Travail d'Etude et de Recherche

Université de Nice Sophia-Antipolis
Maîtrise Informatique

10 juin 2004

BN1

Travail sur les gestionnaires de contenu en ligne

Etudiants

Mario Di Miceli
Maxime Lamure
Damien Mandrioli
Romain Raugi

Encadrants

Michel Buffa, maître de conférence
Colas Nahaboo, WebCore, ILOG Sophia Antipolis

Table des matières

Introduction

Le sujet de ce TER consiste à ajouter à l'outil web coopératif TWiki un éditeur WYSIWYG online/offline et un outil de refactoring permettant le renommage, le déplacement et la suppression d'un ensemble de pages www de manière plus efficace que la méthode standard.

Ce TER est co-encadré par Michel Buffa et par M. Colas Nahaboo de la société ILOG. L'outil TWiki est au coeur de l'intranet du département informatique de l'UNSA et de la société ILOG, il est par ailleurs très utilisé dans le monde.

TWiki est un "outil web collaboratif" permettant à plusieurs personnes de créer/modifier des pages web, partager des documents attachés aux pages, etc... Un article résumant ses fonctionnalités et l'usage qui en est fait au département informatique est disponible dans le numéro 10 de la revue ISDM (http://www.isdm.org).

Le principal reproche que l'on fait à TWiki est la pauvreté de l'éditeur de pages www standard : une simple « text area » HTML dans un navigateur. L'avantage évident est que si un utilisateur voit une page www, il peut la modifier simplement en pressant le bouton « edit » et modifier son contenu au sein même de son navigateur www. Cependant, l'édition se fait un mode texte, et une syntaxe certes simple mais rebutante pour les non initiés est utilisée.

Un projet réalisé au premier semestre par des étudiants de l'IUP Miage a débouché sur la possibilité d'utiliser un éditeur WYSIWYG écrit en javascript en lieu et place de l'éditeur standard.

Un traducteur utilisant XML comme syntaxe intermédiaire a été développé, permettant la traduction de la syntaxe TWikiML vers HTML et vice-versa. L'idée consistant à ce que le contenu des pages twiki, qu'il soit édité par l'éditeur standard ou par un éditeur HTML quelconque, ne change pas radicalement. *Il est en effet fondamental que l'on puisse re-éditer une page à l'aide de l'éditeur standard, après qu'elle ait été modifiée par un éditeur HTML, sans qu'elle soit "polluée" par des tags HTML en surnombre*.

La solution développée par les élèves de la Miage fonctionne mais l'éditeur javascript a plusieurs défauts :

- Il ne permet pas une édition offline,
- Il s'agit d'un éditeur HTML et son interface graphique propose de nombreuses options qui n'existent pas en syntaxe TWiki. Par conséquent, le traducteur HTML vers TWikiML se trouve dans l'obligation soit d'insérer du HTML dans la page TWikiML finale, soit d'ignorer ces spécificités. En outre le code est complexe et difficile d'accès, donc très difficile à personnaliser en y apportant des options propres à TWiki, comme la complétion automatique des noms des liens locaux, etc...

Dans le cadre de ce projet nous avons réalisé un éditeur WYSIWYG écrit en java, se lançant automatiquement en cliquant sur un nouveau lien "edit with WYSIWYG editor". L'éditeur s'installe sur le disque dur de l'utilisateur via la technologie java Web Start (qui

assure que tout le monde a bien une version à jour) et communique avec le serveur TWiki via des web services que nous avons également développés. Cet éditeur propose trois vues distinctes et synchronisées : la vue WYSIWYG, la vue TWikiML et la vue HTML. La gui de l'éditeur de permet que de faire en WYSIWYG que ce que la syntaxe TWikiML peut supporter : gras, italique, tableaux, liens, etc...

Par ailleurs un outil de "refactoring" permettant d'organiser la structure globale du contenu du serveur TWiki a été développé, via les mêmes technologies : java web start et web services. Il permet beaucoup plus facilement que via l'interface web de déplacer/renommer/supprimer des pages ou des ensembles de pages.

Dans une première partie nous présentons l'architecture de TWIKI, ainsi que la manière dont nous avons implanté une couche de services web permettant de l'exploiter à distance.

Dans une seconde partie nous présentons l'éditeur WYSIWYG, et dans une troisième partie l'outil de refactoring est détaillé. Enfin, dans une dernière partie nous faisons le bilan de ce projet : adéquation avec le cahier des charges, état du logiciel livré, etc...

Finalement, dans la conclusion, nous tirons un bilan personnel de cette expérience.

Vous trouverez en annexe des détails sur les technologies utilisées ainsi que les ressources web que nous avons répertoriées...

Première partie : TWiki et les web services

Architecture de TWiki

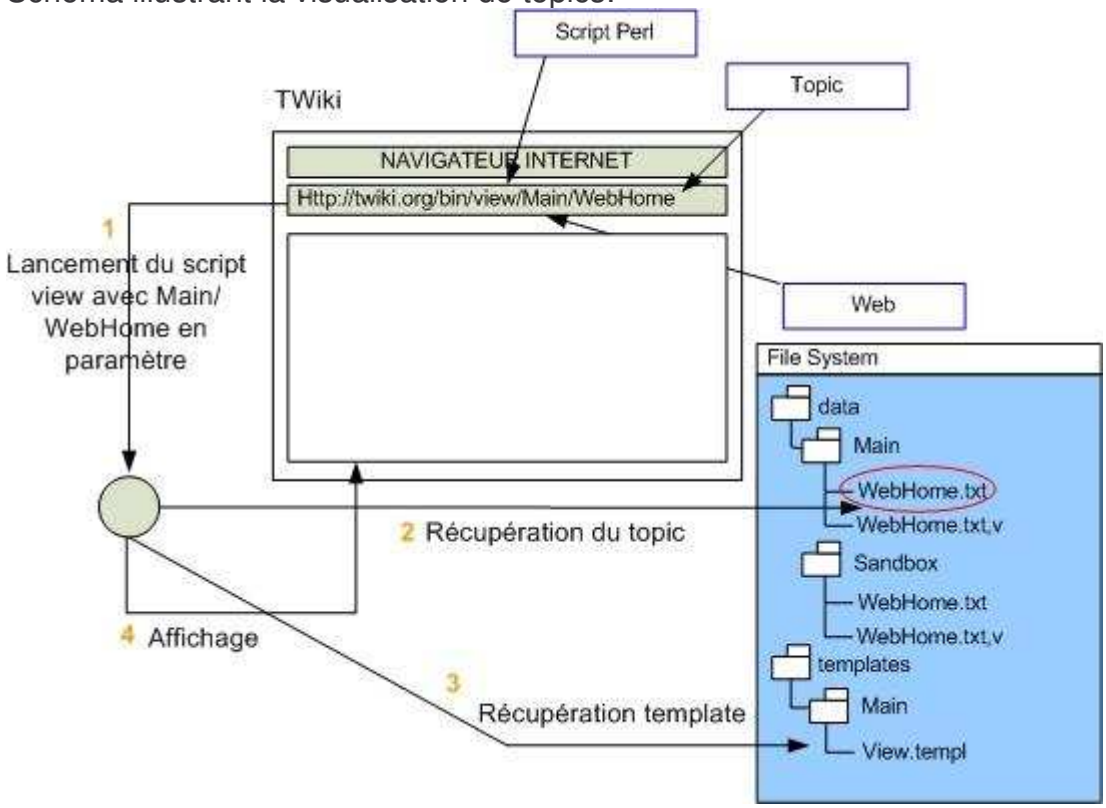
TWiki est avant tout un ensemble de scripts Perl permettant la gestion dynamique d'un ensemble de pages web. Par dynamique il faut comprendre que toutes les pages sont « calculées au vol » et non pas de simples fichiers HTML statiques. Il repose essentiellement sur des scripts CGI et des modules. Ces derniers constituent des bibliothèques de fonctions Perl réutilisables par les scripts mais aussi par d'éventuels plugins qui peuvent se greffer à TWiki. Ces modules en constituent le cœur. Les fonctions qui y sont implémentées gèrent la manipulation de topics et d'attachements, la gestion des autorisations, la gestion de RCS pour empiler les versions des topics et attachements ainsi que beaucoup d'autres toutes aussi nécessaires.

Les scripts CGI contiennent aussi du code Perl mais celui-ci n'est pas réutilisable. C'est le code exécuté directement lors de l'utilisation de TWiki. Par exemple, visualiser un topic consiste à faire appel au script « view » et l'éditer à faire appel au script « edit ».

Contrairement à un site web traditionnel où les scripts sont en quelque sorte les pages, ici ils travaillent (en général) sur les pages : les topics. Ces derniers constituent un autre fondement de TWiki. Ce sont les pages que l'on peut voir lorsqu'on utilise TWiki. Ils sont classés dans des dossiers appelés webs. Au niveau implémentation, les webs sont représentés par des dossiers et les topics par des fichiers textes. La mise en forme est plus ou moins exclue de l'information contenue dans les topics, ils ont pour objectif de contenir uniquement l'information. Leur présentation est faite par des « templates » (pour

être global, des fichiers HTML avec des macros). Voici un schéma illustrant un cas concret, la visualisation de topics:

Schéma illustrant la visualisation de topics:



Lors de la visualisation, on fait appel au script « view » avec en paramètre dans l'url la location du topic. Une fois lancé, « view » va aller récupérer le contenu texte du topic. Ici, le fichier va être cherché dans le dossier data/Main (le web). Une fois le contenu textuel récupéré, le script va le mettre en forme avec l'aide d'un template, qui peut être général ou spécifique à un web. La transformation réalisée sera affichée dans le navigateur. En fonction de leur rôle, les scripts fonctionnent plus ou moins selon ce système. Vous pouvez voir dans ce schéma des fichiers .txt,v accompagnant les topics : il s'agit des piles de versions RCS. En ce qui concerne les attachements, ils sont sauvegardés dans un autre répertoire « pub » dans une hiérarchie de type pub/web/topic/attachement_name. Etant donné qu'il ne peut y avoir deux topics de même nom dans un web, il n'y a pas de conflit.

Pour en revenir aux scripts, nous avons illustré avec le schéma précédent comment fonctionnait « view » uniquement. Il y en a évidemment d'autres parmi lesquels « edit » pour éditer des topics et qui fonctionne suivant le même modèle, « rename » pour le refactoring en général, « save » pour la sauvegarde de topics, « upload » pour l'upload d'attachements et « password » pour l'authentification.

Les web services

Définition

Un service Web est une technologie permettant à des applications de dialoguer à distance via Internet (par le protocole HTTP), et ceci indépendamment des plates-formes et des langages sur lesquelles elles reposent. Pour ce faire, les services Web s'appuient sur un ensemble de protocoles standardisant les modes d'invocation mutuels de composants applicatifs. On parle alors de WSDL, de message SOAP et d'annuaire UDDI :

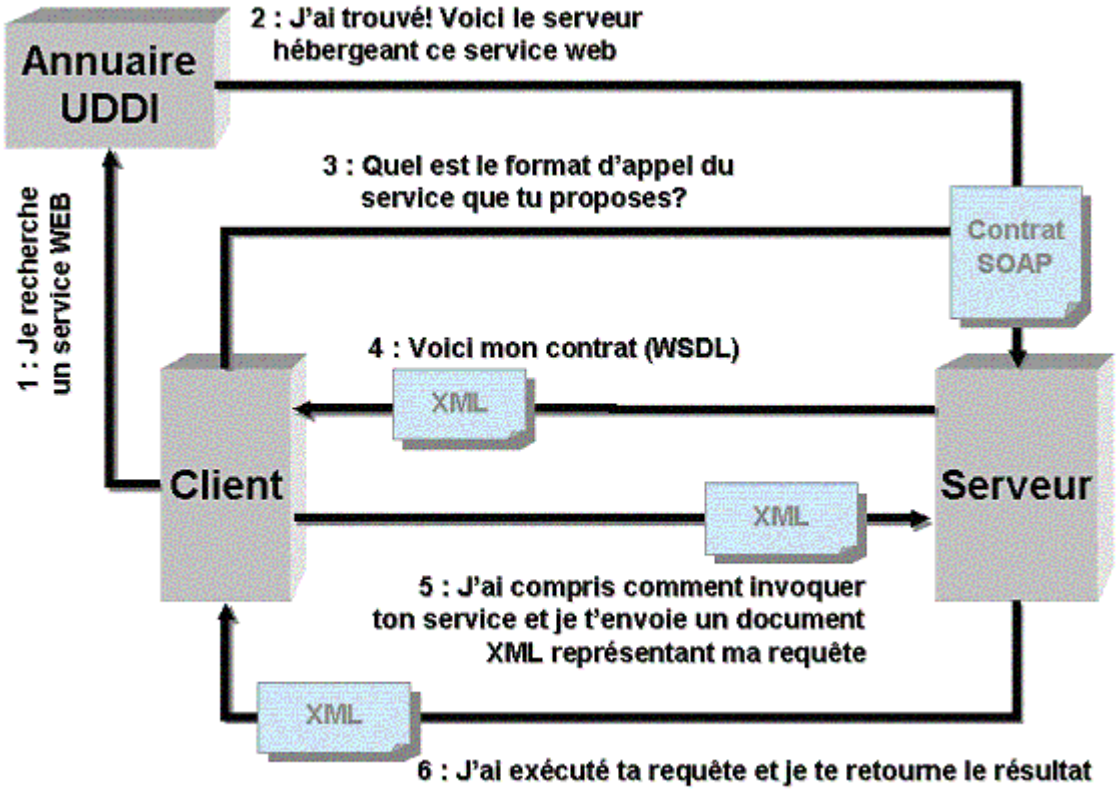
- La WSDL ou Web Service Description Language fournit un modèle et un format XML pour décrire les Web Services. Elle permet de décrire précisément les fonctionnalités offertes par le service, en nous indiquant, par exemple, comment et où les appeler.
- SOAP ou Simple Object Acces Protocol permet de définir les messages que l'on va pouvoir envoyer et recevoir du Service Web. Cette spécification encapsule des messages XML. C'est pour cette raison que les applications peuvent être écrite dans des langages différents.
- L'UDDI représente un annuaire électronique référençant les services Web. C'est par cet annuaire qu'on arrive connaître l'Id (ou adresse) des Services Web disponible qui va nous permettre de les appeler dans nos applications.

Fonctionnement

Principe général

- Un client recherche un service dans l'annuaire UDDI
- Il appelle le Service Web à partir de son application
- le serveur traite la demande et renvoie le résultat au client
- le client utilise le résultat

Le schéma ci-dessous illustre ce fonctionnement :



En ce qui concerne nos applications:

- Le client n'a pas besoin de rechercher les services dont il a besoin dans l'annuaire, puisqu'ils ont été développés pour lui.
- Le client récupère un stub, généré automatiquement avec WSDL2Java à partir de la WSDL.
- Le client peut appeler les services comme une fonction locale à partit du stub.

