

Extraction et présentation graphique d'informations à travers Internet

Guillaume Fabien

Impasse du Bugnon 4

1752 Villars-sur-Glâne

fabien.guillaume@unifr.ch

Table des matières

1. Introduction.....	4
1.1. Problématique du sujet	4
1.2. Buts du travail	4
1.3. Questions traitées et manière de procéder	4
2. Analyse	5
2.1. Accès aux données	5
2.2. Affichage de documents via le web	7
2.3. Différences thin client / fat client.....	8
2.4. Choix d'une option	10
3. Présentation de la méthode choisie	11
3.1. Choix techniques	11
3.2. Java.....	12
3.3. JDBC	12
4. Mode d'emploi de l'applet.....	13
4.1. Afficher le développement du cours d'une action particulière	14
4.2. Afficher le développement du portefeuille du joueur.....	15
4.3. Comparer les cours de deux actions	15
5. Conclusion.....	16
6. Références	17

Table de illustrations

Figure 1: Modèle 2-tiers	6
Figure 2: Modèle 3-tiers	7
Figure 3: Tableau comparatif thin client/fat client.....	10
Figure 4: Evaluation des modèles thin client et fat client.....	11
Figure 5: Chargement d'une applet java	12
Figure 6: Mécanisme de fonctionnement de JDBC.....	13
Figure 7: Sélection du cours d'une action	14
Figure 8: Affichage du portefeuille du joueur	15
Figure 9: Comparaison de deux actions	16

Introduction

1.1. Problématique du sujet

La question principale traitée dans ce travail de séminaire est la suivante: quelles sont les différentes options actuellement disponibles pour afficher des données via le web, et quelle est la meilleure d'entre elles? Cette recherche du résultat le plus satisfaisant vise à aider l'utilisateur final (client) à se faire une idée précise du sujet traité: on part en effet du principe qu'un graphique de cours boursier est plus "parlant" que l' étalage brut de cours extraits d'une base de données. Mais cet output graphique engendre certains désavantages (temps de chargement plus long, mise en forme du graphique) dont il faut tenir compte et qui ne doivent pas décourager l'utilisateur d'atteindre son but initial, c'est-à-dire la recherche d'informations qui lui semblent pertinentes.

L' idée initiale de cet exposé était de proposer aux joueurs du Jeu Boursier des Universités Suisses (BSU, *Börsenspiel der Schweizer Universitäten*) un outil leur permettant d'analyser les cours d'actions sous forme graphique plutôt que sous forme brute de texte. BSU est une association d' étudiants qui a pour but d'offrir aux autres étudiants des hautes écoles la possibilité d'approfondir leurs connaissances sur des sujets économiques divers. Elle propose notamment chaque année un jeu de simulation boursière sur Internet dans lequel les participants s'efforcent de faire fructifier un capital de départ fictif. C'est dans le cadre de ce jeu que sera inséré le programme issu du travail de séminaire.

1.2. Buts du travail

La finalité de ce travail est de déterminer et de choisir la meilleure façon d'afficher graphiquement chez le client des informations tirées d'une base de données relationnelle. Ce séminaire a une application directe, puisque le résultat obtenu sera utilisé pour la version web du Jeu Boursier des Universités Suisses.

1.3. Questions traitées et manière de procéder

La définition du meilleur moyen de procéder passe par la description des différentes options disponibles, l'analyse de celles-ci pour finalement faire un choix définitif et passer à son implémentation logicielle. Les différents points traités et analysés dans cet exposé seront les suivants:

- Y-a-t-il un avantage à utiliser le web pour afficher des données? (phase générale permettant de prouver les bénéfices que l'on peut tirer de l'emploi du web comme moyen de publication de données)

- Quelle alternative choisir pour afficher des graphiques: générer des graphiques basés serveur ou client? Ce choix ne dépend-il que de la volonté du client d'avoir des graphiques dynamiques plutôt que statiques, ou alors est-ce que d'autres critères (emploi du réseau, des ressources physiques du serveur) doivent-ils rentrer en ligne de compte? Il n'existe probablement aucune option qui soit la meilleure dans tous les cas de figure. Chaque méthode a des avantages et des désavantages: le problème est de trouver une solution qui maximise les premiers et minimise les derniers.

