la feuille de style CSS afin de les paramétrer pour les placer et leur donner les couleurs, les polices, etc. que nous désirons. Vous pouvez par exemple remarquer que dans le fichier « style.css », nous pouvons retrouver tous les DIVS (*postContainer*, *postInstance*, *title*, *remove*, *date*, *postComments* et *titleComments*) de cette page.

```
div.postContainer{
    width: 98%;
    border: 1px solid #1187CB;
    overflow: hidden;
    margin: 1%;
}
div.postContainer .postInstance {
    width: 100%;
    height: 40px;
    overflow: hidden;
    background-color: #4F81BD;
    color: white;
}
div.postInstance .title {
    width: 85%;
    font-size: 18px;
    font-weight: bold;
    text-indent: 5px;
    float: left;
}
div.postInstance .date {
    font-size: 12px;
    font-weight: italic;
    text-indent: 15px;
    width: 85%;
}
div.postInstance .remove {
    margin-top: 2px;
    margin-bottom: 2px;
}
div.postContainer .postComments {
    margin: 1%;
    min-height: 30px;
    text-indent: 10px;
}
div.postContainer .titleComments {
    width: 100%;
    font-size: 16px;
    font-weight: bold;
    text-indent: 5px;
    background-color: #4F81BD;
    color: white;
}
```

Afin de charger notre feuille de style, il suffit de rajouter dans l'entête de « *index.php* » la ligne suivante :

```
<link href="style.css" rel="stylesheet" type="text/css">
```

Recharger la page, elle devrait ressembler à la page ci-dessous :



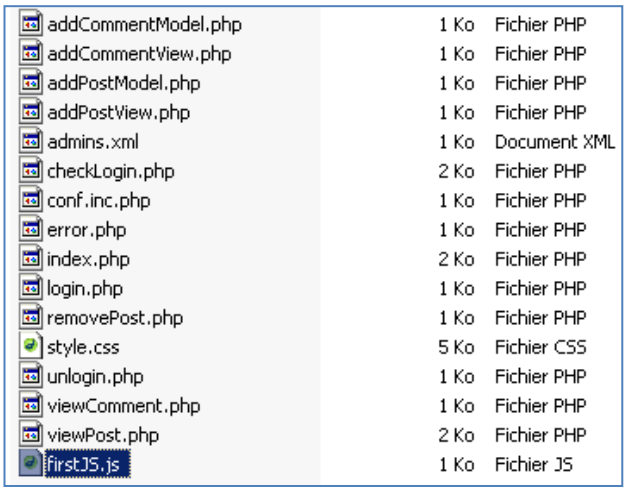
C'est bluffant, non ? En plus d'être rapide à modifier, cela permet de vraiment se concentrer entre la couche métier, la gestion des données et l'interface. De plus, vous pouvez par exemple créer des thèmes pour votre site web. En effet, en créant plusieurs fichiers CSS qui ont chacun un style bien particulier, vous pourriez par exemple, charger un fichier en fonction de la saison, le style étant bien différent entre l'hiver et l'été par exemple.

4.1.6 Validation

Dans le monde des applications Web, il est important de réussir à optimiser les accès vers les serveurs. Ainsi, il est important de n'appeler le serveur que lorsque les valeurs ont déjà été vérifiées par exemple, afin de ne pas devoir recharger la même page avec un message d'erreur.

Dans cette optique, nous allons dans un premier temps, à l'aide de JavaScript, vérifier que le commentaire d'un utilisateur ne soit pas vide du côté client. Puis, lorsque nous aurons réalisé cette vérification, nous lancerons le comportement par défaut, c'est-à-dire le submit du formulaire.

Il est fortement conseillé de créer un fichier JavaScript séparé du reste de la page plutôt que de l'intégrer directement dans la tête. Créez donc un nouveau fichier nommé « *firstJS* » ayant comme extension « *.js* ».



Ouvrez ce fichier et coder notre 1er fonction nommé « firstTest » qui s'occupera uniquement d'afficher le message « *test* » à l'utilisateur.

```
function firstTest(){
    alert("test");
}
```

Comme précédemment pour le fichier CSS, afin de rendre le fichier JS disponible pour notre site web, il est nécessaire de l'importer dans l'entête du fichier « *index.php* » comme ceci :

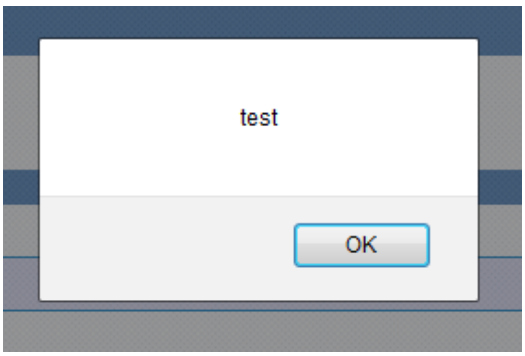
```
<head>
    <title>Le 1er blog en PHP</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <link href="style.css" rel="stylesheet" type="text/css">
    <script src="firstJS.js"></script>
    </head>
<body>
```

Nous allons appeler cette fonction depuis le bouton « *Commenter* » de la page « *addComment*View». Pour ce faire, modifier le type du bouton qui est « submit » vers la valeur « *button* ». En effet, le type « *submit* » signifie que le bouton va automatiquement envoyer le formulaire alors que la type « *button* » va réagir en fonction des événements « *onClick* », « *onBlur* », etc.

Comme le bouton est maintenant de type « *button* », éditer l'événement « *onClick* » et appeler la fonction « *firstTest* » comme ceci :

```
<input type="button" onClick="javascript: firstTest();" class="submitComment" value="Commenter">
```

Recharger votre site web sur *localhost*. Le comportement du bouton devrait être d'afficher le message suivant :



Maintenant que nous sommes sûrs de pouvoir appeler JavaScript, nous allons valider notre formulaire et l'envoyer si les valeurs sont correctes ou afficher un message d'erreur si les valeurs ne le sont pas.

Grâce à JavaScript, il est possible d'interagir directement avec les éléments de la page. Pour ce faire, il nous suffit juste de l'appeler en utilisant son identifiant. Nous allons donc rajouter un élément de type « *DIV* » nommé « *erreur* » qui se chargera de contenir les éventuels messages d'erreurs, de rajouter un identifiant à la zone de texte du commentaire et finalement à notre formulaire afin que nous puissions l'envoyer lorsque les valeurs seront correctes.

En parlant d'identifiant, on parle surtout d'unicité. Ainsi, ces différents éléments que nous devons atteindre avec JavaScript doivent être uniques dans la page. C'est pourquoi, nous allons utiliser l'identifiant provenant du post afin d'assurer l'unicité de ceux-ci. De plus, nous allons passer cet identifiant à la seconde méthode « *secondTest* » que nous allons créer. Le fichier « *addCommentView* » devrait donc ressembler à ceci :

```
<form id="form_<?php echo $post_id;?>" method="post" action="addCommentModel.php">
    <div id="erreur_<?php echo $post_id;?>"></div>
    <div class="addCommentsContainer">
        <div class="title">Ajout d'un nouveau commentaire</div>
        <div class="descriptionContent">
            <textarea id="comment_<?php echo $post_id;?>" name="comments" cols="40"></textarea>
            <input name="post_id" type="hidden" value="<?php echo $post_id;?>">
        </div>
        <div class="buttonBox">
            <input type="button" onClick="javascript: secondTest('<?php echo $post_id;?>');" class="submitComment" value="Commenter">
        </div>
    </div>
</form>
```

Le fichier « addCommentView » est disponible dans les sources sous « *php/step5* »

Modifions le fichier « *firstJS.js* » et rajoutons la méthode « *secondTest* » recevant un paramètre qui est l'identifiant du post comme montré ci-dessous. Comme vous pouvez le voir, le JavaScript n'est pas typé non plus.

```
function secondTest(postId){
    alert(postId);
}
```

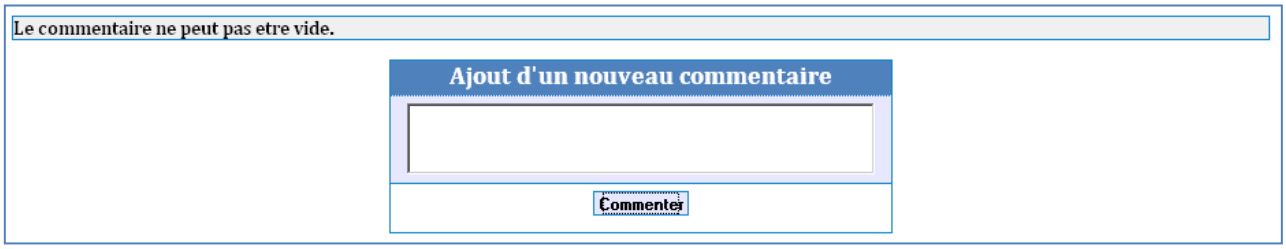
Testez votre application, le clique devrait faire apparaître l'identifiant du post.