Interfaçage avec TWiki

TWiki est écrit en Perl, il était donc intéressant de développer nos services dans ce langage afin que nos applications s'intègrent mieux avec le serveur. Même si l'apprentissage de ce langage n'est pas intuitif, il nous a permis d'utiliser directement certaines fonctions de TWiki ce qui se traduit par un gain de temps de développement. L'interfaçage avec TWiki s'est fait par le biais de la toolkit SOAP::Lite et de l'application que nous avons développé qui est décrite ci-après ...

Application réalisée

Principe

Les services développés ne sont pas des fonctions réparties au sein du code source de TWiki mais font l'objet d'une application à part entière. Elle est constituée d'un ensemble de modules (principalement) et de scripts dont les fonctions utilisent elles-mêmes ce qui est réutilisable dans les modules de TWiki. Mais ce n'est pas tout.

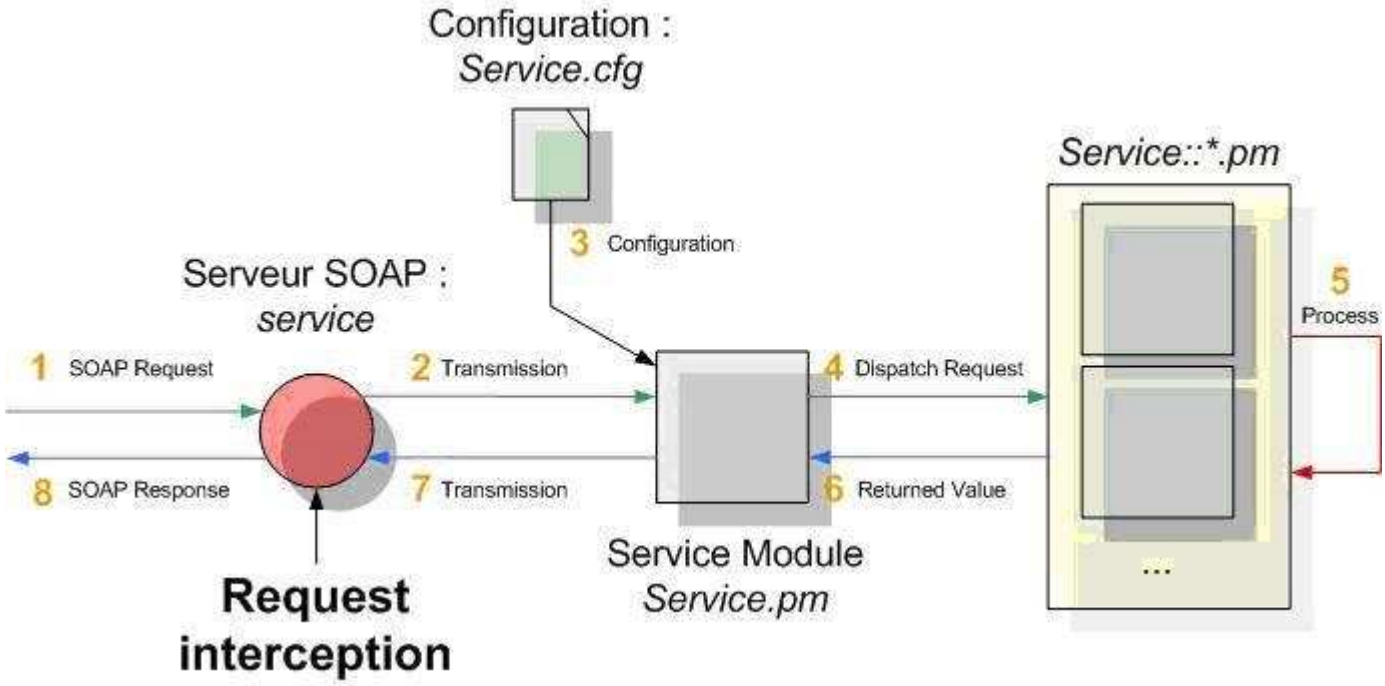
Cette partie du TER n'a pas consisté uniquement à faire appel à des fonctions TWiki mais à bâtir bel et bien une application afin que l'interfaçage entre TWiki et nos applications clientes soit réussi, en prenant en compte certains aspects non fonctionnels.

Cette API, bien que « pluguée » sur TWiki a été conçue de manière à pouvoir s'interfacer avec n'importe quel Wiki. Le plus gros du travail de conception a consisté à déterminer les services disponibles pour un client externe à TWiki.

Un système de connexions avec authentification, un mécanisme de gestion de verrous bâti au-dessus de celui de TWiki et un « verrou administratif » pour gérer la concurrence entre applications de *refactoring* ont été ainsi réalisés afin d'avoir des clients qui interfacent correctement avec TWiki. La "documentation pour développeurs" située en annexe 4 donne des détails concernant les choix d'implémentations et ce qui a été fait dans ce sens.

Du point de vue du serveur, l'application s'initialise lors de l'interception d'une requête et se termine une fois le traitement effectué; les requêtes pouvant être une demande d'opération de refactoring, de récupération de topic, de sauvegarde etc. Voici un schéma illustrant ce qu'il se passe entre ces deux instants clefs:

Schéma illustrant le cycle de vie de notre application:



Dans ce schéma, ce qui est décrit comme "process" (étape 5) concerne le corps de ce que sont les services web. C'est durant cette étape que les appels aux fonctions TWiki sont faits, en plus d'autres traitements que nous avons faits nous-mêmes. Le serveur SOAP a pour objectif d'écouter sur un port donné afin d'intercepter des messages SOAP. Une fois le message intercepté, il le désérialise et transmet la requête à un module Service. Ce module se charge d'initialiser la configuration du service en paramétrant des variables globales, de la même manière que TWiki. Une fois la configuration effectuée, le module répartit les requêtes dans des sous modules et renvoie le résultat (au serveur SOAP). Il agit en quelque sorte comme un skeleton de serveur. Le serveur lui-même retourne un message SOAP contenant la(les) valeur(s) de retour au client.

Services fonctionnels proposés

Dans cette partie sont présentés succinctement les services principaux proposés par notre application et qui ont été développés selon les besoins des clients. Ils sont le résultat d'études et de réflexion de la part de toute l'équipe du TER. Le manuel en annexe 4 décrit tous les services proposés de manière plus détaillée.

Pour le refactoring

A ce niveau, il y a deux types de services: les services d'accès aux données pour connaître les propriétés d'un topic, d'un web et les services de traitement.

- Envoi de données

Ces derniers permettent d'envoyer au client une représentation concrète de l'arbre correspondant à leurs requêtes.

- Opérations

Les opérations de refactoring réalisées avec l'outil doivent être validées sur le serveur TWiki. Cette « validation » fait l'objet de cinq services web qui correspondent aux opérations de refactoring proposées par l'outil:

    - renommage de topics
    - déplacement de topics
        - déplacement inter-webs
        - changement de topic parent
    - suppression de topics
    - fusion de topics
    - copie de topics

Ces services proposent plusieurs options, notamment pour certains la gestion des attachements et la mise à jour des liens. Ils constituent le coeur de ce qui a été fait dans le refactoring; l'outil Java en est l'interface graphique.

Pour l'édition WYSIWYG

Au niveau WYSIWYG, deux services majeurs "symétriques" ont été développés. L'un permet d'envoyer le contenu d'un topic à l'éditeur WYSIWYG, l'autre de sauvegarder le résultat de l'édition sur TWiki.

- Envoi de données

Les services permettent de fournir à l'éditeur WYSIWYG les données telles quelles sont décrites sur le serveur. De plus, certains services permettent de gérer les attachements.

- Sauvegarde

Un service a été développé afin de permettre à l'éditeur WYSIWYG de sauvegarder un topic sur TWiki. L'upload d'attachements est aussi géré.

Déroulement

Méthodologie

La première étape d'un développement d'un service Web est de demander au client les services dont il a besoin, les fonctions qui lui permettront d'accéder aux données. Cette première phase, est donc une phase de dialogue, de réflexion entre les membres de l'équipe afin de bien cerner les besoins, mais aussi les contraintes de chacun.
La phase de développement commence par une période d'analyse de TWiki. Il s'agit en fait de mieux comprendre son fonctionnement afin d'interagir avec lui. Cette analyse est portée sur les packages que nous avons pu utiliser afin de s'intégrer parfaitement au serveur, mais cela nous permettait aussi un gain de temps considérable au niveau du développement.
Pour finir, afin d'exporter nos fonctions, nous devons décrire les services dans la WSDL. A partir de ce fichier, nous allons générer des stubs afin que les applications clientes puissent utiliser nos fonctions.
Lors du développement, nous testions nos programmes sur notre TWiki, installé en local sur nos machines. Une fois l'application stable, nous l'étendions sur le TWiki de test de la miage de Nice afin que ceux qui développent les applications clientes puissent interagir avec les services Web. Cela nous permettait également de tester nos services avec les contraintes d'un véritable serveur. Dans cette partie, sont détaillées les taches les plus importantes qui ont participé à la réalisation de cette application.

Prototype service

- Effectifs

1 personne, 1 semaine et demi de travail, estimation : 3h/jour en moyenne.

- Choix
    - Développer un prototype du service avant le début du TER

Etant donné l'ambition du projet, les risques auxquels nous devions faire face et le peu de connaissances que nous avions concernant les technologies à utiliser, un prototype du service a été développé avant le début du TER.

    - Ecrire une description WSDL dès le départ

La description WSDL fut envisagée dès le départ pour jouer le rôle de contrat avec les clients des services sur les structures de données et les signatures des méthodes. Ce ne fut qu'utopique étant donné qu'elle a évolué en permanence durant ce projet. Par contre, elle nous a servi à générer les stubs automatiquement pour l'éditeur et l'outil de refactoring Java, chose que nous n'envisagions pas lors de la rédaction du cahier des charges.

    - Etablir un système de connexion à l'application

Cette décision fut prise peu après la rédaction du cahier des charges. Ce système a pour ambition d'améliorer l'interfaçage entre nos outils (et peut-être d'autres à l'avenir) et TWiki sur des aspects non fonctionnels tels que la concurrence et la gestion des verrous.

- Faire nous-mêmes l'authentification

Nous pouvions nous baser sur l'authentification d'Apache ou la faire nous-même. Ayant voulu faire un service « général » et pas spécifiquement dédié aux applications lancées par Web Start par exemple, nous avons opté pour la deuxième solution.

- Bâtir un système de gestion de verrous au-dessus de celui de TWiki

Cela a été fait peu après la réalisation du prototype. Ne trouvant pas la gestion des verrous suffisamment précise pour nos services Web, il fut développé un système au-dessus n'altérant en rien le bon fonctionnement de TWiki. Le manuel en annexe 4 donne plus de précisions à ce sujet. Ceci n'était pas prévu initialement.

- Proposer une trace d'exécution

Le développement d'une application serveur s'étant soldé par un récent échec (pour l'un d'entre nous), un module pour faire des traces d'exécution a de suite été développé.

Envoi de la représentation arborescente des topics à l'outil de refactoring

- Effectifs

1 personne, 2 semaines, environ 8h/jour. Il s'agissait au départ de fournit un premier service pour tester la communication puis de le modifier selon les besoins des applications clientes.

- Difficultés
  - Performance

Comme son nom l'indique, un Service Web utilise le Web (protocole HTTP) pour transmettre ses messages (appels de fonctions, réception de résultat). L'inconvénient majeur dans notre cas est le problème de performance, même sur des connexions hauts débits. A l'origine, il fallait faire un appel au service pour chaque topic ce qui rendait l'application pratiquement inutilisable. Pour palier à cela, nous avons essayé de faire une représentation arborescente des topics au niveau serveur et de la transmettre au client. Ce dernier fait donc moins d'appels au serveur, réalise moins de traitement et son rôle est donc de fournir une représentation graphique des données, ce qui améliore les performances.

Opérations de refactoring

- Effectifs

1 personne, 2 semaines, environ 8h/jour. Cela comprend les analyses et la mise en place de systèmes qui leur sont liés tels le "verrou administratif".

- Difficultés

  - Analyser les scripts et modules de TWiki

N'ayant pas trouvé de tutorial expliquant TWiki dans les moindres détails, il fut nécessaire de se baser sur son code source pour comprendre. Des traces d'exécution

furent ainsi utilisées, ainsi que des modifications pour voir son comportement. Ce n'était pas réellement une difficulté, TWiki était suffisamment clair dans son code. Au final, la réalisation des opérations de refactoring s'est très bien passée, beaucoup plus que ce à quoi nous nous attendions lors de la rédaction du planning initial. Les analyses faites y sont aussi pour beaucoup.

    - Copie d'attachements pour la fusion et la copie

Toutes les opérations concernant les attachements se basent sur le déplacement. Pour faire la copie, il a fallu reprendre ce qu'il y avait de plus « bas » dans TWiki, les opérations au niveau fichier et répertoire. Il ne fut pas nécessaire de descendre aussi bas pour les autres opérations, TWiki proposant des opérations de haut niveau parfaitement adaptées.

Rapatriement de topics vers l'éditeur

- Effectifs

1 personne, 2 semaines, environ 8h/jour. Les fonctions de ces services sont de rechercher les informations nécessaire et de les formater afin qu'elles soient directement intégrable dans l'application.

- Difficultés
    - Passage d'accent

Pour faire transiter nos messages à travers le protocole HHTP, les services web utilisent le XML. Pour générer nos messages, nous utilisions le package SOAP ::Lite. Or ce denier utilise l'encodage UTF8 , ce qui ne permet pas le passage de caractère accentué. Pour palier à ce problème, il est possible dépasser le codage en iso-8859-1 mais il faudrait ajouter et modifier un certain nombre de package et le temps qu'il nous restait ne nous le permettait pas. Cependant, pour permettre la transmission de données « accentuées » nous avons développé des fonctions pour remplacer tout caractères spéciaux par leur représentation unicode. (é -> é)

Sauvegarde de topics

- Effectifs

1 personne, 3 jours, environ 8h/jour.

- Choix

    - Etablir un protocole de communication autour des attachements

Un protocole de communication autour des attachements a été établi afin que l'éditeur WYSIWYG puisse les gérer simplement avec les services relatifs aux topics. Ce protocole est détaillé en annexe 4 dans la section consacrée à la sauvegarde de topics.