2. Analyse

2.1. Accès aux données

L'accès aux données depuis une interface web est en soi une évolution de l'architecture client-serveur. D'un point de vue strictement technologique pourtant, ces applications web liées à une base de données sont simplement une autre forme d'applications distribuées client-serveur. Mais du côté utilisateur, on peut remarquer un changement important: d'une information généralisée transmise à tous les utilisateurs en même temps, on est passé à une personnalisation des données pour chaque client. Les entreprises maintiennent maintenant d'énormes bases de données dans lesquelles sont enregistrées les profils de chacun de leurs clients et sont à même de leur fournir des renseignements adaptés à leurs besoins. Cela démontre bien la nécessité absolue d'avoir de puissants systèmes de gestion de bases de données, mais surtout la valeur des données qu'elles contiennent!

L'accès aux données depuis une interface web a grandement bénéficié de l'émergence des technologies de programmation et de liaison de systèmes jusqu'alors incompatibles ainsi que de la détermination de standards acceptés par tous. Ainsi le langage SQL (*Structured Query Language*) est aujourd'hui un standard pour l'accès aux bases de données relationnelles, alors qu' ODBC (*Open Database Connectivity*) peut être décrit comme la norme en matière de communication de données entre client base de données.

On distingue aujourd'hui 2 types essentiels d'accès aux bases de données depuis le web:

Modèle 2-tiers: dans ce type d'architecture entrent en jeu un serveur web et un module de connexion à une base de données. Le principe est centré sur les outils de bases de données relationnelles. Il y a un avantage important avec cette méthode: la facilité de développement. De plus, l'application est écrite comme si cette base de données était en mode local. Par contre, du point de vue négatif, on s'aperçoit que ce modèle implique une dépendance presque totale des modules d'application vis-à-vis de la base de données qui est le plus souvent propriétaire. L'administration est relativement complexe, puisque la couche application est physiquement répartie sur

plusieurs postes clients. Les performances sont assez faibles, car les requêtes SQL sont directement envoyées sur le réseau, ce qui implique aussi un niveau de sécurité des données peu élevé [D-tec 98].

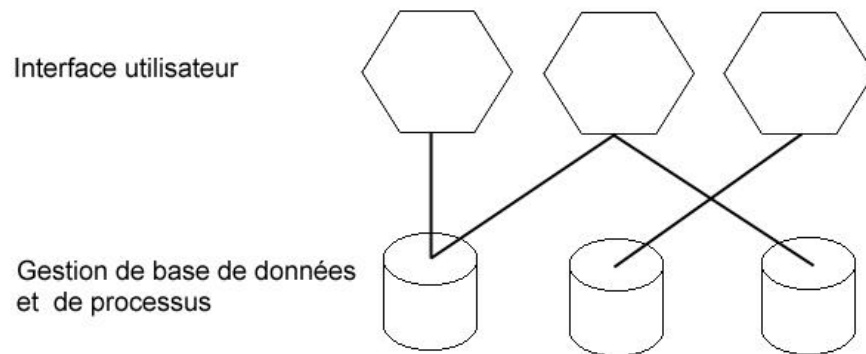


Figure 1: Modèle 2-tiers

Modèle 3-tiers (appelé aussi modèle n-tiers): 3 intervenants existent dans ce cas de figure: un serveur web, un serveur d'application et une base de données. Par rapport au modèle 2-tiers, on s'aperçoit que l'administration est nettement moins complexe, puisque les applications peuvent être gérées ou modifiées centralement sur le serveur. Il y a une nouvelle couche par rapport au modèle 2-tiers: c'est la couche de gestion de processus. Ces derniers peuvent être très divers: par exemple un programme java de type "servlet" qui gère les questions de sécurité ou renvoie à l'utilisateur des données extraites d'une base de données. Les performances s'en retrouvent accrues car seuls les appels de services et les réponses transitent par le réseau. En effet, les applications de type "thin-client" sont facilement téléchargeables et les appels de service répartissent la charge sur un ou plusieurs serveurs.