Il suffit maintenant de contrôler la valeur du champ « *comment_POST_ID* ». Voici le code permettant de contrôler la valeur et de réagir en fonction de celle-ci, ce code est disponible ci-dessous :

```
function secondTest(postId){
        // Est-ce que la valeur est vide ?
        if(document.getElementById('comment_' + postId).value == ""){
                //Oui, alors affiche un message d'erreur
                var message = "<p><div class='notice'>Le commentaire ne peut pas etre vide.</div></p>";
                document.getElementById('erreur_' + postId).innerHTML = message;
        }else{
                //Non, alors submit le formulaire
                document.getElementById('form_' + postId).submit();
        }
}
```

Vous pouvez remarquer que nous recherchons les éléments par leur identifiant en appelant la méthode « *document.getElementById* ».

Testez votre application. Lorsque vous entrez un commentaire vide, vous devriez obtenir le comportement ci-dessous et un ajout du commentaire dans le cas contraire.



4.1.7 XMLHTTPRequest

Imaginez que la page contienne de nombreuses images et des milliers de posts contenant eux-mêmes des milliers de commentaires. Il serait inutile de rafraîchir toutes la page afin de rajouter le commentaire. Le mieux serait donc de ne rajouter dans le page que le commentaire qui vient d'être ajouté.

Pour ce faire, nous allons utiliser les *XMLHTTPRequest*, il s'agit de la face cachée d'*AJAX* qui sera expliqué dans le prochain chapitre.

Les *XMLHTTPRequest* sont des spécifications qui définissent une API qui fournit des fonctionnalités de script du côté client afin de transférer des données entre le client et le serveur.

Que pouvons-nous faire pour améliorer notre site ? Nous allons utiliser les *XMLHTTPRequest* afin d'appeler l'ajout du commentaire. L'ajout du commentaire se fera par le biais de la page « *addCommentModel.php* » disponible dans les sources sous « *php/step6* ».

```php
<?php
    $comments = $_POST['comments'];
    $post_id = $_POST['post_id'];

    require_once ("conf.inc.php"); // Connection à la base de données
    $sql = "INSERT INTO comments (post_id ,comments) VALUES ('".$post_id."', '".$comments."');";
    $result = mysql_query($sql);
    if($result){
        $sql = "SELECT * from comments WHERE id='".mysql_insert_id()."';";
        $result = mysql_query($sql);
        $comment = mysql_fetch_array($result);
        require("viewComment.php"); // Ok, donc recharge le post commenté
    }else{
        echo "0"; // Pas ok, renvoyer 0
    }
    exit();
?>
```

Vous remarquerez que dans le cas d'un échec, la page renvoie 0 et dans le cas d'un succès, on sélectionne l'élément que le dernier élément a ajouté, on le sauvegarde dans la variable « *$comment* » afin qu'elle soit affichée dans la page « *viewComment.php* » dans un nouveau DIV.

Maintenant que l'on a décidé de la façon avec laquelle nous allons répondre de la demande, il faut appeler cette page. Tout d'abord, aller dans la page « *addCommentView.php* » et changer le lien du bouton « *Commenter* » par la fonction « *thirdTest* » que nous allons créer et rajouter un DIV nommé « *add_* » suivi du « *$post_id* » dans lequel nous insérons les nouveaux commentaires.

```php
<div id="add_<?php echo $post_id;?>"></div>
<form id="form_<?php echo $post_id;?>" method="post" action="addCommentModel.php">
    <div id="erreur_<?php echo $post_id;?>"></div>
    <div class="addCommentsContainer">
        <div class="title">Ajout d'un nouveau commentaire</div>
        <div class="descriptionContent">
            <textarea id="comment_<?php echo $post_id;?>" name="comments" cols="40"></textarea>
            <input name="post_id" type="hidden" value="<?php echo $post_id;?>">
        </div>
        <div class="buttonBox">
            <input type="button" onClick="javascript: thirdTest('<?php echo $post_id;?>');" class="submitComment" value="Commenter">
        </div>
    </div>
</form>
```

Nous allons maintenant coder la fonction qui utilisera *XMLHTTPRequest* pour ajouter le commentaire. Copiez donc le fichier « *firstJS.js* » contenu dans les sources sous « *php/step5* ». Vous y trouverez la fonction « *thirdTest* ».

```
function thirdTest(postId){
    var commentaire = document.getElementById('comment_' + postId).value;
    // Est-ce que la valeur est vide ?
    if(commentaire == ""){
        //Oui, alors affiche un message d'erreur
        var message = "<p><div class='notice'>Le commentaire ne peut pas etre vide.</div></p>";
        document.getElementById('erreur_' + postId).innerHTML = message;
    }else{
        //Non, alors appeler l'ajout
        var xhr;
        // Créer le XMLHTTPRequest
        try{
            xhr = new ActiveXObject("Microsoft.XMLHTTP");    // Essayer Internet Explorer
        }catch(e){ // Ce n'est pas Internet Explorer
            xhr = new XMLHttpRequest();  // Autres navigateurs
        }
        // Préparer l'envoie
        xhr.open('POST', 'addCommentModel.php', true); //commande GET ou POST, URL du document, true pour asynchrone.
        // Préparer l'en tete pour le POST UNIQUEMENT
        var params = "comments=" + commentaire + "&post_id=" + postId;
        xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
        xhr.setRequestHeader("Content-length", params.length);
        xhr.setRequestHeader("Connection", "close");
        // Envoie de la requête
        var params = "comments=" + commentaire + "&post_id=" + postId;
        xhr.send(params); //Parametre avec POST seulement, données à envoyer au serveur.
        //Récupérer la réponse
        xhr.onreadystatechange = function(){ // instructions de traitement de la réponse
            if (xhr.readyState == 4) {// Reçu, OK
                var response = xhr.responseText;
                if(response == "0"){ //Erreur de DB
                    message = "<p><div class='notice'>La base de donnees n'a pas repondu.</div></p>";
                    document.getElementById('erreur_' + postId).innerHTML = message;
                }else{
                    document.getElementById('comment_' + postId).value = ""; //Initialiser le formulaire
                    // Ajouter la reponse devant la partie d'ajout
                    document.getElementById('add_' + postId).innerHTML = document.getElementById('add_' + postId).innerHTML + response;
                }
            } else {
                // Attendre...
            }
        };
    }
}
```

Etudiez l'élément *XMLHTTPRequest*. Vous pouvez remarquer que la connexion se passe en 5 étapes :

- Création de l'objet, la création est différente selon que le navigateur soit Internet Explorer ou un autre navigateur.
- Préparation de l'envoie avec la méthode « *open* »
- Préparation de l'entête
- Envoie de la requête
- Récupération de la réponse, les états du *readyState* sont les suivants, seul le dernier est vraiment utile :
    - 0 : non-initialisé
    - 1 : connexion établie
    - 2 : requête reçue
    - 3 : Réponse en cours
    - 4 : terminé

Testez l'application. Votre site web n'a plus besoin d'être rechargé !

4.1.8 AJAX avec la librairie Prototype

Précédemment, nous avons vu l'utilisation d'*XMLHTTPRequest* afin d'ajouter nos commentaires. Ceci est, on l'a déjà dit, la face cachée d'AJAX. De nombreuses librairies ont tenté de simplifier ce travail.

Pour notre exemple, nous allons utiliser une des librairies les plus abouties nommée « *Prototype* ».

Tout d'abord, ajouter le fichier « Prototype.js » contenu dans les sources sous « *php/step6* » vers la racine du site. Afin de lier la librairie qui est contenu dans cet unique fichier JavaScript, il vous suffit comme précédemment de la lier dans l'entête de l'index.

```
<head>
    <title>Le 1er blog en PHP</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <link href="style.css" rel="stylesheet" type="text/css">
    <script src="firstJS.js"></script>
    <script src="Prototype.js"></script>
    </head>
```

Les JavaScripts liés dans ce site web peuvent accéder librement entre eux. Nous pourrons donc sans problème accéder depuis notre fichier « *firstJS.js* » vers la librairie.

Nous allons créer une nouvelle méthode JavaScript nommée « *fourthTest* » qui sera appelé depuis notre page « *addCommentView.php* ». Modifier donc l'événement du bouton de celui-ci.

```
<input type="button" onClick="javascript: fourthTest('<?php echo $post_id;?>');" class="submitComment" value="Commenter">
```

Ouvrez le fichier « *firstJS.js* » et ajoutez cette nouvelle méthode « *fourthTest* » recevant le post id en paramètre.

```
function fourthTest(postId){
    var commentaire = document.getElementById('comment_' + postId).value;
    // Est-ce que la valeur est vide ?
    if(commentaire == ""){
        //Oui, alors affiche un message d'erreur
        var message = "<p><div class='notice'>Le commentaire ne peut pas etre vide.</div></p>";
        document.getElementById('erreur_' + postId).innerHTML = message;
    }else{
        var page = "addCommentModel.php";
        var methode = "post";
        var params = "comments=" + commentaire + "&post_id=" + postId;
        new Ajax.Request(page,
            {
                method: methode,
                postBody: params,
                onSuccess: function (transport) {
                    var response = transport.responseText;
```

```
                    if(response == "0"){ //Erreur de DB
                        message = "<p><div class='notice'>La base de donnees n'a pas repondu.</div></p>";
                        $('erreur_' + postId).innerHTML = message;
                    }else{
                        $('comment_' + postId).value = ""; //Initialiser le formulaire
                        // Ajouter la reponse devant la partie d'ajout
                        $('add_' + postId).innerHTML = $('add_' + postId).innerHTML + response;
                    }
                },
                onFailure: function(){
                    message = "<p><div class='notice'>La base de donnees n'a pas repondu.</div></p>";
                    $('erreur_' + postId).innerHTML = message;
                }
            }
        );
    }
}
```

Comme vous pouvez le voir, les requêtes AJAX deviennent plus claires à l'aide de la librairie. En effet, il n'y a plus besoin de créer un objet en fonction du navigateur, de préparer l'entête ou de modifier la façon d'envoyer les paramètres en fonction de la méthode *Post* ou *Get* comme précédemment.