- Difficultés

    - Upload d'attachements et données binaires

L’upload d’attachements sur TWiki repose en partie sur Apache. En ce qui concerne le service de sauvegarde de topics, il attend les données binaires des attachements dans le message. La difficulté résidait d'une part à comprendre comment les attachements étaient gérés par TWiki après l'upload et d'autre part à convertir des données binaires sauvegardées dans un texte en données pour un fichier binaire. Pour ce dernier, le problème a été vite résolu en cherchant dans les forums de discussions (qui ont constituées une grande partie des sources de recherche de ce TER).

- Evolutions

La sauvegarde doit sans doute être adaptée afin de pouvoir sauvegarder de nouveaux topics. Un ou deux jours supplémentaires auraient été suffisants.

Notifications

- Effectifs

1 personne, 3 jours, environ 8h/jour.

- Choix

Ce système est un choix en lui-même et n'était pas prévu lors de la rédaction du cahier des charges. Les notifications sont gérées uniquement au sein du service, pas dans TWiki en général car cela aurait nécessité de modifier son code (donc de faire un patch). Dans ce sens, ce système n’est pas terminé mais s’avère être un projet ouvert à d’autres développeurs.

- Evolutions

Il faudrait généraliser les notifications à TWiki (et donc faire un patch) et intégrer les fonctionnalités à l'outil de refactoring. Deux semaines semblent nécessaires.

Deuxième partie : l'éditeur WYSIWYG standalone

La solution Java Web Start

Java Web Start est une technologie qui permet de lancer une application Java (pas une applet) simplement en cliquant sur un lien dans une page HTML. Le mécanisme de téléchargement et d'installation est entièrement automatisé, cela permet de véritablement installer et exécuter une application en un seul clic.

Contrairement aux applets, le téléchargement n'est pas systématique : si l'utilisateur à déjà lancé l'application et si l'application n'a pas changé sur le serveur, les classes java en cache localement sont utilisées.

Pour déployer une application avec cette technologie, il faut créer une archive jar signée et exécutable contenant l'application ainsi qu'un descripteur de déploiement qui porte pour extension JNLP (Java Network Launching Protocol). Ci dessous un exemple de descripteur JNLP:

```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+" codebase="http://miageprojet.unice.fr/twikitestnice/"> (Définition du codebase de l'application : l'emplacement ou télécharger le jar)
  <information>
    <title>TWiki WYSIWYG Editor</title>
    <vendor>* * * *</vendor>
    <homepage href="http://miageprojet.unice.fr/twikitestnice/view/%WEB%/%TOPIC%"/>
  </information>
<security>
    <all-permissions/> (Demande tous les droits sur la machine cliente, l'application doit être signée)
  </security>
<resources>
    <j2se version="1.3+" />
    <jar href="twe.jar" /> (la référence au fichier de l'application)
  </resources>
<offline-allowed/>
  <application-desc main-class="twe.gui.MainFrame"> (Classe principale de l'application)
  <argument>%ENDPOINT%</argument> (Arguments passé à l'application)
  <argument>%WEB%.%TOPIC%</argument>
  </application-desc>
</jnlp>
```

Choix de conception

L'éditeur proposé est avant tout un éditeur de contenu. Il a pour but d'aider l'utilisateur à composer ses documents TWiki à l'aide de la souris et d'un rendu visuel mais ce rendu ne correspond pas à ce qui sera affiché sur le site une fois l'édition terminée. *Ainsi, seules les commandes nécessaires à l'édition d'un document TWiki sont disponibles.* On ne peux pas par exemple centrer un paragraphe ce qui est possible en HTML mais pas en syntaxe TWiki.

L'éditeur permet de composer des documents *offline*. Ceci peut s'avérer pratique pour un utilisateur qui n'aurait pas toujours accès à Internet.

Concernant la stratégie de développement avec l'API Java, nous avons choisis d'utiliser les composants Swing tels quels, sans créer de classes héritant de ces derniers. Ce choix est dans certains cas une limite mais il s'est imposé du fait du manque de temps et de *main d'oeuvre* (une seule personne).

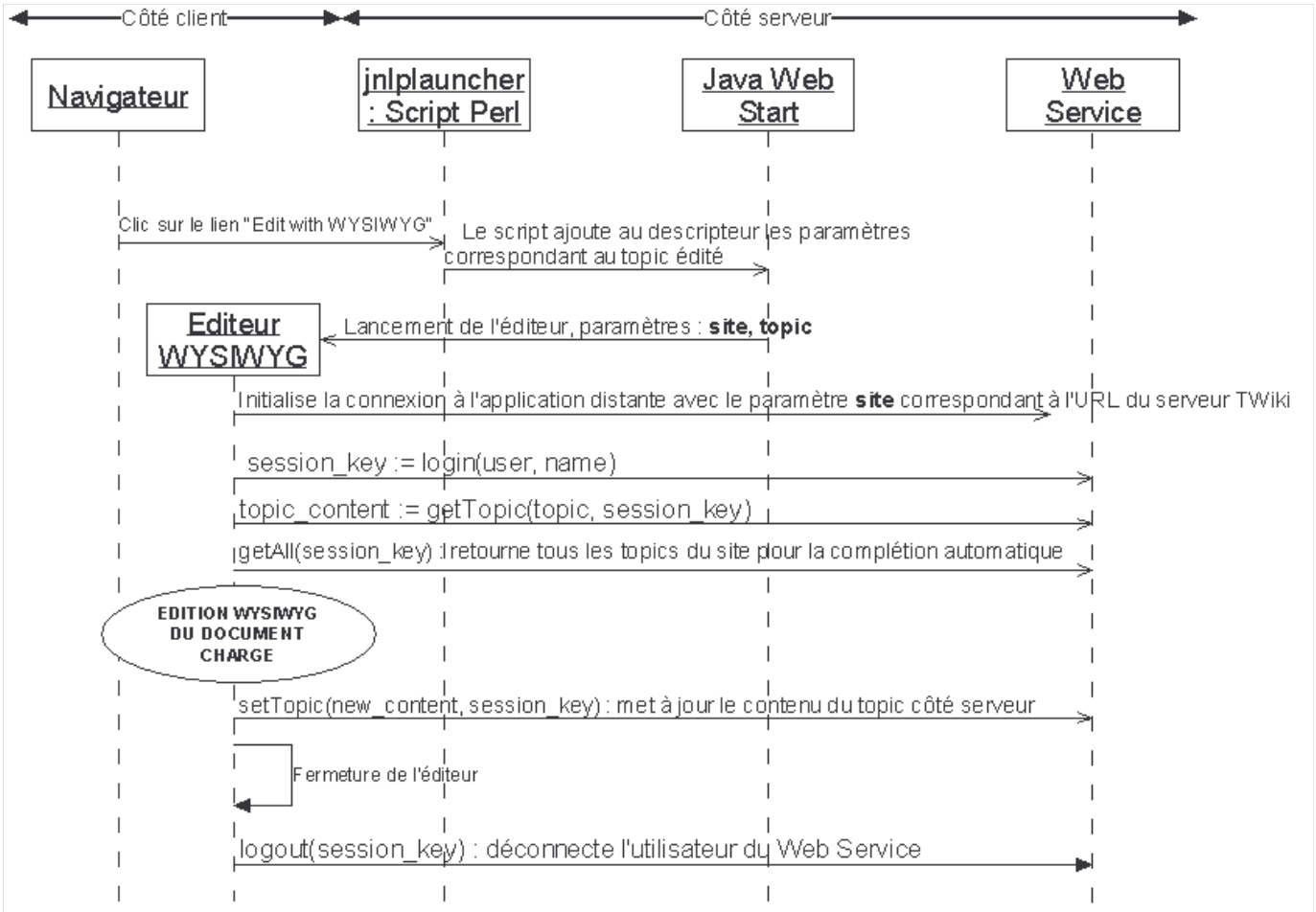
Le traducteur utilisé n'a pas été modifié, certaines imperfections de l'éditeur proviennent de certaines lacunes du traducteur.

Deux *composants* Swing sous licence GNU ont été utilisés :

- La fenêtre de dialogue pour l'authentification
- La fenêtre de dialogue pour la recherche - suppression

Communication avec « TWiki »

Ci dessous, un diagramme de séquence représentant une session d'édition du début à la fin :



NB : Les appels à getTopic() et getAll() sont réalisés en parallèle.

Fonctionnalités de l'éditeur

On trouvera en Annexe 2 : *Manuel utilisateur de l'éditeur wysiwyg* la liste des fonctionnalités de l'éditeur ainsi qu'un exemple d'utilisation.

Troisième partie : l'outil de refactoring

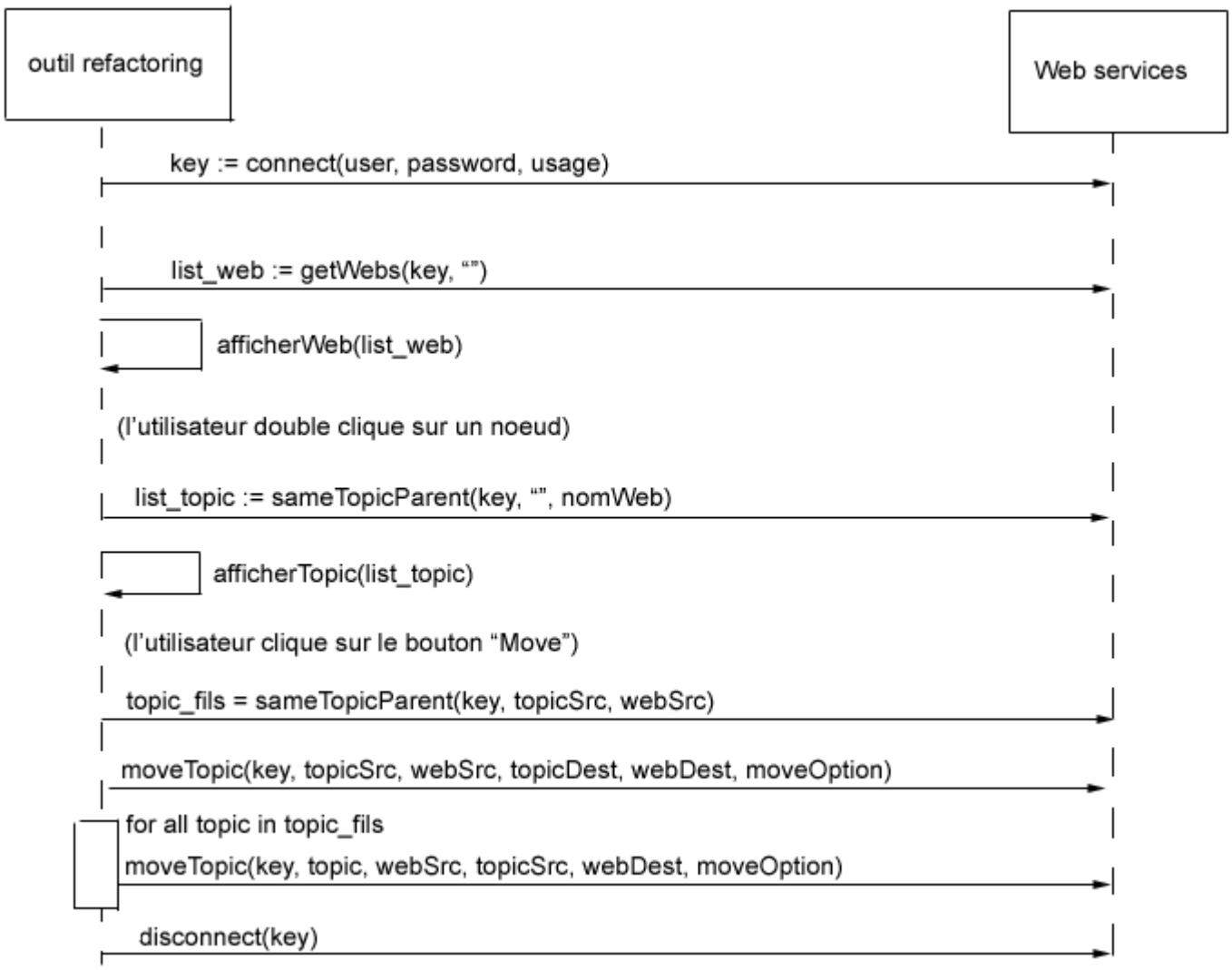
Le refactoring c'est quoi ?

Le refactoring est une méthode qui consiste à modifier la structure d'une arborescence de fichiers, en appliquant diverses actions comme le renommage, le déplacement et la suppression de fichiers par exemple. De nombreux environnements de programmation professionnels, comme JBuilder, Eclipse ou Dreamweaver, possèdent un outil de refactoring. Nous avons donc développé un outil, s'inspirant de celui de JBuilder, mais adapté à TWiki. Il permet le renommage, la suppression, la copie, le déplacement et la fusion d'un ou de plusieurs topics de l'arborescence de TWiki et effectue les modifications adéquates dans les fichiers concernés, grâce aux Web services.

Fonctionnement de l'outil

L'outil de refactoring est exécuté à partir d'une page web en cliquant sur le lien "Refactoring tool", grâce à la technologie Java Web Start décrite précédemment. Une boîte de dialogue d'authentification s'affiche puis, une fois que l'utilisateur a saisi ses paramètres de connexion, la fenêtre principale apparaît. On peut voir sur celle-ci deux arborescences et des boutons d'action (renommage, ...). Pour ne pas surcharger les performances, l'arborescence n'est pas chargée entièrement. En effet les webs sont chargés dans un premier temps, puis au fur et à mesure que l'on déploie les noeuds des arbres, les niveaux suivant de l'arborescence sont téléchargés (les noms de topic et de web) puis stockés en mémoire afin d'éviter de refaire des appels au serveur. De la même façon, lorsque l'on clique sur un élément de l'arbre de gauche, les propriétés correspondantes sont également téléchargées puis stockées en mémoire sous forme d'objet (Topic ou Web), de telle sorte qu'un si on reclique sur l'élément les propriétés sont affichées instantanément sans refaire appel au serveur.

Le schéma suivant montre plus en détail, à travers un diagramme de séquence simplifié, les communications entre l'application et les web services lorsque l'utilisateur déplace un topic



La description détaillée des fonctionnalités et de l'utilisation de l'application se situe en Annexe 3 dans le guide de l'utilisateur.

Difficultés rencontrées

Les principales difficultés que nous avons rencontrées pendant le développement de l'outil de refactoring, sont liées aux performances.

Pour éviter les problèmes de performances dus au chargement de l'arborescence, nous avons décidé de ne charger l'arborescence qu'au fur et à mesure que les nœuds des arbres étaient déployés.