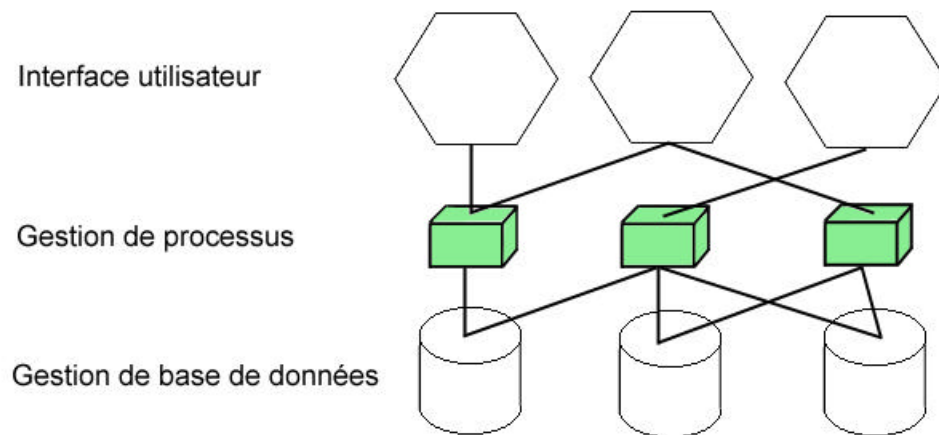


Figure 2: Modèle 3-tiers

2.2. Affichage de documents via le web

L'affichage électronique de documents via le web poursuit principalement 3 buts¹:

- *rendre un document consultable*: ceci permet à presque tous les documents (pas seulement ceux au format HTML) d'être consulté à l'aide d'un browser. Il n'y a aucun travail de conversion à fournir du côté du compositeur de documents ou du webmaster, puisqu'il a l'assurance presque totale que les browsers pourront afficher ces fichiers. Le désavantage de cette facilité d'édition se situe bien naturellement du côté du browser, puisque celui-ci, dans le cas où il ne peut afficher un document, doit démarrer si possible l'application dans laquelle celui-ci a été rédigé. Ceci engendre une consommation accrue de mémoire vive (RAM, *random access memory*). On remarque que dans ce cas l'accroissement de la facilité d'édition du côté utilisateur engendre également une augmentation de la demande de ressources.
- *permettre l'édition en-ligne*: le point étudié ci-dessus permet un accès en lecture seule aux documents, tandis que le droit d'éditer en-ligne des données offre un environnement de travail bi-directionnel. On passe de l'édition de documents à un processus d'échange et de corrections d'informations. Ainsi les relations entre clients et fournisseurs s'en trouvent grandement simplifiées (une commande erronée est corrigée par le client puis renvoyée à son fournisseur).

¹ l'emploi du terme *document* englobe la notion d'affichage de données traitée dans ce travail de séminaire

- *faciliter une communication personnalisée*: L'étape suivante est la génération automatique de documents personnalisés pour le client. L'utilisateur peut ainsi lire une page web puis faire une requête. Le browser traite ensuite cette requête (il doit dans certains cas accéder à une base de données) puis renvoie les informations obtenues au client. Le contenu est adapté à un utilisateur unique qui s'est identifié en s'inscrivant sur le système. Dans le cas de ce séminaire, seul le joueur concerné à accès aux graphiques représentant l'évolution de son portefeuille durant une période.

2.3. Différences thin client / fat client

Faire la différence entre des graphiques générés côté client ou serveur revient à peu près à distinguer les deux systèmes *thin client* et *fat client*. Voici une explication de ces deux méthodes ainsi que leurs avantages et inconvénients [TCC 00]:

Thin Client: Cette approche exige que seul l'interface graphique utilisateur (GUI, *Graphical User Interface*) soit active sur le client, tandis que l'application de base et la base de données tournent du côté serveur. Ce système est tout de même préconisé pour des documents qui requièrent peu d'accès à une base de données. Le thin client communique avec un serveur web par l'intermédiaire du protocole HTTP. Le rôle principal du serveur est donc de délivrer des fichiers sous forme HTML au client. Le serveur peut aussi exécuter un script CGI (*Common Gateway Interface*) qui accèdera à des données disque (bases de données, informations systèmes...). Le problème inhérent à cette technique est que toutes ces opérations (requête du client, transfert par HTTP, exécution du CGI) ne sont pas liées à un processus commun, mais exécutées indépendamment. Le serveur peut voir ses performances fortement dégradées lorsqu'un trop grand nombre de CGI sont exécutés simultanément.