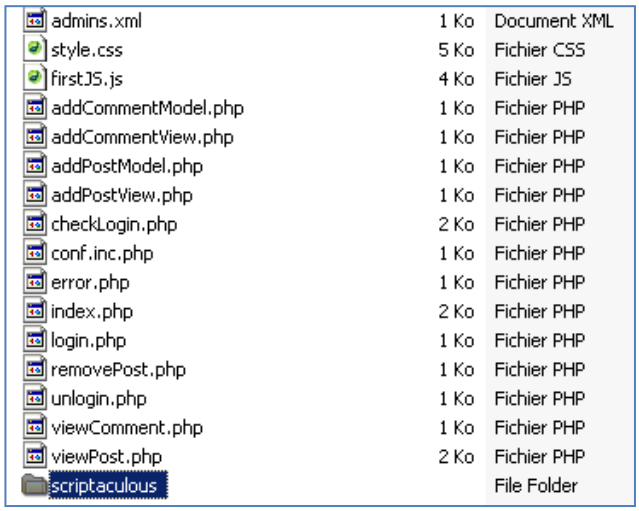
A noter que cette librairie offre un raccourci très pratique pour appeler un élément du document. Sans librairie, nous avons vu que nous devions écrire la commande « *document.getElementById* » afin d'obtenir l'élément alors qu'avec son utilisation, il nous suffit d'écrire « *$('nomElement')* ».

Tester votre application. Le résultat est toujours le même, mais de manière plus simple !

4.1.9 PHP vers RIA avec la librairie Scriptaculous

Nous avons optimisé les accès vers le serveur, maintenant nous allons utiliser une librairie JavaScript pour le graphisme afin de créer des animations pour la requête afin que l'utilisateur croie en une application standalone.

Copiez le dossier « Scriptaculous » contenus dans les sources sous « *php/step6* » vers la racine.



Ajoutez le lien « *scriptaculous/scriptaculous.js* » dans l'entête de l'index comme vu précédemment.

```
<head>
    <title>Le 1er blog en PHP</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <link href="style.css" rel="stylesheet" type="text/css">
    <script src="firstJS.js"></script>
    <script src="Prototype.js"></script>
    <script src="scriptaculous/scriptaculous.js"></script>
    </head>
```

Cette librairie offre de nombreux effets visuels. Vous pouvez en voir toute l'étendu sur le site http://script.aculo.us/.

Afin de tester un de ces effets, rajouter la ligne suivante « *new Effect.Pulsate('add_' + postId);* » après la mise à jour de l'élément comme ci-dessous (dans *firstJS.js*):

```
// Ajouter la reponse devant la partie d'ajout
$('add_' + postId).innerHTML = $('add_' + postId).innerHTML + response;
new Effect.Pulsate('add_' + postId);
```

Tester à nouveau votre application. Après l'ajout, tous les nouveaux éléments se mettent à clignoter !

4.1.10 Questions importantes

1) Quels sont les balises de PHP ?
2) Qu'est-il important de savoir concernant les variables en PHP?
3) Connaissez-vous les étapes de connexion depuis PHP vers SQL ?
4) Qu'est-ce que sont les injections SQL et comment s'en protéger ?
5) Qu'est-ce que les variables de sessions et comment les utiliser ?
6) Comment faire pour sécuriser le mot de passe des administrateurs afin qu'il ne puisse pas être lus directement en clair ?

7) Pourquoi n'y a-t-il pas besoin de l'ouverture de session « session_start » au début des pages incluses à l'aide de la commande « *require* » ou « *include* » ?
8) Comment charger un fichier CSS ?
9) Arrivez-vous imaginer l'allure des éléments après chargement d'une feuille de style ?
10) Dans quel cas peut-il être utile d'avoir plusieurs feuilles de style ?
11) Quelle est la différence entre un « button » et un « submit » ?
12) Comment charger un fichier JavaScript ?
13) Comment appeler JavaScript depuis un bouton ?
14) Comment obtenir un élément de la page en JavaScript ?
15) Qu'est-ce que l'objet XMLHTTPRequest ?
16) En quoi AJAX simplifie-t-il l'utilisation du XMLHTTPRequest ?
17) Quel est la grande restriction d'AJAX ?

TP 11 – Web framework

Django

Omar ABOU KHALED, Stefano CARRINO, Joël DUMOULIN

Buts du travail

- Mettre en place un environnement de développement basé sur un Framework web.
- Prendre en main le Framework web Django.
- Réaliser une petite application web basée sur Django.
- Juger de l'utilité d'Ajax dans une application web.

Infrastructure nécessaire

Afin de pouvoir réaliser ce travail pratique, il est nécessaire d'installer les composants suivants sur son poste de travail :

Composants	Remarques
Visual Studio 2010	Installé
Stack de développement pour Django	<Chapitre Installation>

Travail à réaliser

1. Installation
2. Hello Django !
    a. Première application django
3. Application « Todo »
    a. Création d'une application de gestion d'une liste de tâches

Rappels des concepts

Django
Django est un framework qui s'inspire du principe MVC ou MTV (la vue est gérée par un template) composé de 3 parties distinctes :

- Un langage de templates flexible qui permet de générer du HTML, XML ou tout autre format texte ;

- Un contrôleur fourni sous la forme d'un "remapping" d'URL à base d'expressions rationnelles ;
- Une API d'accès aux données est automatiquement générée par le framework compatible CRUD. Inutile d'écrire des requêtes SQL associées à des formulaires, les requêtes SQL sont générées automatiquement par l'ORM.

En plus de l'API d'accès aux données, une interface d'administration fonctionnelle est générée depuis le modèle de données. Un système de validation des données entrées par l'utilisateur est également disponible et permet d'afficher des messages d'erreurs automatiques.

Sont également inclus :

- un serveur web léger permettant de développer et tester ses applications en temps réel sans déploiement ;
- un système élaboré de traitement des formulaires muni de widgets permettant d'interagir entre du HTML et une base de données. De nombreuses possibilités de contrôles et de traitements sont fournies ;
- un framework de cache web pouvant utiliser différentes méthodes (MemCached, système de fichier, base de données, personnalisé) ;
- le support de classes intermédiaires (middleware) qui peuvent être placées à des stades variés du traitement des requêtes pour intégrer des traitements particuliers (cache, internationalisation, accès…) ;
- un support complet d'Unicode.

Par framework, Django peut être considéré comme une boîte à outils où chaque module peut fonctionner de façon indépendante. (Source : http://fr.wikipedia.org/wiki/Django_(framework))

Ajax

L'architecture informatique, Ajax (acronyme d'Asynchronous JavaScript and XML) permet de construire des applications Web et des sites web dynamiques interactifs sur le poste client. Ajax combine JavaScript, les CSS, XML, le DOM et le XMLHttpRequest afin d'améliorer maniabilité et confort d'utilisation des Applications Internet Riches.

Dans une application Web classique, chaque manipulation entraîne la transmission et l'affichage d'une nouvelle page et l'utilisateur doit attendre l'arrivée de la réponse pour effectuer d'autres manipulations.

En utilisant Ajax, le dialogue entre le navigateur et le serveur se déroule la plupart du temps de la manière suivante : un programme écrit en langage de programmation JavaScript, incorporé dans une page web, est exécuté par le navigateur. Celui-ci envoie en arrière-plan des demandes au serveur Web, puis modifie le contenu de la page actuellement affichée par le navigateur

Web en fonction du résultat reçu du serveur, évitant ainsi la transmission et l'affichage d'une nouvelle page complète.

En Ajax, comme le nom l'indique, les demandes sont effectuées de manière asynchrone : le navigateur Web continue d'exécuter le programme JavaScript alors que la demande est partie, il n'attend pas la réponse envoyée par le serveur Web et l'utilisateur peut continuer à effectuer des manipulations pendant ce temps.

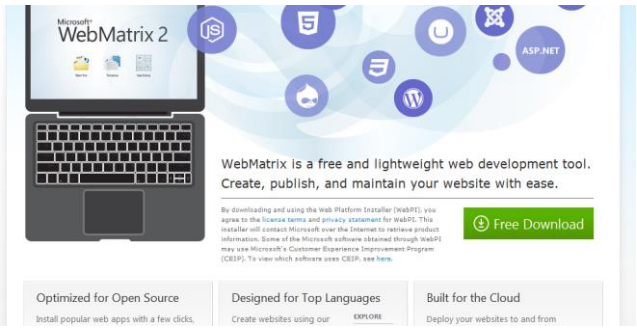
1 Installation

Il est nécessaire d'installer différents éléments afin de débuter dans le développement Django. Habituellement, ce Framework est utilisé dans un environnement Linux, mais afin de vous simplifier la tâche (éviter de devoir installer une nouvelle machine virtuelle par exemple !), nous allons rester sous Windows.