Malheureusement nous étions trop optimistes puisque lors de l'intégration des appels de fonctions au serveur dans l'application, nous nous sommes aperçu que les performances étaient très mauvaises et que l'application était inutilisable. En effet, lors du déploiement d'un nœud de l'arbre, nous avons besoin de savoir si un topic est parent ou non pour pouvoir afficher un nœud ou une feuille. Il faut donc explorer un niveau de plus de l'arborescence, c'est-à-dire appeler une fonction du serveur pour chaque topic du nœud afin de savoir s'il a un fils.

Sachant qu'un appel de fonction prend une à deux secondes et qu'un topic peut avoir plusieurs centaines de fils, cette méthode n'était pas adaptée. La première solution que

nous avons envisagée, était de réduire le nombre d'appels au serveur. Pour cela la fonction qui renvoie la liste des topics fils pour un topic donné a été modifiée : si un topic possède un ou plusieurs fils, son nom est suffixé par le caractère '#'. Cette solution permet de ne faire qu'un appel de fonction pour déployer un nœud de l'arborescence.

Cette fois-ci, c'est coté serveur que les performances ont chuté. Le problème fut résolu en optimisant les fonctions coté serveur en s'appuyant sur la fonction « grep » pour les accès aux fichiers.

A la suite du temps perdu pour trouver ces solutions, nous avons pris du retard par rapport à nos objectifs. Nous avons donc choisi d'abandonner certaines fonctionnalités qui ne nous semblaient pas prioritaire comme la mise en évidence des cycles, des derniers topics créés et de la taille des topics.

Quatrième partie : bilan du projet

Communication :

Pourquoi communiquer

Tout au long de notre projet, nous avons pris soin de nous tenir au courant au sein de l'équipe du TER (étudiants et encadrants) afin que chacun puisse se faire une opinion de l'état d'avancement du projet. Maintenant que celui-ci est fini, on peut se demander si cela n'a pas été une perte de temps.

On peut également se demander quelle a été l'importance de cette communication dans l'aboutissement du projet. Nous verrons dans la description de la méthode de travail que pratiquement tout le monde a travaillé en parallèle. Il est donc indispensable de savoir ce que fait son équipe afin de juger si notre travail va bien correspondre à ce qui est attendu et poursuivre notre développement dans la même direction.

Pour illustrer cela, imaginez que vous deviez faire une voie ferrée avec un train, une personne s'occupe des rails et l'autre du train. Chacun fait son travail indépendamment, mais il faut faire attention que à ce que la largeur des rails corresponde a celle des roues et inversement, sinon le train ne pourra pas rouler alors que chacun à fait parfaitement ce qui lui était demandé. *Il est donc évident que la communication a contribué à l'aboutissement de notre projet.*

Méthodes utilisées

Communiquer est une chose, mais la façon de le faire en est une autre. En effet, il ne s'agit pas de déranger tout le monde dès que l'on rencontre un problème ou dès que l'on désire faire une requête sur tel ou tel point. Il est fort probable qu'à ce moment là, les membres de l'équipe soient en plein travail et ne soit pas complètement à votre disposition.

Pour palier à cela, nous avons donc utilisé le système de trace, qui consistait à décrire précisément ce que l'ont voulait faire communiquer. En fonction de l'organisation respective de chacun, on pouvait prendre connaissance des messages des autres, d'y répondre voire même d'organiser des réunions par chat afin de rassembler tout le monde.

De plus, à chaque étape de développement, nous contactions nos encadrants afin de leur montrer notre travail. Ils nous donnaient alors leur avis afin que nous poursuivions dans la bonne direction.

Outils et moyens mis à dispositions

Pour communiquer, nous avons utilisé plusieurs logiciels qui se sont avérés très utiles. Tout d'abord, nous avons utilisé l'outil TWiki pour tout le suivi écrit (travail effectué, suivi de bug,...).

Pour communiquer directement, nous utilisions MSN Messenger, et Skype pour les chats vocaux. Nos travaux étaient publiés sur un serveur CVS.

Enfin, lorsque c'était nécessaire, nous nous déplacions à Sophia-Antipolis (ILOG) afin de faire une démonstration de notre projet.

Méthodologie :

Organisation du travail

Notre projet comprend deux applications Java indépendantes et une application serveur bâtie autour des services Web pour leur permettre de communiquer avec TWiki. Nous avons décidé de faire développer cette dernière par deux personnes, et chaque application Java par une seule. Cela peut surprendre car les applications semblent demander beaucoup plus en terme de développement. La raison est simple, d'une part, le développement des services nécessite l'apprentissage de technologies, d'environnement et d'outils auxquels nous ne sommes pas familiers (Perl, spécification WSDL, TWiki etc.).

D'autre part, il fallait fournir ces services le plus rapidement possible afin de tester la communication des applications avec le serveur. Enfin, ces services se devaient d'être le plus optimal possible et il était indispensable qu'ils soient opérationnels afin que notre projet aboutisse (à quoi sert de faire un train si on n'a pas les rails ?).

Mise en place du développement

Au début du projet, des conflits engendrèrent quelques tensions du côté de l'équipe des services web. Mais très vite, un consensus su être trouvé pour ne pas mettre en péril la suite du développement. Le reste du temps, le développement se passa très bien, grâce notamment au suivi personnel et aux nombreux échanges d'opinions.

D'autre part, nous pouvons constater que le planning été relativement bien respecté pour les applications clientes, mais il est vrai qu'au niveau Web Service il y a eu un décalage, notamment du à l'apprentissage des technologies. Cependant, ce retard a été rapidement rattrapé une fois les notions acquises.

Aboutissement

Le fait de développer dans ces conditions nous a permis de ne pas bloquer et d'avancer à un rythme régulier. Le résultat s'en fait ressentir, on est arrivé au bout de nos objectifs en respectant le temps qu'il nous était donné. De plus, on peut dire que ce projet est le fruit d'un travail d'équipe dans lequel, la contribution de chacun était indispensable.

Evolution

Etat actuel

Le projet livré marche, et a été montré à nos encadrants. Le résultat les satisfait pleinement. On peut donc dire qu'au niveau développement, c'est une réussite. Il est vrai qu'il reste certaines améliorations à faire, mais nous préférons passer du temps sur l'analyse des différentes méthodes à mettre en œuvre, plutôt que de se plonger dans du code, sans être sûr d'aboutir.

Notre encadreur chez ILOG nous a proposé de continuer le projet en stage, ce qu'e vont faire deux d'entre nous.

Améliorations

- Gestions des accents au niveau Web Service

Changer l'encodage par défaut de SOAP::Lite de UTF8 en iso-8859-1

- Gestion des attachements du côté de l'éditeur et des services

Passage des données binaires entre le Web Service et l'éditeur

- Généralisation des notifications à TWiki et intégration à l'outil de refactoring
- Amélioration de la robustesse de l'éditeur

Gérer l'état des boutons d'actions en fonction de la position du curseur

- Lancement de l'éditeur à partir de l'outil de refactoring
- Ajout d'une barre de statut permettant de visualiser les communications avec les Web Services
- Amélioration de la fiabilité de l'outil de refactoring

Poursuites

L'analyse faite à la fin va permettre de poursuivre et d'estimer avec plus de facilités les améliorations qui restent à faire. Cette poursuite va être entamée par deux des membres du TER (Damien et Romain) qui vont travailler dessus *lors d'un stage chez ILOG*.

De plus, l'ensemble de notre projet est suivi par la communauté TWiki et les sources sont disponibles librement sur le CVS.

Conclusion

M. Buffa et M. Nahaboo ont été présents tout au long du projet, ils nous ont guidé dans nos choix, nous ont montré comment se déroulait un projet de cette envergure et ont répondu à nos questions. Mais de notre côté, il fallait aussi leur montrer qu'on savait ce que l'on faisait, il fallait les rassurer.

Ce TER nous a donc beaucoup appris, que ce soit au niveau du travail en groupe, dans l'apprentissage de nouvelles technologies, mais aussi par le simple fait d'avoir rencontré et travaillé avec M. Nahaboo qui nous a fait bénéficié de son expérience.

Notre projet est fonctionnel, et facilement intégrable dans un TWiki. On espère qu'il sera massivement utilisé, notamment par notre faculté. Nous sommes assez fière de ce que nous avons réalisé.

Une chose importante, est que notre projet correspond aux attentes formulées au départ. Or, dans le monde de l'entreprise, il est difficile de bien comprendre, cerner ce que souhaite le client et ainsi réaliser la bonne application.

Concernant l'apport de l'enseignement reçu à l'Université, il a été déterminant. Par exemple, dans la réalisation des applications clientes en Java, les nombreux projets que nous avons réalisés au cours des deux dernières années nous ont fourni l'expérience nécessaire à la réussite.

Annexe 0 : Manuel d'installation

Pré requis

TWiki (version du 01 Février 2003)

Nous avons développé nos services sur cette version de TWiki. Nous pouvons difficilement prédire leurs comportements sur de futures versions.

Perl 5.6+

Les services Web ont été développés et principalement testés avec la version 5.8 de Perl. Aucune fonctionnalité spécifique à cette version n?a été utilisée et nous n?avons pas constaté de problème avec la version 5.6.

Librairies Perl

SOAP::Lite 0.55+

SOAP::Lite est requis pour pouvoir utiliser les services Web nécessaires aux applications de refactoring et d?édition WYSWIYG.

XML::TreeBuilder 3.08+ (optionnel)

Cette librairie n'est requise que si le service de notifications est activé.

Ressources

Service

- Hierarchie de fichiers pour services:

Le répertoire *bin* contient des scripts Perl :

- *jnlplauncher* : un script qui génère des descripteurs Web Start JNLP à la volée.
- *service* : un serveur SOAP de type script CGI.

Le répertoire *lib* contient les modules Perl requis par les services Web ainsi qu?un fichier de configuration :

- *Service.cfg* : le fichier de configuration.
- *Service.pm* et le répertoire *Service* : les implémentations des services.

Editeur WYSIWYG

Le répertoire `twe` contient les ressources de l'éditeur :

- `twe`
    - `twe.jar` : jar exécutable signé et déployable sur un serveur TWiki
    - `twe.jnlp` : le descripteur Java Web Start (cf. installation de l'éditeur)

Outil de refactoring

Le répertoire `refactoring` contient les ressources de l'outil de refactoring :

- `refactoring.jar` : jar exécutable signé et déployable sur un serveur TWiki
- `Refactor.jnlp` : le descripteur Java Web Start (cf. installation de l'éditeur)
- `src` : répertoire contenant le code source de l'éditeur
- `doc` : la documentation

Procédure d’installation

Service

Copie de fichiers

Copier le contenu du répertoire *bin* dans **TWiki Install Dir*/bin*, et le contenu du répertoire *lib* dans **TWiki Install Dir*/lib*. Il ne faut pas oublier de donner les permissions d?exécution aux scripts *jnlplauncher* et *service*. Il sera peut être nécessaire de modifier les entêtes des scripts pour être exécutables par l?interpréteur Perl sur votre configuration.

Configuration

Ouvrir dans un éditeur de texte le fichier *Service.cfg*. Les valeurs données par défaut peuvent être laissées telles quelles à l?exception de: *endpoint* et *jnlpDir*. *endpoint* doit définir l?url du serveur SOAP, par exemple http://hostname/bin/service. *jnlpDir* doit définir le répertoire où le script *jnlplauncher* devra chercher les descripteurs Web Start JNLP, par exemple /usr/home/httpd/twiki/jnlp/.

Editeur WYSIWYG

Copie de fichiers

Copier les fichiers `twe.jar` et `twe.jnlp` dans le répertoire spécifié par *jnlpDir* dans *Service.cfg*.

Configuration du descripteur Java Web Start

Ouvrir dans un éditeur de texte le fichier *twe.jnlp*. Par défaut le codebase est placé sur la machine de test du projet (par défaut : http://miageprojet.unice.fr/twikitestnice/) il faut modifier cette valeur par l'URL ou l'ont peu télécharger les fichiers `twe.jar` et `twe.jnlp`.

Exemple : jnlp spec="1.0+" codebase="http://twikiserveur.fr/jnlp/">

Modification des templates TWiki

Pour permettre l'édition en ligne d'un topic TWiki, il faut ajouter un lien dans le *template* correspondant au topic considéré.

Ce lien est le suivant :

```
<a href='/cgi-
bin/twiki/bin/jnlplauncher/Minfo/TerBn1Installation?jnlp=twe.jnlp>
```

Il s'agit d'un appel au script jnlplauncher avec en argument le fichier jnlp et ses arguments.

Outil de refactoring

La procédure d'installation de l'outil de refactoring est très proche de celle de l'éditeur.

Copie des fichiers

Il faut copier les fichiers refactoring.jar et Refactor.jnlp dans le répertoire *jnlpDir* dans *Service.cfg*, puis modifier le codebase dans le fichier Refactor.jnlp à l'aide d'un éditeur de texte, pour mettre la valeur de l'URL d'accés à ces fichiers depuis internet ou l'intranet (http://nom_serveur/twiki/ par exemple).

Modification des templates TWiki

Pour lancer l'application depuis une page web, il faut ajouter le lien suivant dans les *templates* :

```
<a href="/cgi-
bin/twiki/bin/jnlplauncher/Minfo/TerBn1Installation?jnlp=refactor
ing.jnlp">Refactoring tool
```

Annexe 1 : Services Web et structures de données

Structures de données

ConnectionInfo

Champs	Types	Commentaire
key	int	Clef de connexion
timeout	int	Timeout auquel TWiki est réglé avant de nous déconnecter
subwebs	boolean	Sous-webs gérés ou non par le service
adminLock	boolean	Verrou administratif (concurrence entre applications de refactoring) géré par le service ?
locksActualization	boolean	Actualisation des verrous (administratifs et topics) lors d'un ping ou gestion à faire depuis le client ?
completeRemoveAllowed	boolean	Suppression complète des topics (et attachements) autorisée ?

User

Champs	Types	Commentaire
usage	String	Renseignement sur l'usage du service qu'en fait l'utilisateur
login	String	Login de l'utilisateur
connection	java.util.Calendar	Date de connexion de l'utilisateur

Attachment

Champs	Types	Commentaire
name	String	Nom de l'attachement
path	String	Chemin du fichier avant l'upload
attr	string	Attachement caché ?
date	int	Date (en secondes)
user	String	Utilisateur qui l'a posté
size	int	Taille
version	Float	Version de l'attachement
comment	String	Commentaire associé
content	byte[]	Contenu binaire

Topic

Champs	Types	Commentaire
web	String	Web dans lequel est stoqué le topic
name	String	Nom du topic

version	Float	Version du topic (RCS)
author	String	Auteur du topic
parent	String	Topic parent
date	int	Date de création du topic (en secondes)
format	String	Format TWiki du topic (1.0)
attachments	Attachment[]	Tableau d'attachements
data	String	Contenu du topic

Services Web

---

connect

- Auteur : Romain RAUGI
- Version : CVS 1.7
- Package : **Service::Connection**

Description

Connexion d'un utilisateur au *service* (tout service requiert une connexion préalable).