Dans le cas où les données doivent être extraites d'une base de données, il y a la nécessité d'avoir un processus séquentiel qui ouvre la base de données, trie les données désirées et referme la base. Ce processus séquentiel, souvent sous la forme d'un programme produit exclusivement à ce propos, demande beaucoup moins de travail et est optimisé pour ces tâches, ce qui n'est pas le cas d'un CGI (celui-ci n'est en effet pas capable d'ouvrir une base de données, exécuter une requête et ensuite se mettre en attente de la prochaine requête).

Fat Client: lorsque l'on passe d'un document statique à un document dynamique, on s'aperçoit que HTML n'est plus à même de répondre à ce besoin supplémentaire. Il n'est en effet pas possible de créer des listes déroulantes ou des menus en HTML, et pourtant ces fonctions sont primordiales dans des sites web qui offrent une interactivité client-serveur. On doit alors insérer dans ces pages web des plug-ins. Ces plug-ins sont stockées sur le serveur et sont téléchargées lorsqu'un client désire voir une page en contenant. Leur durée de vie varie: elle peut être très courte (certains ne survivent que le temps de leur exécution sur le client) ou relativement longue (les plug-ins de type Macromedia Flash). Le principal problème réside justement dans le fait que ces plug-ins doivent être téléchargés si nécessaire, ce qui augmente le temps d'attente d'affichage de la page web.

Le langage de programmation java propose deux approches différentes aux cas étudiés ci-dessus: en général, les applets permettent d'implémenter des applications de type *fat client*, tandis que les servlets sont utilisées pour les *thin clients*.

2.3.1. Tableau récapitulatif

	Graphiques basés client	Graphiques basés serveurs
Accès à la base de données	Multiplés. Une connexion est établie, puis les requêtes sont exécutées séquentiellement.	Un seul. La requête est exécutée, puis un graphique est généré du côté serveur.
Lien BD-Client	Une applet java ou un plug-in.	http s'occupe d'acheminer la requête jusqu'au serveur.
Coût en processeur	Moindre du côté serveur. Elevé du côté client, puisque celui-ci doit mettre en forme les graphiques.	Elevé du côté serveur: il doit générer des images. Moindre du côté client: il affiche une page web.

Encombrement du réseau	Relativement faible: ne transitent par le réseau que des requêtes de petite taille.	Assez élevé: une page web avec une image transite en retour de chaque nouvelle requête.
Mise en forme du graphique	Se fait du côté client: les graphiques sont dessinés par le browser lors de la réception du résultat des requêtes.	Se fait côté serveur: un CGI interroge la BD, produit un fichier image, l'insère dans un fichier HTML et envoie celui-ci au browser.
Initialisation et affichage des images	Assez lente dans le cas d'une applet java: celle-ci doit charger les pilotes JDBC et établir une connexion avec la base de données	Dépend de la taille de l'image et de la bande passante (<i>bandwidth</i>) disponible.
Temps d'exécution des requêtes.	Rapide. Celles-ci sont exécutées par une application développée et optimisée à cet effet.	Moins rapide. Les CGI ne sont pas prévus pour gérer les requêtes d'une base de données.

Figure 3: Tableau comparatif thin client/fat client

2.4. Choix d'une option

Tous les critères décrits et analysés dans le tableau récapitulatif ci-dessus n'ont pas la même importance dans le processus d'une solution acceptable: l'encombrement du réseau, par exemple, est un critère mineur dans notre cas puisque la quantité de données échangée entre le client et le serveur est relativement petite. La sécurité des données n'est pas non plus primordiale: les cours de la bourse sont des données accessibles gratuitement sur le web. On postule toutefois que les systèmes choisis (langage de programmation, drivers etc...) offrent à l'utilisateur une protection élémentaire contre tous types d'attaques qui résulteraient de l'emploi de l' applet java.