Microsoft met à disposition un outil de développement Web gratuit et léger, permettant la création, la publication ainsi que la maintenance de sites web de manière simplifiée : WebMatrix. Nous n'allons ici pas l'utiliser pour ce à quoi il est destiné, mais pensez-y lors de vos développement futurs. Nous allons en fait utiliser l'installeur de WebMatrix pour nous faciliter l'installation des composants suivants :

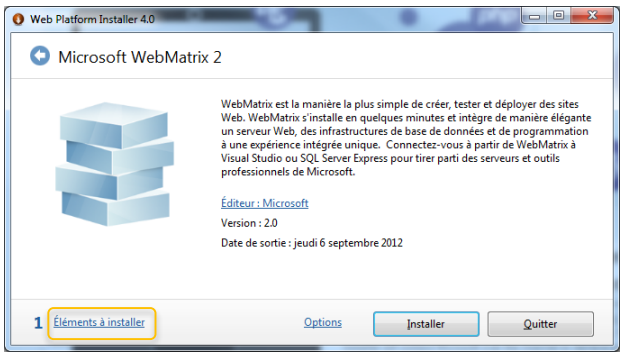
- Python
  - Langage de programmation
- Django
  - Framework de développement web en Python
- MySQL
  - Système de gestion de base de données
- Wrapper MySQL pour Python
  - Wrapper pour utiliser MySQL en Python
- Outils Python pour Visual Studio 2010
  - Outils facilitant le développement Python et Django sous Visual Studio
- PyPI
  - Outil permettant de facilement installer des paquets Python disponibles sur le dépôt PyPI

Lancez l'installation de WebMatrix à partir de http://www.microsoft.com/web/ en cliquant sur le bouton « Free Download » :

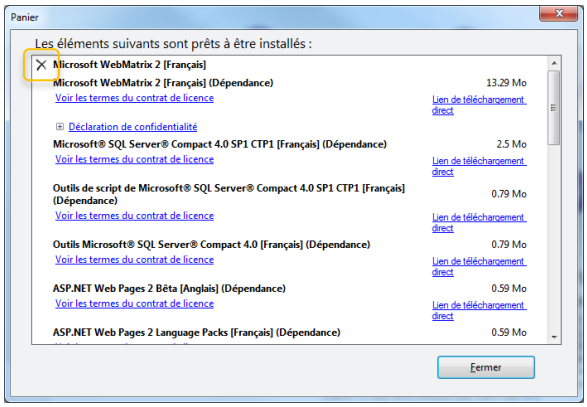


L'installeur va vouloir installer WebMatrix. Hors, nous n'en avons pas besoin (dans le cadre de ce TP) :

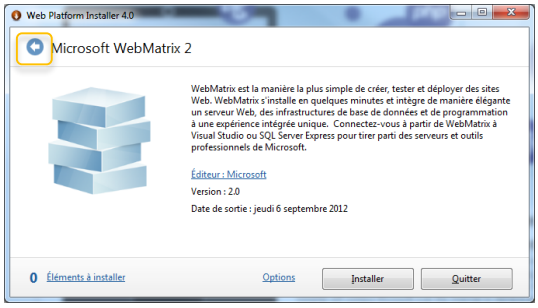
- Cliquez sur « Eléments à installer »



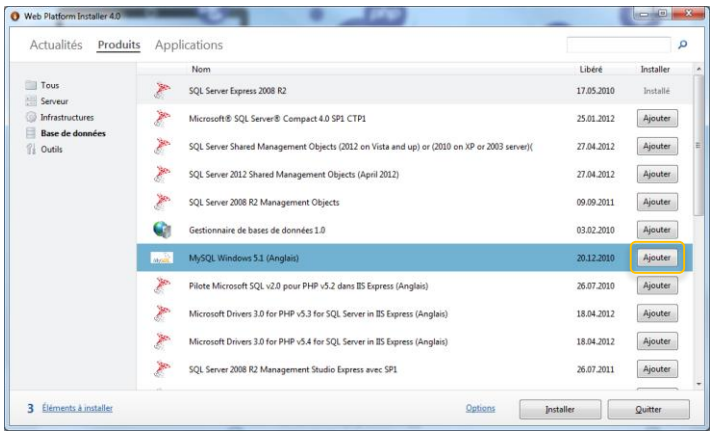
- Supprimez « Microsoft WebMatrix 2 » de la liste en cliquant sur la croix, puis cliquez sur « Fermer » :



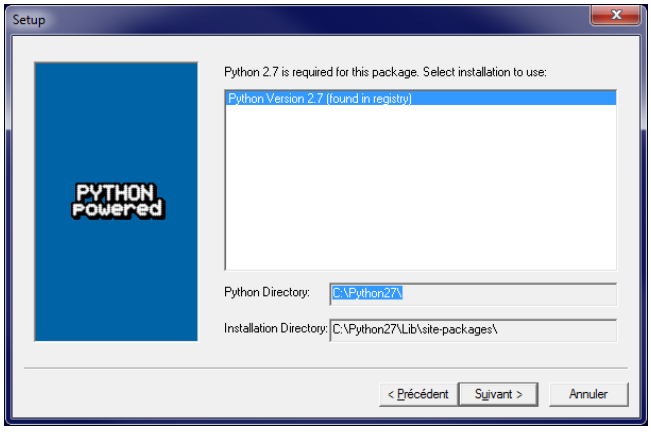
- Ensuite, cliquez sur la flèche de retour :



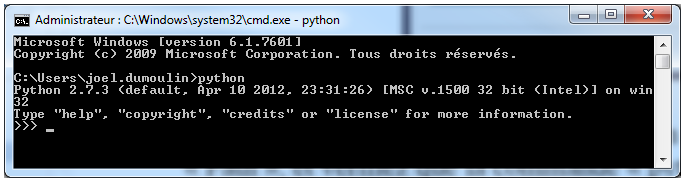
- Sous la catégorie « Produits »
  - Sous la catégorie « Infrastructures », ajoutez :
    - Python 2.7 (32-bit)(Anglais)
    - PyPI (Python)
    - Django 1.4 (Python)
    - MySQLdb (32-bit Python 2.7)(Anglais)
  - Sous la catégorie « Base de données », ajoutez :
    - MySQL Windows 5.1(Anglais)
  - Sous la catégorie « Outils », ajoutez :
    - Python Tools for Visual Studio 2010-1.5 (Anglais)



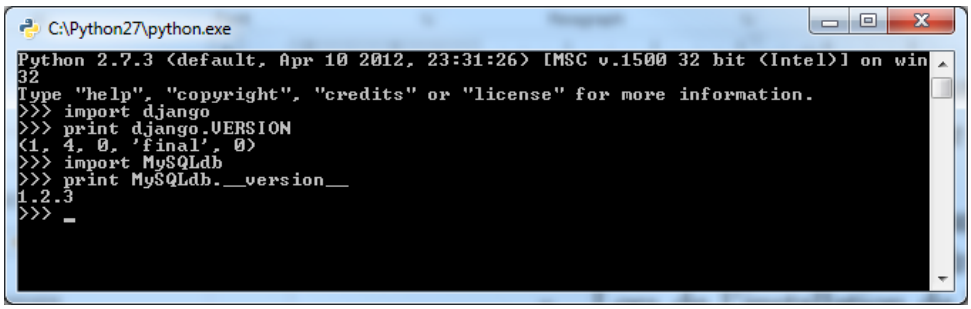
- Cliquez ensuite sur Installer
- L'installeur va vous demander un mot de passe pour MySQL, saisissez-en un
- L'installeur va ensuite vous demander d'accepter la configuration. Cliquez sur « J'accepte »
- L'installeur va maintenant télécharger automatiquement les différents composants, et les installer (cela peut prendre un certain temps)
- Lors de l'installation de MySQL pour python, le chemin d'installation de Python devrait normalement être trouvé automatiquement :



- Ajoutez le chemin d'installation de Python « C:\Python27 » à la variable d'environnement « Path », et vérifiez que la commande « python » est bien reconnue :



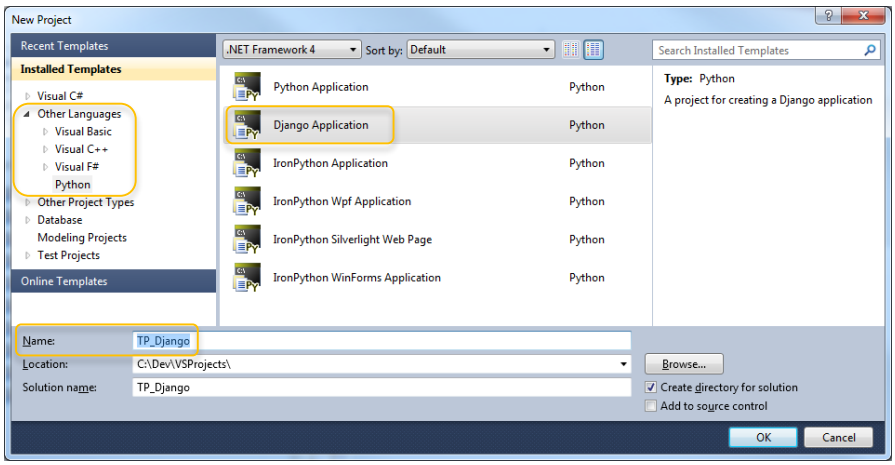
- Vérifiez l'installation de Python, Django, MySQL : démarrez une console Python et tapez les commandes suivantes :



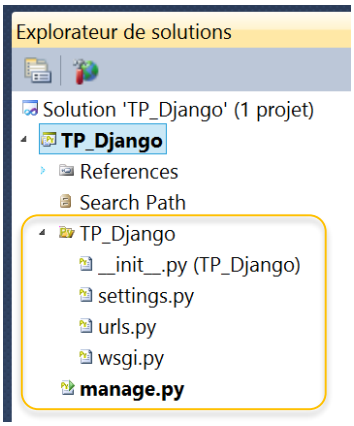
- *Facultatif* : Afin de simplifier la gestion de la base de donnée, vous pouvez installer un outil graphique comme « MySQL Workbench » (http://dev.mysql.com/downloads/workbench/5.2.html), ou encore « PhPMyAdmin » (http://www.phpmyadmin.net/home_page/index.php). Nous n'en auront cependant pas l'utilité durant ce TP.