Signature

- Paramètre(s) :
  - **String** usage : Usage que veut faire un utilisateur du service
  - **String** login : Login de l'utilisateur
  - **String** password : Password
- Valeur(s) retournée(s) :
  - **ConnectionInfo** : Connexion établie
  - null : Login/pass incorrect ou service saturé

---

copyTopic

- Auteur : Romain RAUGI
- Version : CVS 1.5
- Package : **Service::Refactoring**

Description

Copie de topic avec changement possible de parent et de nom.

Signature

- Paramètre(s) :
  - **int** key : Clef de connexion
  - **String** srcWeb : Web du topic source (syntaxe = Web[/Web]*)
  - **String** topic : Topic à copier ("WebHome" par exemple)

        - **String** dstWeb : Web destination
        - **String** newName : Nouveau nom
        - **String** parent : Topic parent auquel le rattacher ("" = WebHome)
        - **boolean** attachments : Copie des attachements ?
- Valeur(s) retournée(s) :
    - **int** : Code d'erreur
        - 0 : OK
        - 1 : Non connecté
        - 2 : Pas de verrou adminsitartif déposé
        - 3 : Le topic source n'existe pas
        - 4 : Le nouveau nom correspond à un topic déjà existant
        - 5 : Le topic parent destination n'existe pas
        - 6 : Le nouveau nom n'est pas un WikiWord
        - 7 : Pas les permissions pour copier
        - 8 : Impossible de verrouiller le fichier cible (en cours de création par un autre)
        - 9 : Problème lors de la copie, c.a.d lors de la création du topic cible
        - 10 : Sous-webs pas autorisés

---

disconnect

- Auteur : Romain RAUGI
- Version : CVS 1.7
- Package : **Service::Connection**

Description

Déconnexion du service.

Signature

- Paramètre(s) :
    - **int** key: Clef de connexion
- Valeur(s) retournée(s) :
    - true : Déconnexion effectuée
    - false: Déjà déconnecté auparavant probablement

---

getAttach

- Auteur : Maxime LAMURE
- Version : 1.0
- Package : **Service::Editeur**

Description

Ce service permet de récupérer un objet de type *Attachment* correspondant au nom de fichier donné en paramètre.

Signature

- Paramètre(s) :
    - param1: le chemin du topic
    - param2: le nom du fichier
- Valeur(s) retournée(s) :
    - objet de type *Attachment* dont les propriétés correspondent au fichier donne en paramètre.

Exemple

```
Attachment att=stub.getAttach("Main/WebList.txt",”toto.avi”);
System.out.println(att.getDate());
System.out.println(att.getName());
```

---

getListAttach

- Auteur : Maxime LAMURE
- Version : 1.0
- Package : **Service::Editeur**

Description

Ce service renvoie la liste de tous les noms de fichiers en attachement au topic passé en paramètre.

Signature

- Paramètre(s) :
    - param1 : chemin du topic
- Valeur(s) retournée(s) :
    - Un tableau de chaîne de caractères contenant les noms des fichiers d’attachement.

Exemple

```
String [] liste=stub.getListAttach("Main/WebList.txt");
System.out.println("fichier : "+liste[0]);
-> fichier : toto.avi
```

---

getTopic

- Auteur : Maxime LAMURE
- Version : 1.0
- Package : **Service::Editeur**

Description

Récupère un objet de type Topic instancie avec les propriétés du topic dont le nom est donné en paramètre.

Signature

- Paramètre(s) :
    - param1 : chemin du topic
    - param2 : La clé unique de l'utilisateur
- Valeur(s) retournée(s) :
    - Une instance d'un objet de type Topic correspondant au topic donné en paramètre.

Exemple

```
Topic p=stub.getTopic("Main/WebSearch.txt",125);
System.out.println("Autheur: "+p.getAuthor());
-> Auteur: PeterThoeny
```

---

getUsers

- Auteur : Romain RAUGI
- Version : CVS 1.7
- Package : **Service::Connection**

Description

Liste des utilisateurs connectés au service.

Signature

- Paramètre(s) :
- Valeur(s) retournée(s) :
    - User[] : Liste des utilisateurs (null si aucun)

---

getWebs

- Auteur : Romain RAUGI
- Version : CVS 1.5
- Package : **Service::Topics**

Description

Liste des webs ou des sous-web si un web est passé en paramètre.

Signature

- Paramètre(s) :
    - **String** root : "" pour obtenir les webs, un web pour en obtenir les sous-webs.
- Valeur(s) retournée(s) :
    - **String[]** : Liste des webs

Exemple

```
String[] webs = stub.getWebs("");
System.out.println(webs);
-> Main _default TWiki Trash Sandbox
```

---

giveHierarchy

- Auteur : Maxime LAMURE
- Version : 1.0
- Package : Service::Arborescence

Description

Ce service ressemble à la commande ls en UNIX.

Signature

- Paramètre(s) :
    - param1 : Le chemin du Web
    - param2 : La clé unique de l'utilisateur
- Valeur(s) retournée(s) :
    - Un tableau de chaîne de caractères contenant les noms des Web et des topics présent dans param1. Les fichiers se terminant par .txt sont des topics, les autre des Webs.

Exemple

```
String [] result=stub.giveHierarchy("../data/Main",125);
for(int i=0;i<result.length ;i++)
   System.out.println(result[i]);

-> ANewPage.txt
  ImaeWeb.txt
  WebHome.txt
  ...
```

---

giveTopicProperties

- Auteur : Maxime LAMURE
- Version : 1.0
- Package : Service::Arborescence

Description

Ce service permet de récupérer les propriétés d'un topic.

Signature

- Paramètre(s) :
    - param1 : Le chemin du topic
    - param2 : La clé unique de l'utilisateur

Valeur(s) retournée(s)

Un tableau de chaîne de caractères répertoriant les propriétés du topic. Pour les droits, la convention est la suivante :

1.: aucun droit d'écriture
2.: droit de change
3.: droit de renomage
4.: droit de change et de renomage

Exemple

```
String[]prop=stub.giveTopicProperties("Main/toto.txt",125);
System.out.println(“ taille: ”+prop[0]);             ->  25685
System.out.println(“ web: ”+prop[1]);                -> Main
System.out.println(“ chemin: ”+prop[2]);             -> /twiki/data/Main
System.out.println(“ topic parent: ”+prop[3]);  -> MainPage
System.out.println(“ auteur: ”+prop[4]);             -> guest
System.out.println(“ date creation: ”+prop[5]); -> lundi 18 avril
System.out.println(“date modification: ”+prop[6]); -> mardi 1 mai
System.out.println(“ version: ”+prop[7]);            -> 1.1
System.out.println(“ format: ”+prop[8]);             -> 1.0
System.out.println(“ droit: ”+prop[9]);              -> 3
```

---

giveWebProperties

- Auteur : Maxime LAMURE
- Version : 1.0
- Package : Service::Arborescence

Description

Permet de récupérer la propriété du Web.

Signature

- Paramètre(s) :
    - param1 : le chemin du Web.
- Valeur(s) retournée(s) :
    - Un entier représentant le nombre en octet. Cette taille n'est pas récursive.

Exemple

```
int taille=stub.giveWebProperties("Main");
System.out.println("La taille est de: "+taille);
```

---

lockTopic

- Auteur : Romain RAUGI
- Version : CVS 1.5
- Package : **Connection::Topics**

Description

Verrouillage ou déverrouillage explicite de topic. Le déverrouillage n'est autorisé qu'au détenteur du verrou (via username si déposé par TWiki lui-même, via clef de connexion si déposé via service).

Signature

- Paramètre(s) :
  - **int** key : Clef de connexion
  - **String** web : Web du topic (syntaxe = Web[/Web]*)
  - **String** topic : Topic à renommer (syntaxe = WikiWord)
  - **boolean** doLock: Verrouiller (true) ou déverrouiller (false)?
- Valeur(s) retournée(s) :
  - **boolean** : Echec ou succès

A noter : retournera succès lors d'un déverrouillage même s'il n'y a pas de verrou déposé.

---

mergeTopics

- Auteur : Romain RAUGI
- Version : CVS 1.5
- Package : **Service::Refactoring**

Description

Fusion de topics.

Signature

- Paramètre(s) :
  - **int** key : Clef de connexion
  - **String** webTarget : Web du topic n°1 (syntaxe = Web[/Web]*)
  - **String** topicTarget : Topic "destination" participant à la fusion (syntaxe = WikiWord)
  - **String** webFrom : Web du topic n°2
  - **String** topicFrom : Topic n°2 participant à la fusion (syntaxe = WikiWord)
  - **boolean** attachments : Gérer les attachements dans la fusion ?
  - **boolean** identify : Génération ou non d'un rapport (page en mode verbatim) indiquant de manière détaillée le résultat de la fusion
  - **int** removeOption : Options concernant le topic n°2
    - -1 : Pas de suppression
    - cf. removeTopic pour les autres codes
  - **boolean** dontNotify : Notifier ou non du changement
- Valeur(s) retournée(s) :
  - **int** : Code d'erreur
    - 0 : OK
    - 1 : Non connecté
    - 2 : Pas de verrou administratif
    - 3 : Le topic n°1 n'existe pas
    - 4 : Le topic n°2 n'existe pas
    - 5 : Les topics 1 et 2 sont les mêmes

            - 6 : Pas les permissions de modifier le topic n°1
            - 7 : Topic n°1 déjà verrouillé
            - 8 : Problème lors de la modification du topic n°1
            - 9 : Problème lors de la suppression du topic n°2 (si demandée)
            - 10 : Sous-webs pas autorisés

---

moveTopic

- Auteur : Romain RAUGI
- Version : CVS 1.5
- Package : **Service::Refactoring**

Description

Déplacement de topic avec changement de parent possible.

Signature

- Paramètre(s) :
    - **int** key : Clef de connexion
    - **String** srcWeb : Web du topic (syntaxe = Web[/Web]*)
    - **String** topic : Topic à renommer (syntaxe = WikiWord)
    - **String** dstWeb : Web destination
    - **String** parent : Topic parent auquel le rattacher ("" = WebHome)
    - **int** update : Mise à jour des liens qu'y s'y réfèrent
        - 0 : Pas de maj
        - 1 : Maj dans tous les webs
        - 2 : Maj dans les webs source et destination
- Valeur(s) retournée(s) :
    - **int** : Code d'erreur
        - 0 : OK
        - 1 : Non connecté
        - 2 : Pas de verrou administratif
        - 3 : Le topic source n'existe pas
        - 4 : Le topic parent destination n'existe pas
        - 5 : Pas les permissions pour renommer
        - 6 : Topic déjà verrouillé
        - 7 : Le topic existe déjà dnas le Web destination
        - 8 : Problème lors du déplacement Web source/ Web destination
        - 9 : Problème lors de la modification du lien vers le topic parent
        - 10 : Sous-webs pas autorisés

---

ping

- Auteur : Romain RAUGI
- Version : CVS 1.7
- Package : **Service::Connection**

Description

Avertissement au serveur qu'on est toujours présent, actualisation des verrous déposés (si service configuré pour).

Signature

- Paramètre(s) :
    - **int** key: Clef de connexion
- Valeur(s) retournée(s) :
    - true : On est toujours connecté
    - false: On l'est plus

---

removeSubscription

- Auteur : Romain RAUGI
- Version : CVS 1.2
- Package : **Service::Notification**

Description

Résialiation d'abonnement aux événements TWiki.

Signature

- Paramètre(s) :
    - **int** key : Clef de connexion
- Valeur(s) retournée(s) :
    - true : On est abonné
    - false: On l'est pas (plus) ou le service ne gère pas les notifications

---

removeTopic

- Auteur : Romain RAUGI
- Version : CVS 1.5
- Package : **Service::Refactoring**

Description

Suppression de topic. Attention, ce qui s'applique aux topics s'applique aux attachements ...

Signature

- Paramètre(s) :
    - **int** key : Clef de connexion
    - **String** web : Web du topic (syntaxe = Web[/Web]*)
    - **String** topic : Topic (syntaxe = WikiWord)

    - **int** option : Web destination
        - 0 : Déplacement dans le web Trash, génération automatique d'un ID si nom de topic y existe déjà (si service configuré pour)
        - 1 : Déplacement dans le web Trash, suppression de celui qui existerait éventuellement dans Trash (si fonctionnalité autorisée par le service)
        - 2 : Suppression complète (si fonctionnalité autorisée par le service)
        - Autre : Déplacement dans le web Trash, erreur si topic de même nom y existe déjà (pas de génération automatique d'ID)
    - **String** trashName : Nom à donner au topic dans le web Trash (" = topic)
- Valeur(s) retournée(s) :
    - **int** : Code d'erreur
        - 0 : OK
        - 1 : Non connecté
        - 2 : Pas de verrou administratif
        - 3 : Le topic n'existe pas
        - 4 : Pas les permissions de supprimer le topic
        - 5 : Topic déjà verrouillé
        - 6 : TrashName n'est pas un WikiWord
        - 7 : Impossible de supprimer celui qui se trouve dans le web Trash
        - 8 : Problème lors de l'envoi vers le web Trash
        - 10 : Sous-webs pas autorisés

---

renameTopic

- Auteur : Romain RAUGI
- Version : CVS 1.5
- Package : **Service::Refactoring**

Description

Renommage de topic, de la "pile" RCS associée et de la location des attachements.

Signature

- Paramètre(s) :
    - **int** key : Clef de connexion
    - **String** web : Web du topic (syntaxe = Web[/Web]*)
    - **String** topic : Topic à renommer (syntaxe = WikiWord)
    - **String** name : Nouveau nom (WikiWord, sinon retour d'un code d'erreur)
    - **int** update : Mise à jour des liens qu'y s'y réfèrent
        - 0 : Pas de maj
        - 1 : Maj dans tous les webs
        - 2 : Maj au sein du web
- Valeur(s) retournée(s) :
    - **int** : Code d'erreur
        - 0 : OK
        - 1 : Non connecté
        - 2 : Pas de verrou administratif
        - 3 : Le topic n'existe pas

- 4 : Le nouveau nom correspond à un topic existant
- 5 : Le nouveau nom donné n'est pas un WikiWord
- 6 : Pas les permissions pour renommer
- 7 : Topic déjà verrouillé
- 8 : Problème lors du renommage
- 10 : Sous-webs pas autorisés

---

sameTopicParent

- Auteur : Maxime LAMURE
- Version : 1.0
- Package : Service::Arborescence

Description

Ce service retourne une représentation de l'arborescence en fonction des paramètres donnés:

Signature

- Paramètre(s) :
  - param1: Le nom du topic parent
  - param2 : Le nom du Web où commence la recherche
  - param3: La clé unique de l'utilisateur
- Valeur(s) retournée(s) :
  - Un tableau de chaîne de caractères regroupant les chemins des topics ayant le même topic parent. Si le topic retourné est lui-même un topic parent, alors il se termine par #
- Quelques exemples
  - sameTopicParent("","nomWeb",clé) :

Renvoie la liste des topics parents et la liste des topic qui n'on pas de parent et qui ne sont pas parent. Pour connaître la liste des topic enfant, il suffit de rappeler cette méthode avec le nom du topic parent en paramètre. Si un parent est aussi enfant, il ne sera affiche qu'en appelant la fonction avec le nom de son parent. Les cycles sont gérés.