Il est par contre judicieux de privilégier le côté dynamique des graphiques générés du côté client. Le programme implémenté est une applet Java qui met sous forme graphique "à la volée" les données qu'elle reçoit du serveur web (et de la base de données). La relative lenteur d'initialisation des applets Java (qui provient du fait que le client doit interpréter ligne par ligne le code reçu du serveur) est compensée par la possibilité de pouvoir analyser presque en temps réel les évolutions de cours d'actions.

Le tableau qui suit évalue les options du tableau de la figure 3 selon une échelle qui va de 1 à 3 points (représentés par le symbole ♦)

	Graphiques basés client	Graphiques basés serveur	Commentaires
Accès BD	♦ ♦ ♦	♦	Les accès doivent être gérés comme un processus global et non comme une séquence de processus.
Lien BD-Client	♦ ♦	♦ ♦	
Coût en processeur	♦ ♦	♦	La tendance actuelle est plutôt de répartir les charges du côté client.
Encombrement réseau	♦ ♦ ♦	♦	Les requêtes sont de petite taille, contrairement aux images.
Mise en forme graphique	♦ ♦ ♦	♦ ♦	Le client met en forme l'image: ainsi on décharge le serveur de tâches.
Affichage images	♦	♦ ♦	L'initialisation est relativement lente du côté client.
Temps exécution requêtes	♦ ♦	♦	L'optimisation des temps de réponse est un critère important.
TOTAL	16/21	11/21	La solution "graphiques côté client" l'emporte nettement.

Figure 4: Evaluation des modèles thin client et fat client

3. Présentation de la méthode choisie

3.1. Choix techniques

Afin de satisfaire aux contraintes posées par la demande initiale (graphiques dynamiques, rapidité d'exécution et affichage semblable sur des systèmes différents), le

choix s'est porté sur la technologie Java qui allie sécurité dans le traitement et le transfert des données ainsi que portabilité sur les différents OS disponibles sur le marché. Voici un bref descriptif des environnements choisis pour l'implémentation du programme:

3.2. Java

L'implémentation du programme est faite dans le langage orienté-objet Java. Celui-ci est tout particulièrement destiné à produire des applets. Ces applets sont imbriquées dans une page web et elles ont des caractéristiques qui font leur popularité ([Flanagan 99]) :

- *sécurité*: une applet ne peut avoir d'accès direct aux fichiers locaux, ce qui permet d'exclure toute tentative d'intrusion dans les fichiers d'un tiers. En outre, l'applet n'est pas persistante: quand elle cesse d'être exécutée, elle se décharge automatiquement de la mémoire de l'ordinateur.
- *indépendance*: les applets ont été implémentées de telle sorte qu'elles sont indépendantes du hardware. Quand une page web est téléchargée, le code Java de l'applet est transmis en "byte code". Ce langage intermédiaire est ensuite traduit par un interpréteur en code machine et est exécuté par la JVM (Java Virtual Machine). Ce mécanisme assure la portabilité du langage Java ("*write once, deploy anywhere*").

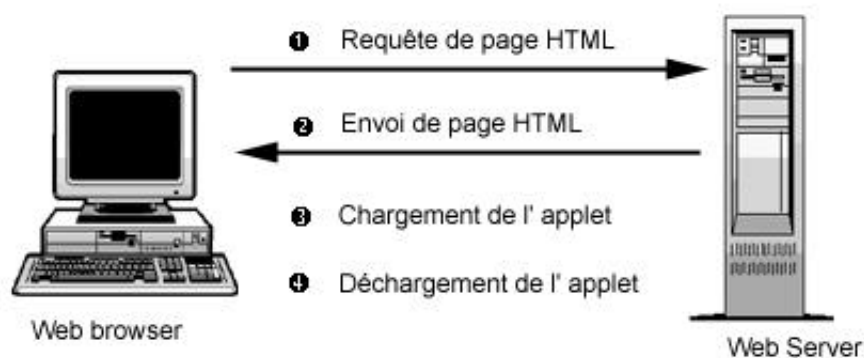


Figure 5: Chargement d'une applet java

3.3. JDBC

Java Database Connectivity (JDBC) a été développé spécialement pour faciliter l'accès aux bases de données. Dans sa version la plus évoluée (pilotes de type 3 et 4), JDBC est constitué de classes Java qui définissent un mécanisme standard pour accéder et