2 Hello Django ! (~10min)

- Démarrez Visual Studio 2010, *File → New Project*
- Sous *Other Languages → Python*, sélectionnez *Django Application*, donnez-lui le nom « TP_Django », puis *OK*

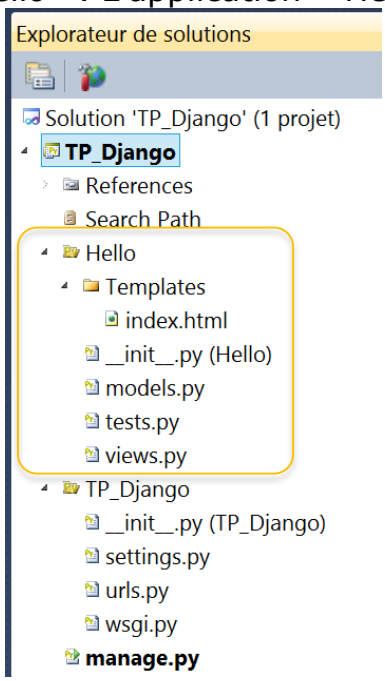


- La structure suivante est alors créée :



- Les fichiers qui nous intéressent ici sont :
  - settings.py : fichier de configuration de notre projet Django
  - urls.py : fichier de configuration des schémas d'urls
  - (manage.py : scripts de gestion en ligne de commande du projet)

- Ajoutez une nouvelle *Django app* : clic droit sur le projet → *Add new Django App…*
- Donnez-lui le nom « Hello ». L'application « Hello » est alors ajoutée :



  - Templates : dossier qui contiendra les « templates » de l'application
  - models.py : description des tables pour la partie stockage des données, c'est le « modèle »
  - views.py : logique de la page, elle est appelée « vue »
  - (tests.py : module de test)

- Tout d'abord, pour que notre application « Hello » soit chargée, il faut l'indiquer dans le fichier « settings.py », sous « INSTALLED_APPS » :

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Uncomment the next line to enable the admin:
    # 'django.contrib.admin',
    # Uncomment the next line to enable admin documentation:
    # 'django.contrib.admindocs',
    'Hello',

)
```

- Ajoutons maintenant dans la vue de notre application le code suivant :

```
from django.http import HttpResponse
from django.template.loader import render_to_string

def home(request):
    return HttpResponse(render_to_string(
                                        'index.html',
                                        {'content': 'Hello Django'}
                                        ))
```

  - Ce code va envoyer une réponse HTTP au client (`HttpResponse()`)
  - La réponse contiendra le code HTML issu du rendu (`render_to_string()`) du template « index.html »
  - Lors du rendu du template, la variable « content » qui contient le string « Hello Django » sera disponible
  - Le dictionnaire contenant dans cet exemple la clé « content » est appelé le « contexte », et peux contenir toutes les entrées dont vous avez besoin dans vos templates.
- Créons maintenant le template « index.html » : clic droit sur le dossier « Templates » → *Add New Item* → *Django HTML Template* → Donnez lui le nom « index.html » → *Add*
- Lors du rendu du template, le contenu de la variable « content » va être récupéré grâce au code `{{ content }}`

```
<html>
<head><title></title></head>

<body>

{{ content }}

</body>
</html>
```

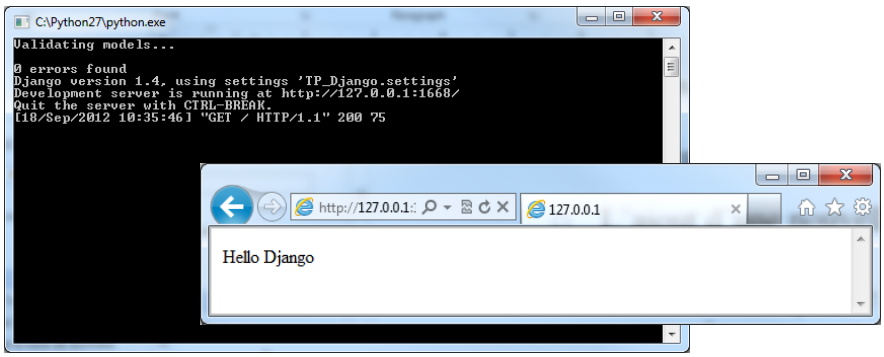
- Il nous reste maintenant à mapper une url à notre vue « home ». Pour cela, ajoutez l'entrée suivante dans le fichier « urls.py » :

```
urlpatterns = patterns('',
    # Examples:
    # url(r'^$', 'TP_Django.views.home', name='home'),
    # url(r'^TP_Django/', include('TP_Django.foo.urls')),

    # Uncomment the admin/doc line below to enable admin documentation:
    # url(r'^admin/doc/', include('django.contrib.admindocs.urls')),

    # Uncomment the next line to enable the admin:
    # url(r'^admin/', include(admin.site.urls)),
    url(r'^$', 'Hello.views.home', name='home'),
)
```

  - L'ajout d'une nouvelle entrée `url(…)` dans la liste `urlpatterns` est à répéter pour chaque vue.
  - Dans cet exemple, le `r'^$'` indique que l'url vide (ici http://localhost/) va être « attrapée » par cette règle

- C'est la vue `home` qui se trouve dans le package `Hello.views` qui va être appelée
- L'attribut `name` n'est pas obligatoire ici, mais il est recommendé de l'indiquer

- Tout est maintenant en place, il ne nous reste plus qu'à tester notre application. Django nous met à disposition un petit serveur web maison de test. On pourrait facilement le démarrer dans une console à l'aide du « manage.py » (`python manage.py runserver`), mais Visual Studio nous simplifie encore les choses ! Il nous suffit de faire F5 :



- Remarque : avec cette façon de démarrer l'application, nous pouvons également ajout des break-points et ainsi facilement débugger l'application.

3 Application : « Todo »

3.1 Création de l'application

Nous allons maintenant réaliser une petite application de gestion d'une liste de tâches.
Voici le cahier des charges de l'application :

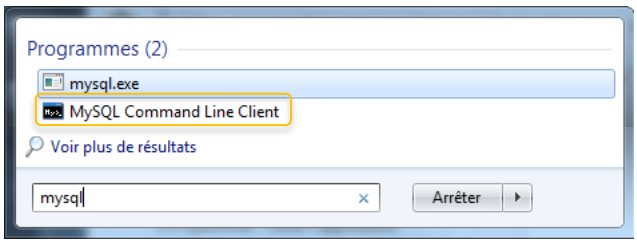
- Ajout d'une tâche
- Marquer une tâche comme terminée
- Supprimer une tâche
- Vider la liste des tâches terminées

Pour commencer, créez une Django app nomée « Todo » dans votre projet « TP_Django », et ajoutez-la dans la liste des applications installées (`settings.py`).

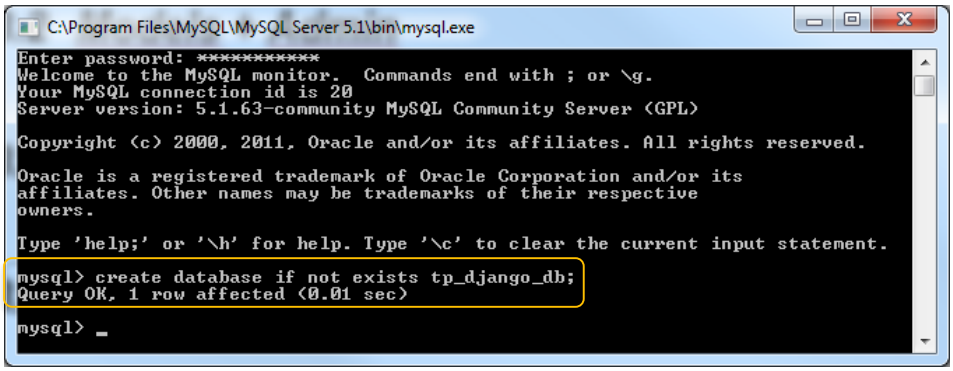
3.2 Création de la base de données (~5min)

Nous avons cette-fois besoin d'une base de données pour stocker nos tâches :

- Ouvrez une console MySQL :



- Entrez le mot de passe que vous avez définis lors de l'installation de MySQL, puis créez la base de donnée « tb_django_db » :



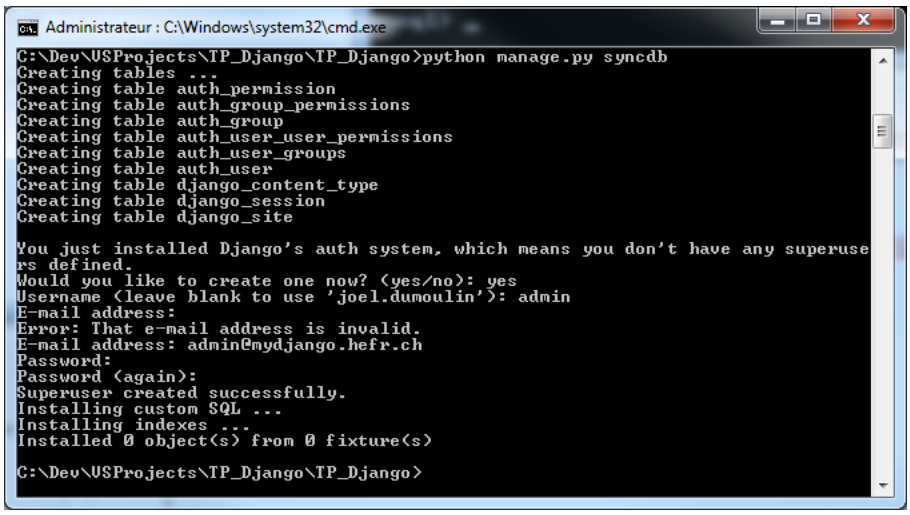
- Configurez la partie relative à la base de donnée dans le fichier `settings.py` comme suit :

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql', # Add 'postgresql_psycopg2',
                                               'postgresql', 'mysql', 'sqlite3' or
                                               'oracle'.
        'NAME': 'tp_django_db',                # Or path to database file if using
                                               sqlite3.
        'USER': 'root',                        # Not used with sqlite3.
        'PASSWORD': 'yourPwd',                 # Not used with sqlite3.
        'HOST': '',                            # Set to empty string for localhost.
                                               Not used with sqlite3.
        'PORT': '',                            # Set to empty string for default. Not
                                               used with sqlite3.
    }
}
```