  - sameTopicParent("Nomtopic","",clé):

Renvoie la liste des topic qui ont Nomtopic comme parent.

  - sameTopicParent("Web.Nomtopic","",clé) :

Renvoie la liste des topic qui ont Nomtopic ou Web.Nomtopic comme parent.

Exemple

```
String []parent=stub.sameTopicParent("MainPage","",125);
for(int i=0;i<parent.length ;i++)
System.out.println(parent[i]);

->    /Main/Toto.txt
```

---

setAdminLock

- Auteur : Romain RAUGI
- Version : CVS 1.7
- Package : **Service::Connection**

Description

Verrou administratif.

Signature

- Paramètre(s) :
    - **int** key: Clef de connexion
    - **boolean** doLock : Verrouiller ou déverrouiller ?
- Valeur(s) retournée(s) :
    - true : Ok
    - false: Pas (dé)verrouillable

---

setTopic

- Auteur : Romain RAUGI
- Version : CVS 1.5
- Package : **Service::Topics**

Description

Sauvegarde d'un topic voire d'attachements

Signature

- Paramètre(s) :
    - **int** key : Clef de connexion
    - **Topic** topic : Objet topic à sauvegarder
    - **boolean** doKeep : Supprimer les attachements dont on n'a pas renvoyé les infos ?
    - **boolean** doUnlock : Déverrouiller ?
    - **boolean** dontNotify : Notifier du changement aux autres ?
- Valeur(s) retournée(s) :
    - **int** : Code d'erreur
        - 0 : OK
        - 1 : Non connecté
        - 2 : Le topic n'existe pas
        - 3 : Topic nul passé en paramètre
        - 4 : Pas les permissions
        - 5 : Topic verrouillé par qqun d'autre
        - 6 : Problème lors de la modification

        - 10 : Sous-webs pas autorisés

---

subscribe

- Auteur : Romain RAUGI
- Version : CVS 1.2
- Package : **Service::Notification**

Description

Abonnement aux événements TWiki.

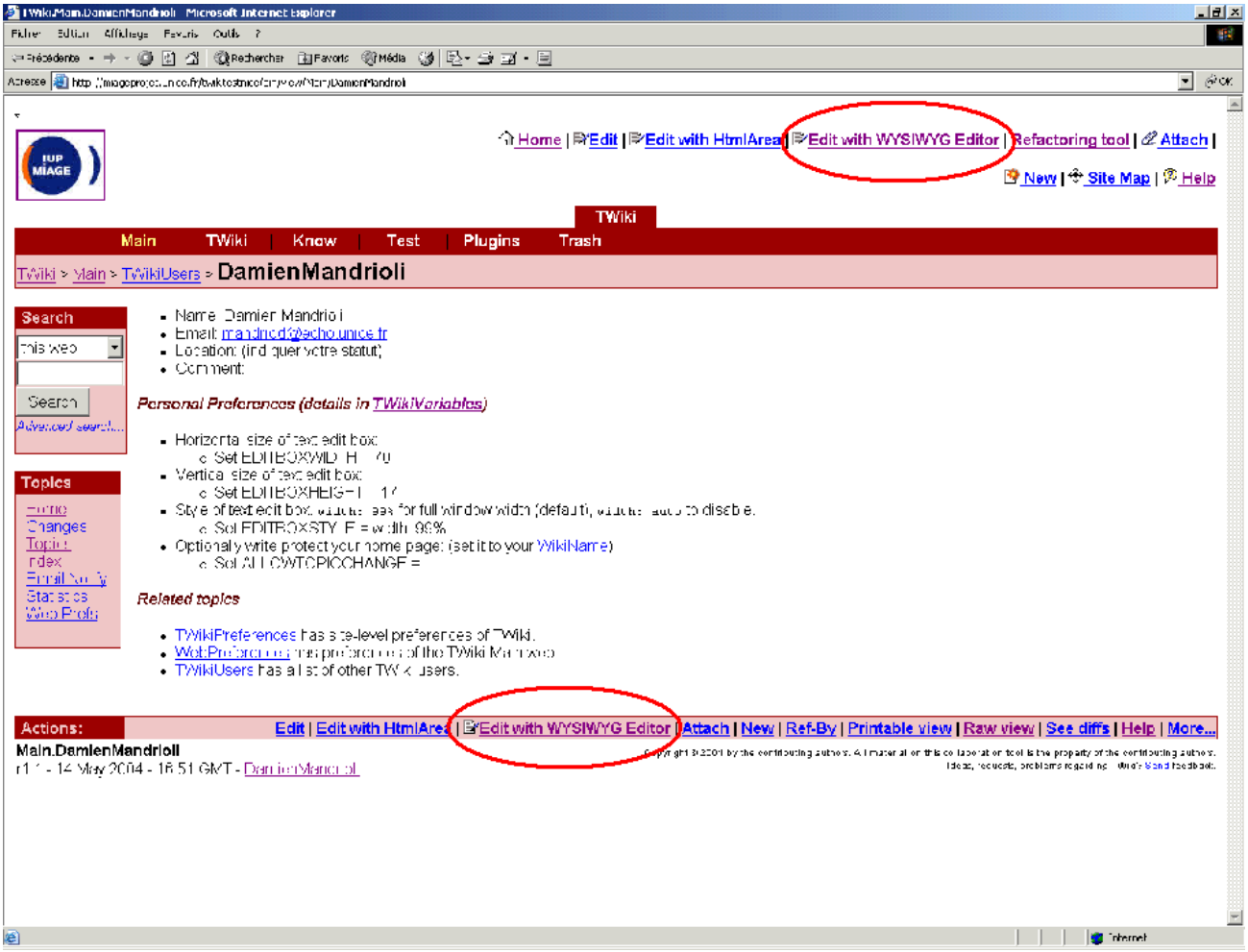
Signature

- Paramètre(s) :
    - **int** key : Clef de connexion
    - **int** port : Port sur lequel on veut recevoir les notifications
- Valeur(s) retournée(s) :
    - true : Ok
    - false: Clef incorrecte ou service de notifications désactivé

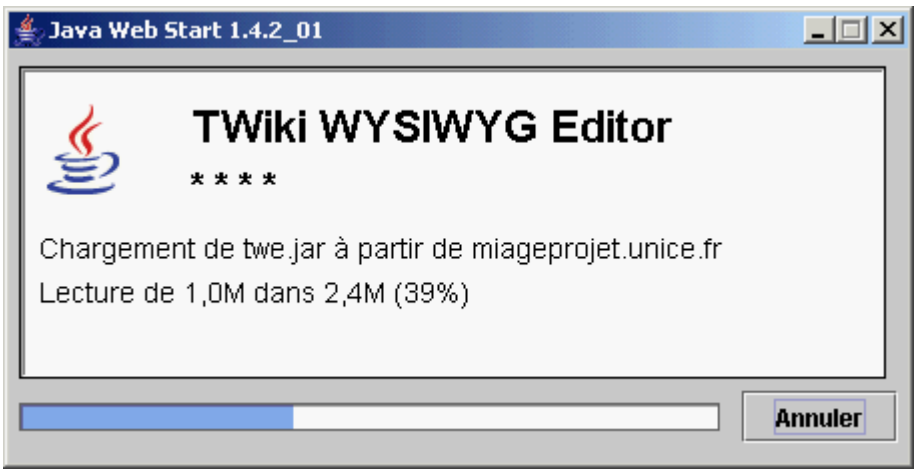
Annexe 2 : Documentation utilisateur de l'éditeur

Premier lancement de l'application

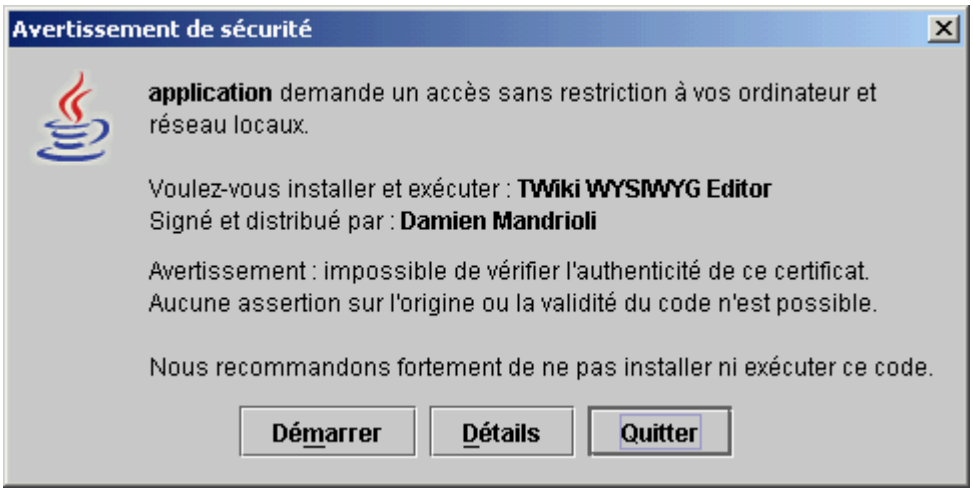
Il est normalement réalisé depuis un topic TWiki par l'intermédiaire de Java Web Start. Il suffit de cliquer sur un lien tel que représenté ici :



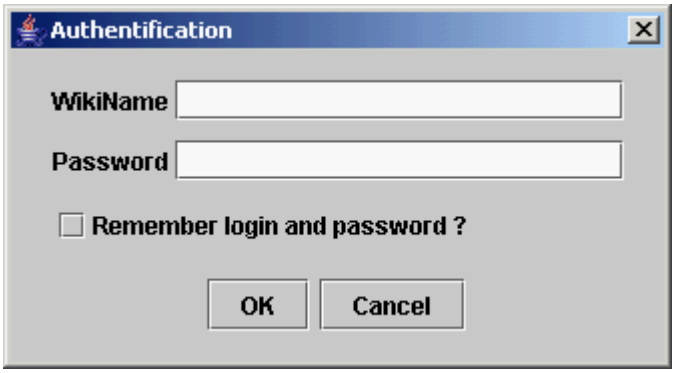
Après avoir cliquer, le téléchargement commence. Ce téléchargement n'aura lieu qu'en cas de mise à jour de l'éditeur côté serveur.



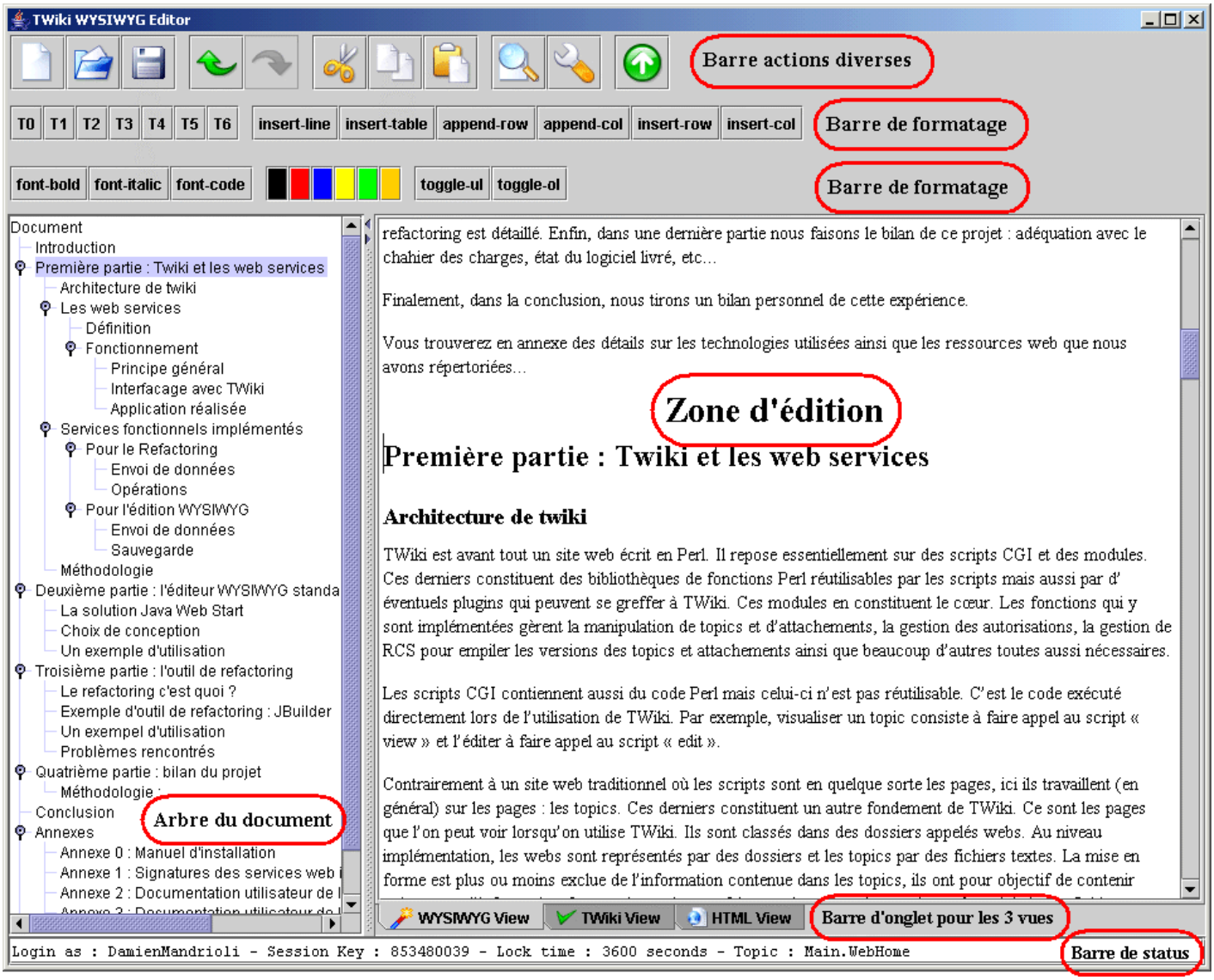
Ensuite apparaît la fenêtre suivante. Il faut répondre **Démarrer**, afin d'autoriser l'éditeur à accéder au poste local pour la sauvegarde d'un topic par exemple.