manipuler les données d'une base de données à l'aide du langage SQL. JDBC est un pas supplémentaire dans la tentative d'unification et de résolution des problèmes de communication entre différents systèmes de gestion de base de données. Toutefois, contrairement à l'environnement de développement du langage java (JDK, *java development kit*), JDBC n'est pas gratuit (sauf pour quelques bases de données non propriétaires). Il existe en effet une multitude de pilotes JDBC développés spécialement pour les plus grands logiciels de bases de données (Oracle, Informix, MS SQL Server) et ne fonctionnant qu'avec ceux-ci. Ceci provient du fait que ces logiciels ont chacun de leur côté développé des extensions au langage SQL, ce qui les rend dans certains cas incompatibles entre eux et qui nécessite des pilotes JDBC spécifiques.

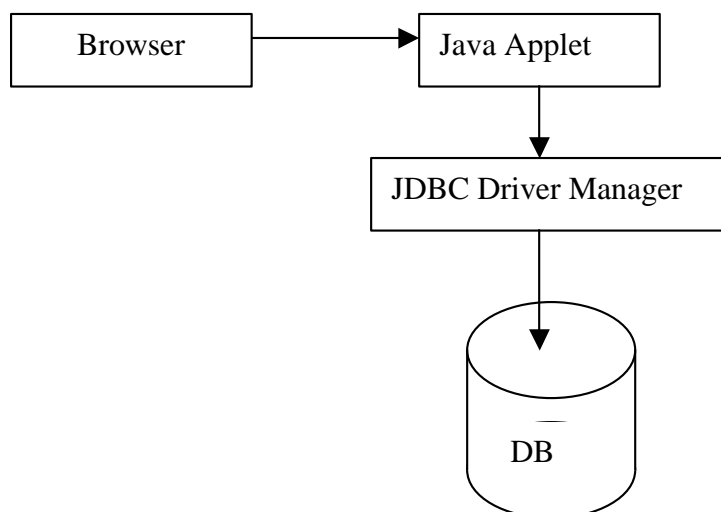


Figure 6: Mécanisme de fonctionnement de JDBC

4. Mode d'emploi de l'applet

L'exécution de l'applet se fait dans l'ordre suivant:

- a) L'applet est mise en mémoire lors du chargement de la page.
- b) L'applet initialise une connexion avec le serveur web de la BSU, puis fait une première requête afin d'afficher la liste des cours disponibles.

- c) Lorsque l'utilisateur sélectionne un cours, l'applet envoie une requête au serveur web, qui à son tour interroge la base de données et renvoie la réponse au client.
- d) Si le joueur quitte la page ou ferme l'applet, celle-ci est automatiquement déchargée de la mémoire. Cette façon de faire garantit à l'utilisateur la sécurité et l'intégrité des données de son ordinateur.

4.1. Afficher le développement du cours d'une action particulière

Jusqu'alors, le site web de la BSU (www.unifr.ch/bsu) ne donnait pas au joueur la possibilité d'avoir un aperçu sous forme graphique du cours d'une action. L'utilisateur devait alors aller sur un autre site web proposant ce genre de service. Cette applet comble cette lacune: le joueur peut analyser le cours d'une action pour une période qui va du début du jeu jusqu' au jour présent. Il a également la possibilité de voir la moyenne du cours l'intéressant. Les variations de cours d'une action sont indiquées sous forme cardinale ainsi qu'en pourcentage.