- Dans une console, placez vous à la racine de votre projet Django (là ou se trouve le module `manage.py`), et entrez la commande suivante :

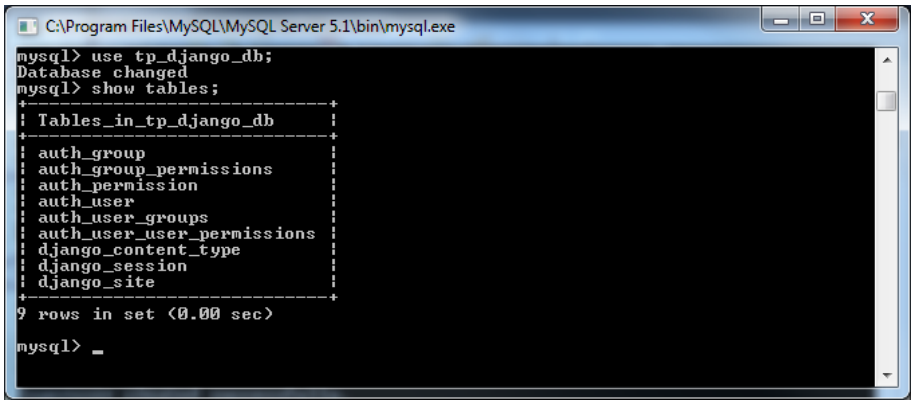
```
python manage.py syncdb
```

  - Après avoir créé les tables de base pour l'application Django, il va nous demander, si c'est la 1ère fois que l'on exécute cette commande, de configurer le « superutilisateur » :



- Tapez « yes »
- Entrez un nom de superutilisateur, par exemple « admin »
- Entrez une adresse mail
- Entrez un mot de passe, puis confirmez-le

- Vérifiez que les tables ont bien été créées dans la base :



3.3 Models + Admin *(~10min)*

On va ici commencer par définir notre modèle de données pour stocker la liste de tâches.

- Dans le module models.py, ajoutez :

```
from django.db import models

class Task(models.Model):
    content = models.CharField('task', max_length=255)
    is_resolved = models.BooleanField('Resolved?')

    def __unicode__(self):
        return 'Task %d : %s' % (self.id, self.content)
```

  - Crée le modèle d'une tâche `Task`
  - Définit un champ `content` pour l'énoncé de la tâche
  - Définit un champ `is_resolved` pour indiquer si la tâche est résolue ou non

- La méthode `__unicode__` permettra d'avoir un affichage propre dans la partie administration que nous verrons plus tard

On va maintenant activer la fonctionnalité d'administration de Django :

- Décommentez l'application `'django.contrib.admin'` dans la liste `INSTALLED_APPS` dans le fichier `settings.py`
- Dans urls.py, décommentez les lignes correspondantes pour activer l'administration :

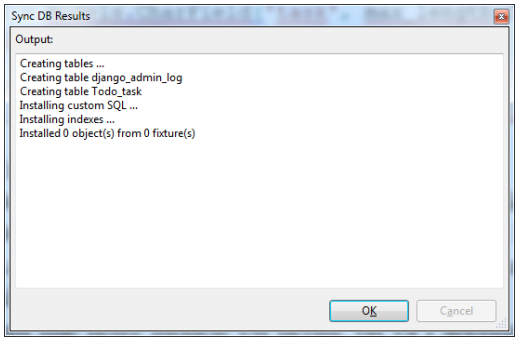
```
# Uncomment the next two lines to enable the admin:
from django.contrib import admin
admin.autodiscover()
…
# Uncomment the next line to enable the admin:
url(r'^admin/', include(admin.site.urls)),
```

- Pour pouvoir administrer un modèle que nous avons créé
  - Créez le fichier `admin.py` dans notre application « Todo » et ajoutez-y :

```
from django.contrib import admin
from Todo.models import Task

admin.site.register(Task)
```

  - Remarque : Pour chaque modèle que nous voudrions administrer, il faut ajouter cet enregistrement `admin.site.register(MyModel)`.

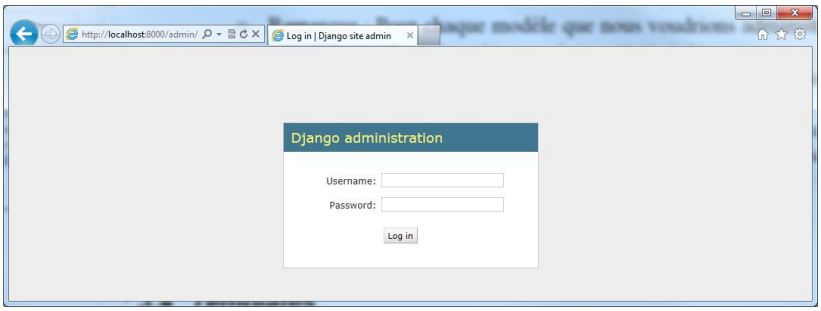
- Maintenant que notre modèle est défini et que l'on a préparé la partie relative à l'administration, on va appliquer les modifications à la base de données avec la commande `syncdb`. On peut la lancer en ligne de commande comme vu précédemment, ou dans Visual Studio en faisant clic-droit sur le projet → *Django Sync DB…*



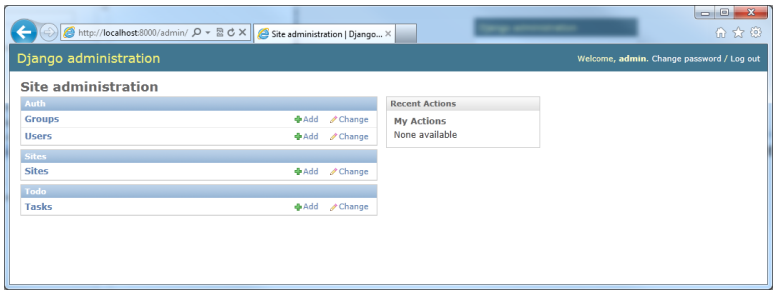
- La table Todo_task est ainsi automatiquement créée, avec les bon types en fonction du SGBD (ici MySQL). On pourrait donc changer de SGBD sans avoir à changer le code du modèle.
- Démarrez le serveur (depuis Visual Studio ou en ligne de commande `python manage.py runserver`)

- Remarque : la commande runserver va démarrer le serveur sur le port 8000, alors que le lancement depuis VisualStudio va choisir un port au hasard.

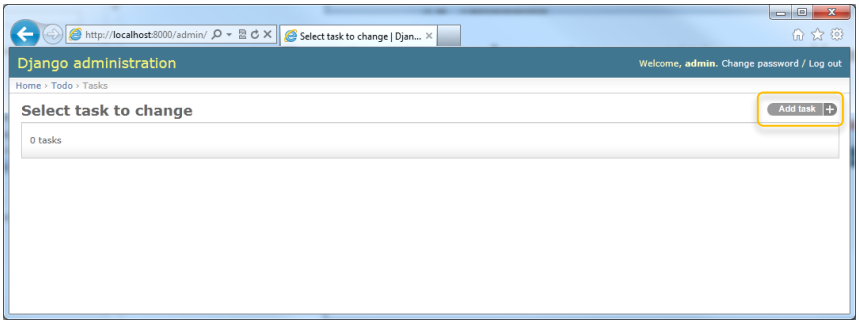
- Aller à l'adresse http://localhost :8000/admin/ (attention au n° de port !)