Enfin, la fenêtre de login s'ouvre, il est alors possible de s'identifier pour éditer la page.



Utilisation de l'éditeur



Barre actions diverses

Ceci est la barre de bouton qui commande les actions suivantes (de gauche à droite) :

- *Nouveau document* : Propose la sauvegarde du document actif et crée un document vide.
- *Ouvrir document* : Propose l'ouverture d'un document au format HTML (.html) ou TWikiML (.twiki). Il est déconseillé d'ouvrir un document HTML avec l'éditeur car il pourrait contenir des balises non reconnues.
- *Enregistrer document* : Propose l'enregistrement du document au format .twiki
- *Défaire* : Annule la dernière opération de formatage réalisée.
- *Refaire* : Le pendant de Défaire.
- *Couper*
- *Copier*
- *Coller*
- *Rechercher*
- *Configurer l'éditeur* : Permet de choisir si le mot de passe doit être enregistré dans la machine.

- *Validation coté serveur* : Une fois l'édition terminée, permet d'enregistrer le topic sur le serveur.

Arbre du document

Il permet de naviguer dans le document. En cliquant sur un élément de l'arbre, le curseur se déplace dans la section correspondante. Il est construit à partir de la hiérarchie de titres du document.

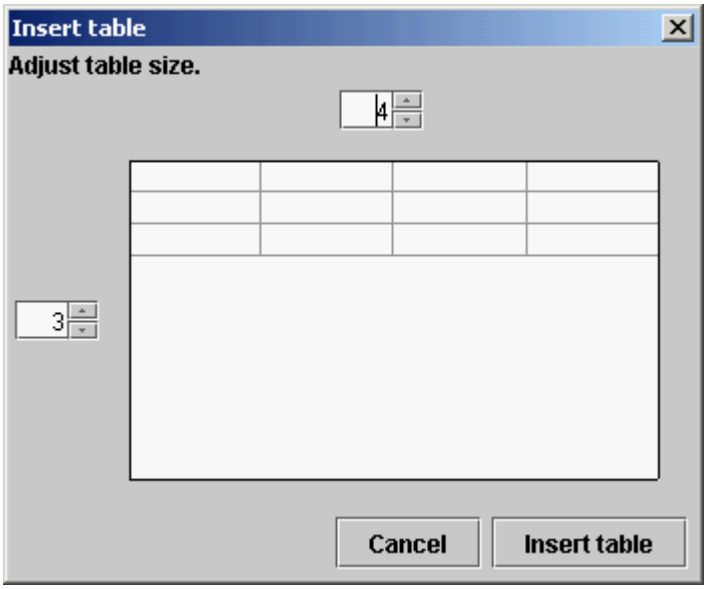
Barre d'onglet

Permet de sélectionner la vue à éditer. Les trois vues sont synchronisées, ainsi on peux éditer n'importe qu'elle vue pour mettre à jour le document.

Barre de formatage supérieure

Barre de bouton pour le formatage du texte. De gauche à droite :

- *T0,T1,...,T2* : Change le texte sélectionné en style : *paragraphe*, titre de taille 1,2,3,4,5,6. Si le texte sélectionné est du même type que le bouton choisis, le texte est mis au format *paragraphe*.
- *insert-line* : Insère à la position du curseur une ligne horizontale de séparation.
- *insert-table* : Insère un tableau. On peux choisir la taille du tableau via la fenêtre suivante :



- *append-row* : Ajoute une ligne à la fin du tableau dans lequel se trouve le curseur.
- *append-col* : Ajoute une colonne à la fin du tableau dans lequel se trouve le curseur.
- *insert-row* : Insère une ligne après la cellule du curseur.
- *insert-col* : Insère une colonne après la cellule du curseur.

Barre de formatage inférieure

- *font-bold* : Met en gras la sélection.
- *font-italic* : Met en italique la sélection.

- *font-code* : La sélection est transformée en police à chasse fixe.
- *Couleurs* : Colorie la sélection.
- *toggle-ul* : Place devant chaque ligne de la sélection un puce.
- *toggle-ol* : Place devant chaque colonne de la sélection un numéro croissant.

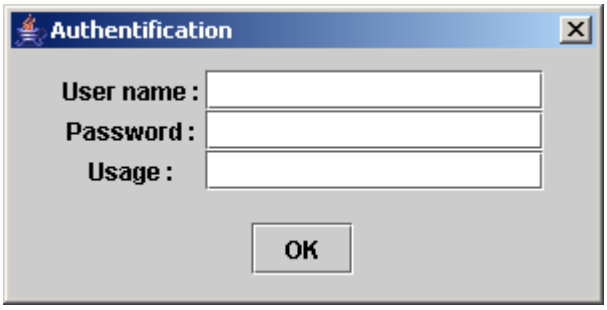
Comportements spécifiques

- Dans un tableau la touche TAB permet de passer à la cellule suivante, si le curseur est dans la dernière cellule, l'éditeur crée une nouvelle ligne
- Si le curseur se trouve dans une liste à puce ou numérotée, la touche ENTREE a pour effet la création d'une nouvelle puce (ou numéro suivant) à la ligne suivante.
- sinon, la touche ENTREE sert à commencer un nouveau paragraphe.

Annexe 3 : Documentation utilisateur de l'outil de refactoring

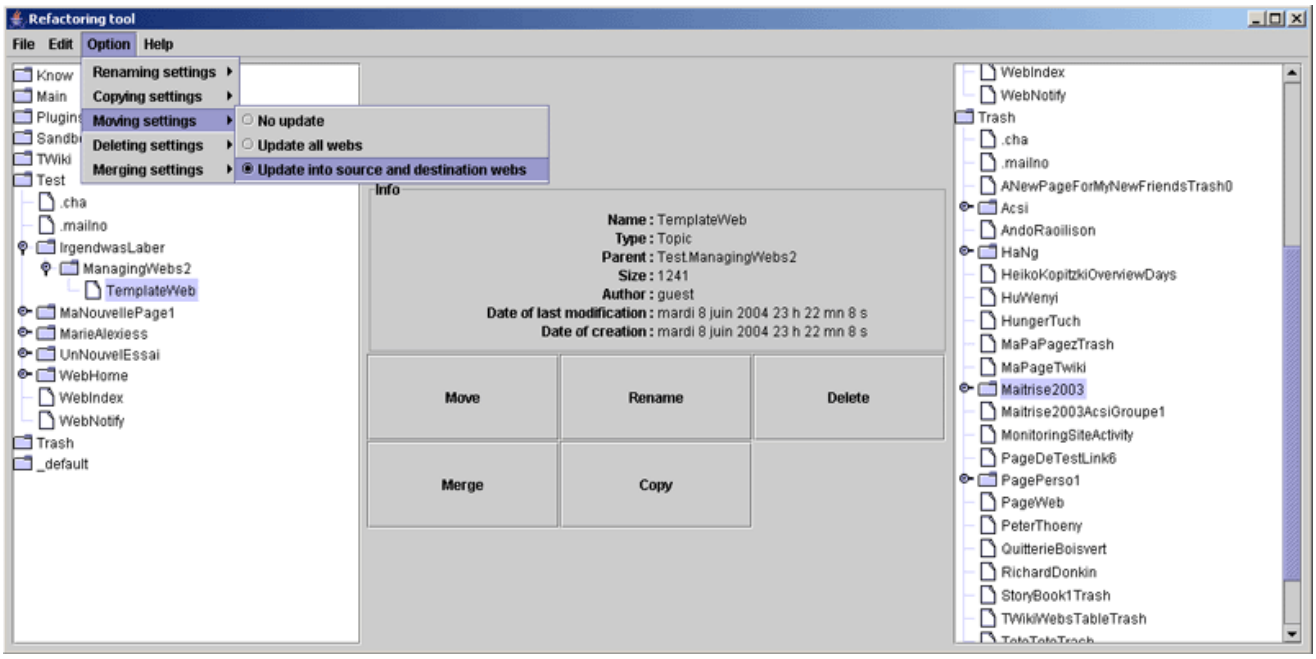
Lancement de l'application

L'outil de refactoring se lance à partir d'une page web du TWiki en cliquant sur le lien « Refactoring tool ». Une boîte de dialogue s'affiche permettant à l'utilisateur de se connecter : il doit saisir son nom d'utilisateur, son mot de passe et éventuellement le champ usage décrivant ses intentions.



Une fois que l'utilisateur est connecté, la fenêtre principale de l'application apparaît. On y retrouve deux arbres décrivant l'arborescence de TWiki, séparés par des boutons d'actions et un cadre de description des éléments de l'arborescence.

Pour effectuer une opération sur un topic, il faut le sélectionner dans l'arbre de gauche, et éventuellement choisir une destination dans l'arbre de droite si on veut effectuer un déplacement ou une copie, puis cliquer sur le bouton correspondant à la modification que l'on désire faire. Lorsque l'on clique sur un élément de l'arbre de gauche, ses caractéristiques sont affichées dans le cadre "info".



Description des fonctionnalités

Il est possible d'effectuer cinq actions différentes :

- le déplacement
- le renommage
- la suppression
- la fusion
- la copie

A chaque action correspond des options qui permettent d'être plus précis. Elle se situe dans le menu « Options ».

Le déplacement

Pour déplacer un topic ou un groupe de topics, il suffit de les sélectionner dans l'arbre de gauche et choisir une destination dans l'arbre de droite, puis d'appuyer sur le bouton « Move ». Si l'élément de destination sélectionné est un web, les topics seront déplacés dans le topic Web Home de ce web. Le webs ne peuvent pas être déplacés.

Option de déplacement

« No update » : pas de mise à jour des liens qui se réfèrent au topic

« update all webs » : mise à jour des liens dans tous les webs

« update into source and destination webs » : mise à jour dans les webs source et destination seulement

Le renommage

Pour renommer un topic, il faut le sélectionner dans l'arbre de gauche puis cliquer sur le bouton « Rename ». Une boite de dialogue s'affiche alors, permettant de saisir le nouveau Wiki Name? du topic. Le webs ne peuvent pas être renommés.

Option de renommage

« No update » : pas de mise à jour des liens qui se réfèrent au topic

« update all webs » : mise à jour des liens dans tous les webs

« update into the web of the topic » : mise à jour dans le web contenant le topic seulement

La suppression

Pour supprimer un topic ou un groupe de topics, il suffit de les sélectionner dans l'arbre de gauche et d'appuyer sur le bouton « Delete ». Le webs ne peuvent pas être supprimés.

Options de suppression

« move in trash web with renaming » : déplacement dans le web Trash, génération automatique d'un ID si le nom de topic y existe déjà

« move in trash web with overwritting » : Déplacement dans le web Trash, suppression de celui qui existerait éventuellement dans Trash

« complete removing » : Suppression complète

« move in trash web » : Déplacement dans le web Trash, une erreur est produite si un topic de même nom y existe déjà

La fusion

Pour fusionner des topics, il faut les sélectionner dans l'arbre de gauche puis cliquer sur le bouton « Merge ». Le topic principale de la fusion (celui dans lequel le contenu des autres topics sera ajouté) correspond au premier topic sélectionné. Le webs ne peuvent pas être fusionnés.

Options de fusion

« don't remove » : Ne supprime pas les topics

« manage attachments » : si elle est cochée, gère les fichiers attachés (les ajoutes au topic final)

« make report » : génère ou non un rapport (page en mode verbatim) indiquant de manière détaillée le résultat de la fusion

« don't notify » : Notification ou non du changement

Les autres options sont identiques aux options de suppression.

La copie

Pour copier un topic ou un groupe de topics, il suffit de les sélectionner dans l'arbre de gauche et choisir une destination dans l'arbre de droite, puis d'appuyer sur le bouton « Copy ». Si l'élément de destination sélectionné est un web, les topics seront déplacés dans le topic Web Home de ce web. Le webs ne peuvent pas être copiés

Option de copie

« Copy attachments » si cette option est sélectionné, les fichiers attachés sont copier dans le web de destinations

Description de la barre de menu

Le menu « File »

Ce menu contient uniquement un bouton exit pour quitter l'application

Le menu « Edit »

Ce menu contient tous les boutons correspondant aux fonctionnalités décrites ci-dessus, et également le bouton « set administrator lock ». Pour pouvoir faire des modifications, il faut que le verrou administratif soit posé afin que plusieurs utilisateurs ne puissent pas

modifier l’arborescence en même temps. Si la case est coché, le verrou est posé sinon il ne l’est pas.

Le menu « Options »

Ce menu contient toutes les options des opérations décrites dans la partie précédente.