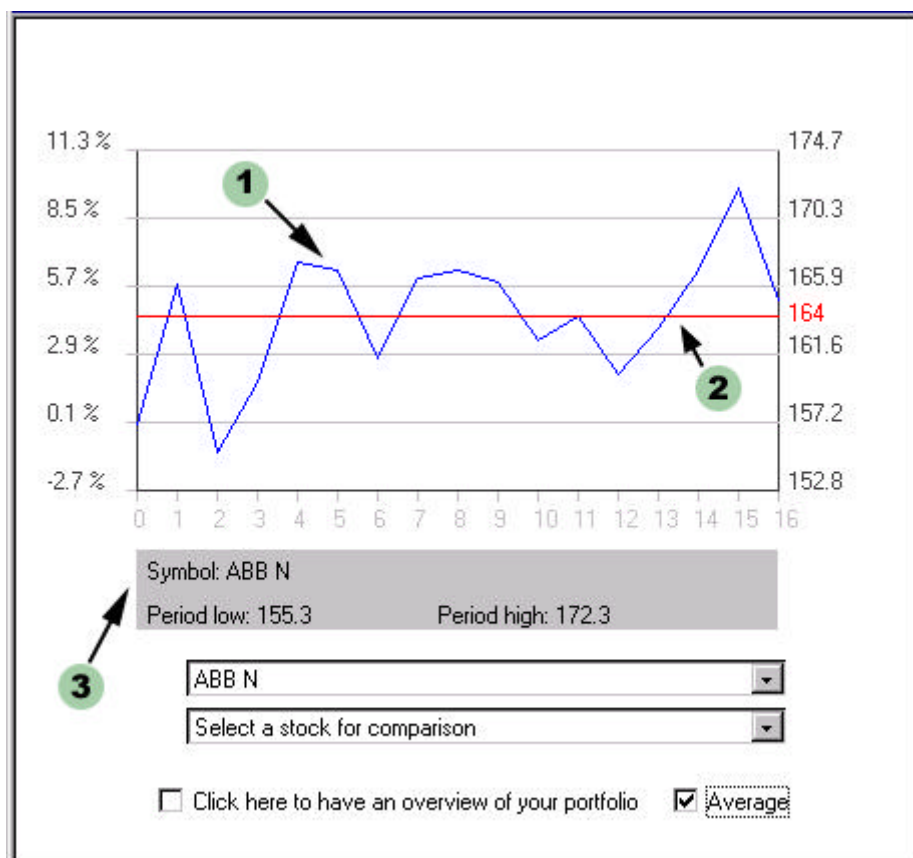


Figure 7: Sélection du cours d'une action

En bleu (*point 1*), le cours de l'action désirée (ABB N dans cet exemple)

En rouge (*point 2*), la moyenne de l'action sur les périodes sélectionnées. Cette option peut être désélectionnée si désiré.

Dans le rectangle grisé (*point 3*), le nom de l'action ainsi que les cours les plus hauts et les plus bas atteints durant la période de jeu.

4.2. Afficher le développement du portefeuille du joueur

Au début du jeu boursier, le participant reçoit un montant (fictif) qui lui permettra de faire des investissements dans des actions, des options ou des obligations de son choix. Ce montant de départ va, selon les capacités du joueur et les fluctuations du marché, augmenter ou descendre. L'applet Java permet ainsi d'avoir un aperçu sous forme de courbe de l'évolution d'un portefeuille. Dans le graphique ci-dessous, le joueur Stefan peut ainsi se rendre compte de la chute de son portefeuille (courbe en vert), ainsi que des minima et maxima atteints. Le joueur peut aussi avoir la moyenne de son portefeuille pour la période de jeu.