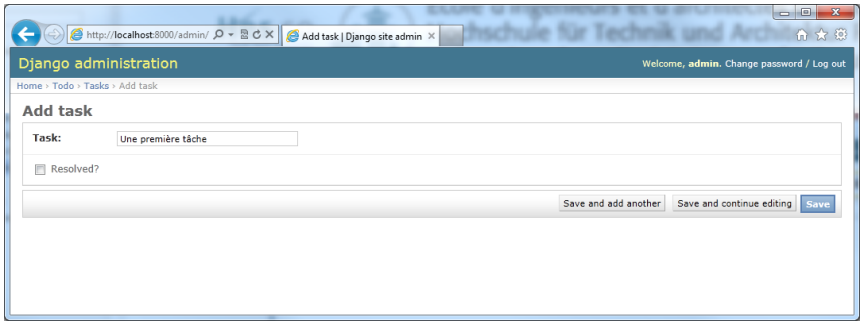


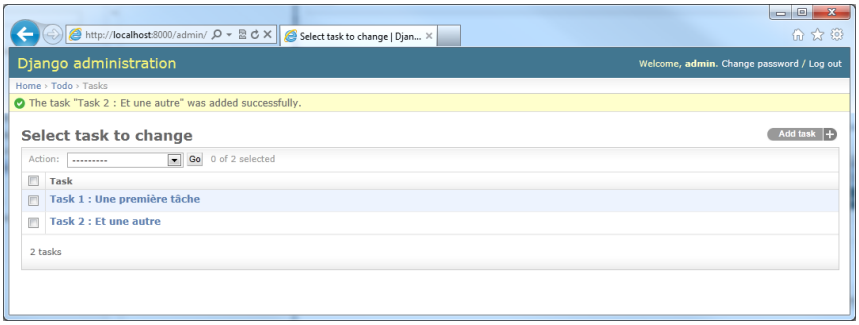
- Connectez-vous avec le compte `superutilisateur` créé précédemment lors de l'execution de la commande `syncdb`.



- On peut aller ajouter des entrées dans notre modèle : *Todo* → *Tasks*







- Cet outil d'administration est extrêmement pratique !
- Remarque : pour des fonctionnalités encore plus avancées (filtres, etc.) dans la console d'administration, ainsi qu'un thème visuel agréable : http://www.grappelliproject.com/

3.4 Affichage de la liste des tâches (~20min)

Maintenant que notre modèle est en place, nous allons nous occuper de l'affichage de la liste des tâches.

Template

- Créez le template `task-list.html` et ajoutez-y le code suivant :

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>'Todo list'</title>
</head>

<body>

    <section id="todoapp">
        <header id="header">
            <h1>Todos</h1>
            <input id="new-todo" placeholder="What needs to be done?" autofocus>
        </header>

        <section id="main">
            <ul id="todo-list">

                <!-- C'est ici que ca se passe -->

            </ul>
        </section>

        <footer id="footer">
            <span id="todo-count"></span>
            <button id="clear-completed">Clear completed</button>
        </footer>
    </section>
</body>
</html>
```

Vue

- Ajoutez la vue suivante :

```
from django.views.generic import TemplateView

class TasksView(TemplateView):
    template_name="tasks-list.html"
```

- Remarque : on utilise ici une vue générique, ce qui nous permet d'économiser du code ! (https://docs.djangoproject.com/en/1.4/topics/generic-views/)

Url

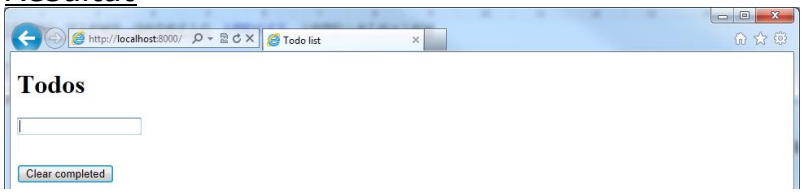
- Il ne faut pas oublier d'ajouter le mapping d'une url vers cette vue (mettez en commentaire l'url pour l'application `Hello` ou mappez-là sur une autre url) :
  - On importe d'abord notre vue

```
from Todo.views import TasksView
```

  - Et on ajoute ensuite un nouveau pattern

```
url(r'^$', TasksView.as_view(), name='tasks-list'),
```

Résultat



Afficher les tâches

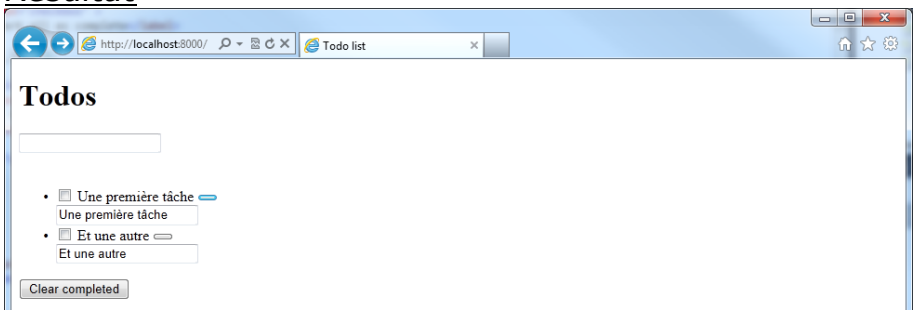
- Modifiez la vue (attention, on modifie également le type de la vue pour en faire une `ListView`) :

```
from django.views.generic import ListView
from Todo.models import Task

class TasksView(ListView):
    template_name="tasks-list.html"
    model = Task
```

- Dans le template, modifiez l'élément todo-list :

```
<ul id="todo-list">
    {% for task in object_list %}
        <li>
            <div class="view">
                <input class="toggle" type="checkbox">
                <label>{{ task.content }}</label>
                <button class="destroy"></button>
            </div>
            <input class="edit" value="{{ task.content }}">
        </li>
    {% endfor %}
</ul>
```

Résultat



Css

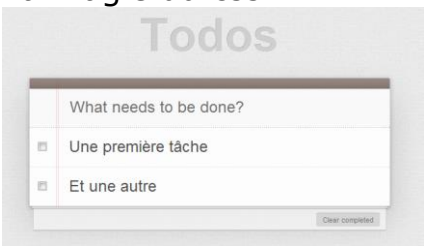
Ajoutons une feuille de style css pour mettre en forme l'affichage…

- A la racine de l'application « Todo », créez un dossier « static » et copiez à l'intérieur de ce dernier les fichier `base.css` et `bg.png` disponibles dans les sources du TP.
- Modifiez le début du template pour charger le fichier css :

```
{% load staticfiles %}
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>Todo list</title>
        <link rel="stylesheet" href="{% static 'base.css' %}">
</head>
```

Résultat

La magie du css :



3.5 Fonctionnalités supplémentaires (~15min)

3.5.1 Ajouter une tâche

Formulaire

- Mettez en place un formulaire pour éditer le modèle
  - Créez le fichier « forms.py » à la racine de l'application « Todo » et copiez-y le code suivant :

```
from django import forms
from Todo.models import Task

class TaskForm(forms.ModelForm):
    class Meta:
        model = Task
        exclude = ('is_resolved',)
```

Vue

- Ajoutez la vue suivante :

```
# [...] On rajoute au document le code suivant

from django.views.generic import ListView, CreateView
from django.core.urlresolvers import reverse_lazy
from django.http import HttpResponseRedirect
from Todo.forms import TaskForm

class TaskCreateView(CreateView):
    form_class = TaskForm
    success_url  = reverse_lazy('tasks-list')

    def form_invalid(self, form):
        # Attention les erreurs du form ne seront pas affichees
        return HttpResponseRedirect(self.success_url)
```

- Pour l'instant la seule erreur possible, c'est que le champ soit vide. Dans notre cas, la tâche ne sera pas sauvegardée si on renvoie un `content` vide.
- On remarque que l'url de succès n'est pas tapée en dur, mais elle est matchée grâce au nom `tasks-list` donné dans le fichier `urls.py`. Un seul endroit est donc concerné si l'on veut modifier le schéma d'url !

Url

- Modifiez le fichier d'urls :

```
from django.conf.urls.defaults import patterns, include, url

# Uncomment the next two lines to enable the admin:
from django.contrib import admin
admin.autodiscover()

from Todo.views import *

urlpatterns = patterns('',
    # Uncomment the next line to enable the admin:
    url(r'^admin/', include(admin.site.urls)),

    #url(r'^$', 'TP_Django.Hello.views.home', name='home'),

    url(r'^$', TasksView.as_view(), name='tasks-list'),
    url(r'^add/$', TaskCreateView.as_view(), name='task-create'),
)
```

Template

- Modifiez l'élément header comme suit :