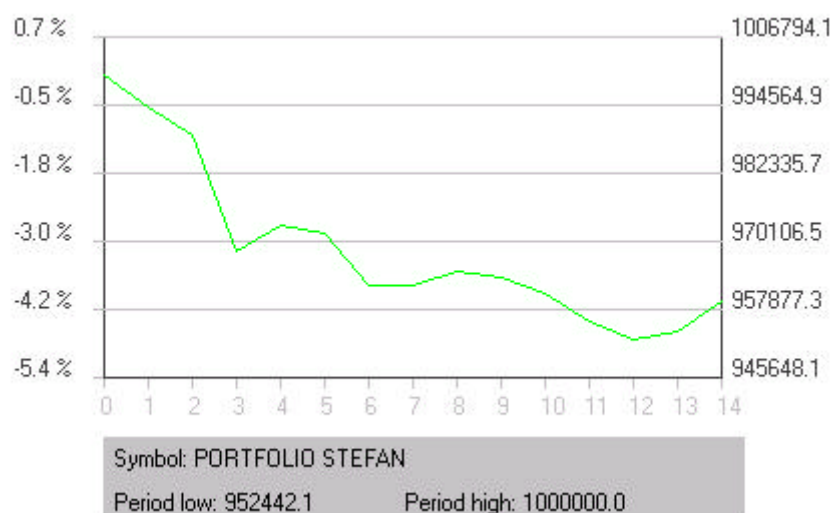


Figure 8: Affichage du portefeuille du joueur

4.3. Comparer les cours de deux actions

Avant de porter son choix sur telle ou telle action, il peut être judicieux, voire nécessaire, de procéder à une comparaison entre deux actions. Cette comparaison peut se faire entre deux actions de la même branche (dans l'exemple ci-dessous deux

compagnies du secteur bancaire) ou non. Ce système permet de se faire visuellement une idée plus précise des choix qui sont offerts.

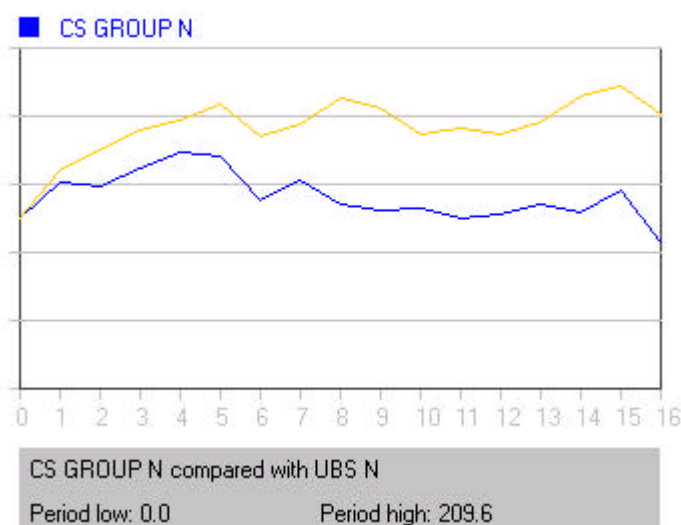


Figure 9: Comparaison de deux actions

5. Conclusion

Le thème de ce séminaire est dans la mouvance du temps: à l'heure actuelle les entreprises qui veulent réussir se rendent compte principalement de deux choses: premièrement, les bases de données acquièrent une valeur primordiale. Les données récoltées auprès des clients permettent de cibler les envies de ceux-ci de mieux en mieux et donc de répondre plus précisément à leur besoin. Deuxièmement, une entreprise se doit d'avoir une vitrine qui touche le plus grand nombre de clients potentiels, et c'est tout naturellement qu'elle se tourne vers le web.

Mais les entreprises qui réussiront dans leur domaine seront celles qui allieront le mieux ces deux composantes essentielles: des données pertinentes proposées de manière intelligente! Selon moi, toute solution jugée efficace se doit d'être dynamique, ciblée, attrayante et rapide. L'accès au web et l'offre d'outils de programmation et de publication en-ligne toujours plus étoffée et bon marché provoque une concurrence entre les entreprises qui doivent se démarquer entre elles par la qualité des services ou des produits offerts. C'est également la démarche de la BSU qui veut offrir à ses membres un nouvel outil d'analyse dynamique.

6. Références

[Flanagan 99] Flanagan, David: Java in a Nutshell, a desktop quick reference. O'Reilly, 3rd edition, Cambridge 1999.

[O'Donnell 99] O'Donnell Jim: HTML 4, XML and Java 1.2. QUE Editions, Los Angeles 1999.

[D-tec 98]: 3- and n- Tier Architectures. disponible: <http://www.d-tec.ch/e/3tier.html>, accédé 27 novembre 2000.

[TCC 00]: What is thin client computing? disponible: <http://www.thinclient.net/technology/history.htm>, accédé 12 décembre 2000.

[Fisher 99]: JDBC Database Access. disponible <http://java.sun.com/docs/books/tutorial/jdbc/index.html>, accédé 7 novembre 2000.

[Berry 97] Berry, Michael J.A.: DataMining Techniques. Wiley Computer Publishing, 1st edition, New York 1997.