```
<header id="header">
    <h1>Todos</h1>
    <form action="{% url task-create %}" method="post">
    {% csrf_token %}
        <input id="new-todo" placeholder="What needs to be done?" name="content"
            autofocus>
    </form>
</header>
```

- On remarque à nouveau que l'url pour la création d'une tâche n'est pas tapée en dur, mais elle est récupérée automatiquement avec `{% url task-create %}`
- Le `csrf_token` est une sécurité de Django pour éviter qu'un script malveillant envoie notre formulaire sans le charger au préalable.
- Le navigateur sait que lorsqu'on appuie sur ENTER il doit envoyer le formulaire.

- Testez cette nouvelle fonctionnalité

3.5.2 Marquer une tâche comme terminée

Vue

```
from django.shortcuts import get_object_or_404

def toggle_task(request, task_id):
    if request.method == 'POST':
        # Modify an object in POST only
        task = get_object_or_404(Task, pk=task_id)

        task.is_resolved = not task.is_resolved
        task.save()

    return HttpResponseRedirect(reverse_lazy('tasks-list'))
```

Url

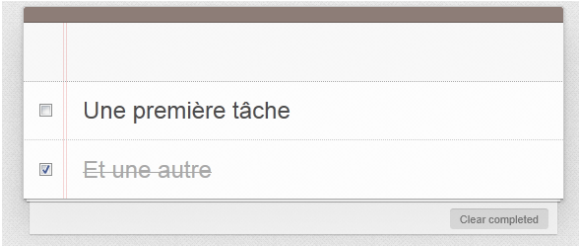
```
url(r'^toggle/(?P<task_id>\d+)/$', toggle_task, name='task-toggle'),
```

- La partie `(?P<task_id>\d+)` va récupérer dans l'url l'id de la tâche, et la passer en paramètre (`task_id`) à la vue

Template

```
<ul id="todo-list">
    {% for task in object_list %}
        <li {% if task.is_resolved %} class="completed"{% endif %}>
            <div class="view">
                <form method="post" action="{% url task-toggle task.id %}">
                {% csrf_token %}
                    <input class="toggle" type="checkbox"{% if task.is_resolved %}
                        checked="checked"{% endif %} onclick="this.parentNode.submit();">
                </form>
                <label>{{ task.content }}</label>
                <button class="destroy"></button>
            </div>
            <input class="edit" value="{{ task.content }}">
        </li>
    {% endfor %}
</ul>
```

Résultat



3.6 A vous de jouer

3.6.1 Supprimer une tâche (~15min)

- Voici le code de la vue :

```
from django.views.generic import DeleteView
class TaskDeleteView(DeleteView):
    model = Task
    success_url = #TASK_LIST_URL
```

- Ajoutez le code nécessaire au bon fonctionnement de cette nouvelle fonctionnalité (Remarque : le nom du paramètre dans la configuration url doit être « pk »)

3.6.2 Supprimer les tâches terminées (~15min)

- Ajoutez par vous-même cette fonctionnalité (bouton « Clear completed »)
- N'hésitez pas à demander de l'aide, que ça soit au Web ou aux responsables du TP !
- Tips :
  - Commencez par tester que la méthode de la requête soit bien « `POST` »
  - Ensuite, utilisez la méthode « `filter()` » sur le modèle, puis « `delete()` »

3.6.3 Ajax (~20min)

La librairie Dajax permet d'utiliser facilement Ajax dans une application Django. Jetez un œil aux différentes démos sous http://www.dajaxproject.com/random/.

Sans la coder, expliquez une fonctionnalité qui pourrait être utile dans notre petite application où l'utilisation d'Ajax serait indiquée/utile, et expliquez pourquoi. (Optionnel : codez cette fonctionnalité.)

3.7 Optionnel : Tester l'application

Il est possible de facilement mettre en place des tests fonctionnels de nos vues. Le module `tests.py` est là pour ça !

Setup

- Pour commencer, nous allons préparer la procédure de test :

```
from django.test import TestCase
from django.core.urlresolvers import reverse

from Todo.models import Task

class TodoTest(TestCase):

    def setUp(self):
        self.task1 = Task.objects.create(content=u'Ma première tâche', is_resolved=True)
        task2 = Task.objects.create(content=u'Ma seconde tâche', is_resolved=False)
        task3 = Task.objects.create(content=u'Ma troisième tâche', is_resolved=True)
        task4 = Task.objects.create(content=u'Ma quatrième tâche', is_resolved=False)
```

Tâches

Pour chaque tâche que nous voulons tester, nous ajoutons une méthode de test. Par exemple

- Affichage de la liste

```
def test_task_list(self):
    url = reverse('tasks-list')
    response = self.client.get(url)
    self.assertEqual(response.status_code, 200)
    self.assertEqual(len(response.context['object_list']), Task.objects.count())
```

- Création d'une tâche

```
def test_task_creation(self):
    url = reverse('task-create')
    nb_tasks = Task.objects.count()
    response = self.client.post(url, {'content': u'Ma cinquième tâche'})
    self.assertEqual(nb_tasks+1, Task.objects.count())
```

- Suppression d'une tâche

```
def test_task_delete(self):
    task_id = self.task1.id
    url = reverse('task-delete', args=[task_id])

    nb_tasks = Task.objects.count()
    response = self.client.post(url)
    self.assertEqual(nb_tasks-1, Task.objects.count())

    response = self.client.post(url)
    assertEqual(response.status_code, 404)
```

Tester

Ensuite, pour lancer les tests : en console : `python manage.py test Todo`. Remarque : la BD `test_tp_django_db` est créée temporairement (et supprimée à la fin des tests, afin de ne pas toucher à la BD de production.

4 A rendre

Une archive contenant le projet Django ainsi qu'un document contenant les codes et réponses du point A vous de jouer est à déposer sur Moodle. Il est demandé d'expliquer vos codes. De plus, dessinez un schéma précisant le modèle MVC particulier de Django (MVT), et intégrez-le a votre document.

Références

[1]	WebMatrix	http://www.microsoft.com/web/
[2]	Django	https://www.djangoproject.com/ http://www.django-fr.org/
[3]	Django Story	http://django-story.ionyse.com/
[4]	Dajax	http://www.dajaxproject.com/

TP – Other Frameworks

Zend framework (PHP), Ruby on Rails (Ruby), Django (Python)

Joël DUMOULIN, Stefano CARRINO, Omar ABOU KHALED

But du travail

1. Familiarisation avec un framework de développement web
2. Développement d'une petite application web

Organisation

Ce TP est un peu particulier, car les différents binômes ne le réaliseront pas tous avec la même technologie. Les binômes seront en effet répartis dans les trois frameworks web suivants : Zend framework, Ruby on Rails et Django.

Par contre, le travail sera similaire pour tout le monde, et est organisé comme suit (~8 périodes à disposition) :

1. Installation du framework et des outils nécessaires, et vérification de l'environnement de développement (1-2 périodes)
2. Réalisation d'une application de base à l'aide du tutoriel officiel du framework (3 périodes)
3. Réalisation d'une application simple, afin de mettre en pratique les concepts vus lors du tutoriel (3+ périodes)

Finalement, les binomes se regrouperont par technologies, afin de préparer et faire une présentation à la classe (10 min) de la technologie (concepts généraux, étapes de développement, vos avis, etc.).

Tutoriels **(avec instructions d'installation)**

- Zend framework : http://framework.zend.com/manual/2.3/en/ref/overview.html
- Ruby on Rails : http://guides.rubyonrails.org/getting_started.html
- Django : https://docs.djangoproject.com/en/1.7/intro/install/

Application à réaliser

Réalisez une application de gestion de tâches avec les fonctionnalités suivantes :

- Ajout + Edition + Suppression d'une tâche
- Marquer une tâche comme complétée
- Supprimer toutes les tâches complétées
- Affichage du nombre de tâches qui ne sont pas complétées

Remarques :

- Les tâches doivent être stockées dans une base de données (SQLite, MySQL, etc.).
- Vous pouvez vous inspirer de http://todomvc.com/examples/angularjs/#/

Présentation

Chacun des 3 groupes (1 par technologie) devra préparer une présentation qui résume les concepts généraux ainsi que les différentes étapes de développement d'une application web à l'aide du framework qu'il aura utilisé. L'application d'un des binome devra également être présentée.

Les présentations (8 minutes + 2 minutes de questions) se dérouleront durant la première période du cours du **5 janvier**. Chaque étudiant doit participer.

Conseils

- Il est fortement conseillé de développer sous un environnement Unix (Linux ou Os X). Cela facilite grandement l'installation, ainsi que l'utilisation en ligne de commande.
- Même si il est tout à fait possible de réaliser ce TP avec n'importe quel éditeur de texte ou IDE, c'est l'occasion d'essayer un éditeur dédié à la technologie. Par exemple (liste non-exhaustive) :
    - Zend framework
        - Zend Studio: http://www.zend.com/fr/products/studio/ (Free 30-day trial)
    - Ruby on Rails
        - Aptana RadRails: http://www.aptana.com/products/radrails.html
    - Django
        - JetBrains PyCharm: http://www.jetbrains.com/pycharm/ (Free 30-day trial)

Synthèse

Créez une archive, contenant votre projet complet, la présentation de groupe, ainsi qu'un rapport contenant vos réflexions et une synthèse du TP, et déposez-la sur Moodle.

Références

[1] http://framework.zend.com
[2] http://rubyonrails.org
[3] https://www.djangoproject.com