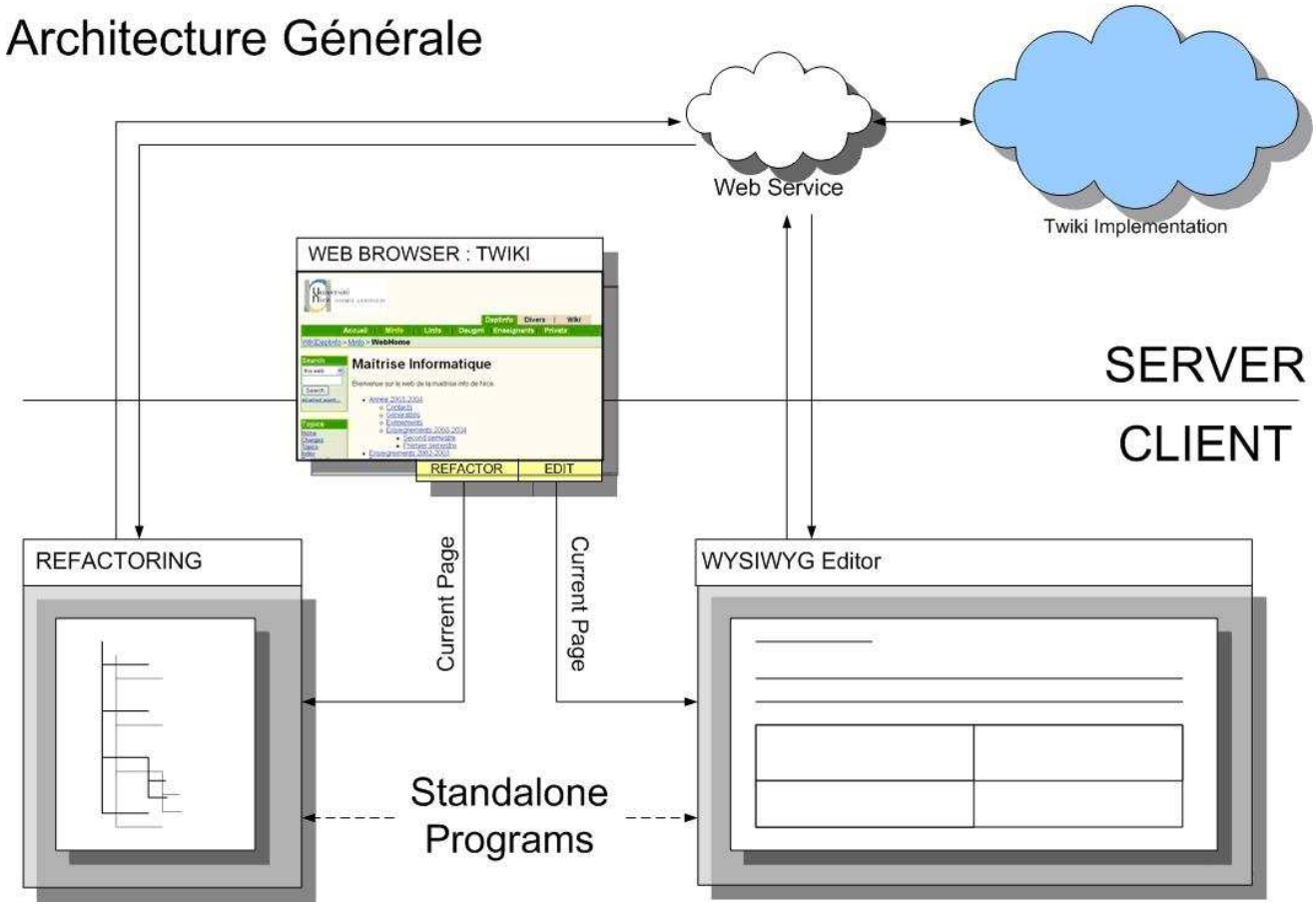
Le menu « Help »

Ce menu contient le bouton « Connected users… » permettant d’afficher dans une boîte de dialogue la liste des utilisateurs connectés au système.

Annexe 4 : Documentation de l'application serveur pour développeurs

Rôle des services Web

Les services Web permettent aux outils d'édition WYSIWYG et de refactoring développés dans le cadre de notre projet d'interagir avec TWiki. Sauvegarder ou rapatrier un topic, le déplacer ou le supprimer par exemple, se fait grâce à eux. TWiki est conçu pour être utilisable directement par un internaute. Les services étendent son utilisation aux applications en général. Voici un schéma illustrant cette idée:

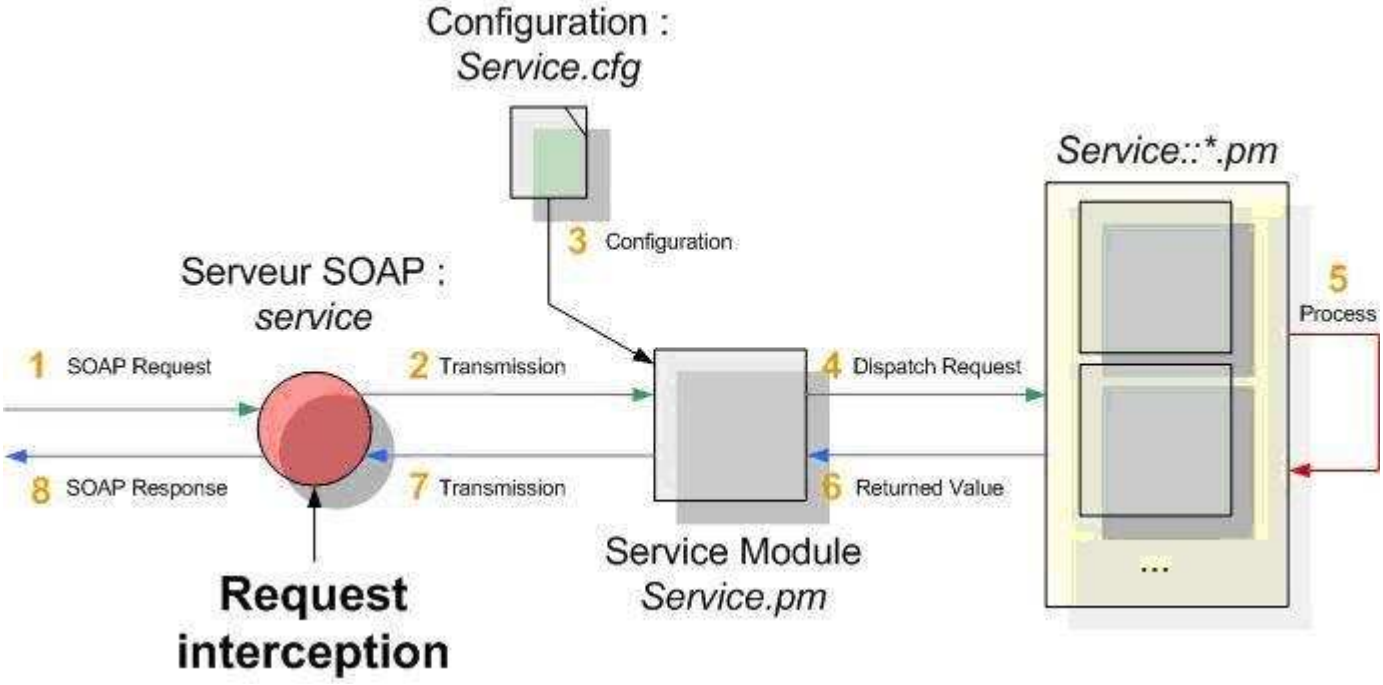


Les services Web font interface entre TWiki et nos applications. Dans le cadre de notre projet, les outils développés sont lancés via Java Web Start mais ce n'est pas du tout obligatoire pour les utiliser. Dans cette illustration, l'indication « Current Page » marquée sur certaines flèches indique des arguments Web Start précisant quel est le topic courant pour le lancement des applications. Le rapatriement de topics ou la récupération de hiérarchies se fait par service et n'a plus rien à voir avec Web Start.

Services

Application

Les services développés ne sont pas des fonctions réparties au sein du code source de TWiki mais font l'objet d'une application à part entière. Du point de vue du serveur, l'application s'initialise lors de l'interception d'une requête et se termine une fois le traitement effectué; les requêtes pouvant être une demande d'opération de refactoring, de récupération de topic, de sauvegarde etc. Voici un schéma illustrant ce qu'il se passe entre ces deux instants clefs:



Le serveur SOAP a pour objectif d'écouter sur un port donné afin d'intercepter des messages SOAP. Une fois le message intercepté, il le désérialise et transmet la requête à un module *Service*. Ce module se charge d'initialiser la configuration du service en paramétrant des variables globales, de la même manière que TWiki. Une fois la configuration effectuée, le module répartit les requêtes dans des sous modules et renvoie le résultat (au serveur SOAP). Il agit en quelque sorte comme un skeleton de serveur.

Le résultat retourné est converti en message SOAP par le serveur SOAP. Le contenu XML du message SOAP peut être écrit de manière automatique ou manuelle (dans le cas de booléens ou de structures complexes par exemple). Perl étant atypé et *SOAP::Lite* n'offrant pas d'équivalence objet/structure complexe XML, on ne peut avoir une sérialisation XML aussi automatique qu'en C# par exemple. Le module *Service::Conversions* a été développé afin de réaliser explicitement certaines sérialisations.

Aspects non fonctionnels

Une grande partie des implémentations réalisées du coté de TWiki concerne les aspects non fonctionnels des services. Parmi eux, la gestion de la concurrence et la gestion des connexions afin de suivre et authentifier les clients du coté du serveur, sont les plus importantes. Mais d'autres aspects ont aussi du être gérés ...

Persistance des données entre appels de services

Principe

Les services Web sont des méthodes accessibles à distance. Une fois exécutées, elles sont lancées dans des threads indépendants du côté du serveur. Il est nécessaire d'avoir un moyen de faire partager des données entre ces threads et de permettre de les sauvegarder une fois un appel terminé. La gestion des connexions a besoin de ce cela par exemple. Plusieurs moyens existent parmi lesquels les segments de mémoire partagée ou plus simplement la sauvegarde sur disque dur dans un fichier. Nous avons opté pour cette dernière technique, la plus simple mais aussi la moins risquée en ce qui concerne la compatibilité entre systèmes d'exploitation. Les services qui requièrent des données persistantes doivent donc dans un premier temps charger le contenu du fichier (dans une table de hachage la plupart du temps) et ensuite le mettre à jour si nécessaire.

Gestion de la concurrence

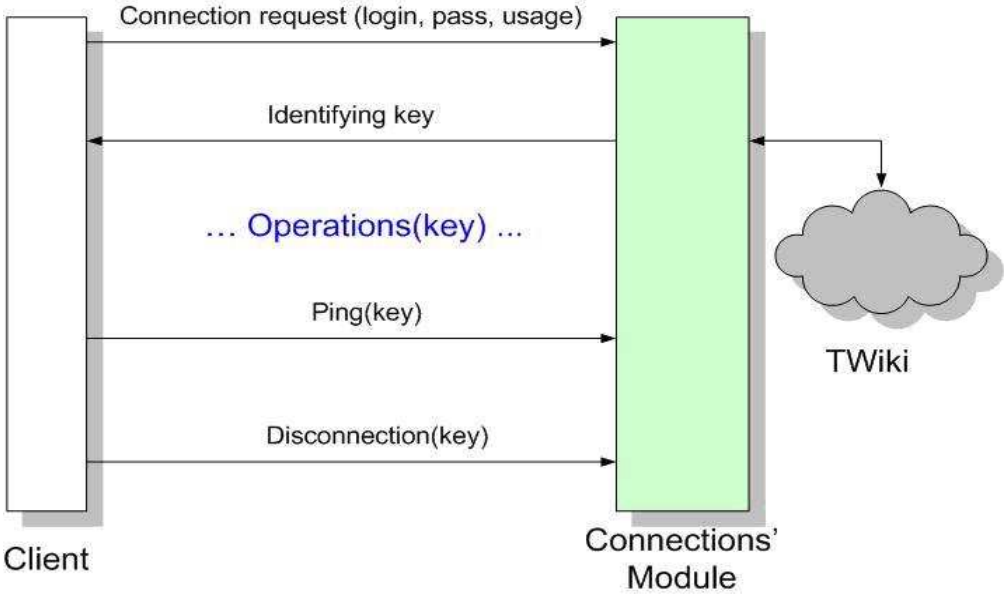
Sauvegarder et lire des données dans et depuis un même emplacement implique gérer la concurrence. Dans le monde UNIX, la commande *flock* aurait été parfaite pour verrouiller l'intégralité du fichier pendant un traitement et le déverrouiller ensuite. De plus, Perl nous permet d'y faire appel, que ce soit sous UNIX ou sous Windows. Mais *flock* a beau être disponible sous Windows, elle n'a aucun effet.

Nous avons donc développé un système plus ou moins similaire, inspiré des verrous utilisés dans TWiki. Le verrou utilisé est un fichier nommé du même nom que le fichier à verrouiller, avec comme extension *.lock*. Il est créé dans le même répertoire et contient sa date de création. Le module *Service::FileLock* implémente ces fonctions de verrouillage et de déverrouillage.

Connexion et authentification

Concept

Pour gérer les communications entre TWiki et nos applications, nous avons imaginé un système de connexion. Il permet notamment d'authentifier une seule fois un utilisateur. Plutôt qu'une longue description, voici un schéma montrant une séquence d'opérations :



Un client envoie une requête de connexion, en donnant son login (*WikiName*), son mot de passe (excepté si l'utilisateur est invité « guest »), et l'usage qu'il veut faire du service. La procédure de connexion va alors faire certains tests parmi lesquels authentifier le client à partir du login et du mot de passe fourni. Si tout se déroule correctement, une clef identifiante sera envoyée au client. Cette clef sera requise par chaque service nécessitant des droits d'accès. Du coté du service, certaines informations seront sauvegardées. Ceci est décrit dans la section suivante. La requête ping mentionnée plus haut permet de savoir si un client est toujours présent. C'est un « acte de présence » que le client devra faire au bout d'un certain temps s'il ne veut pas être déconnecté. Ceci sera décrit plus loin.

Données sauvegardées

Les informations concernant les clients sont sauvegardées à l'issue de la phase de connexion dans un fichier texte (comme décrit dans la section « Persistance des données entre appels de services »). Ce fichier est nommé par défaut *clients*. Retrouver les clients connectés consiste à lire ce fichier, mettre à jour des informations les concernant consiste à y écrire. Voici comment les informations sont sauvegardées :

Key	Usage	User	Connection's datetime	Echo's datetime
163208008	Refactoring	Mario	1082562176	1082562176
16479493	WYSIWYG	Damien	1082562213	1082562213

Le séparateur horizontal est le caractère espace. Voici un exemple :

```
163208008 Refactoring Mario 1082562176 1082562176
16479493 WYSIWYG Damien 1082562213 1082562213
```

La date d' « echo » mentionnée est le dernier moment où un utilisateur a fait un *ping*. Si cela fait trop longtemps, l'utilisateur est déconnecté au prochain chargement du fichier. Dire si cela fait trop longtemps ou pas se base sur un délai de garde (configurable).

Ping

Il s'agit d'un service qu'un client doit appeler au bout d'un certain temps pour indiquer qu'il est toujours là (et qu'il n'a pas planté par exemple). Ce ping, outre le fait qu'il évite à l'utilisateur de se faire déconnecter, propose aussi d'autres services :

- Actualisation des verrous déposés
- Actualisation du verrou administratif
- Mise à jour de l'adresse IP de l'utilisateur pour le service de notifications

Actualiser les verrous lors d'un ping est proposé mais n'est pas imposé, au même titre que le service de notifications. Tout ceci est configurable.

Déconnexion

A noter que la déconnexion, outre le fait qu'elle supprime un client des utilisateurs, supprime automatiquement les verrous posés, verrou administratif du client, si posé, compris.

Module

L'implémentation du mécanisme de connexions se trouve dans le module Service::Connection. Cinq services accessibles à distance y sont proposés:

- *connect* pour se connecter
- *disconnect* pour se déconnecter
- *ping* pour faire acte de présence
- *getUsers* pour obtenir la liste des utilisateurs connectés
- *setAdminLock* pour poser un verrou administratif (cf. section consacrée)

Gestion des verrous

Besoins

Les verrous TWiki sont indispensables pour gérer la concurrence lors de l'édition de topics et lors de la réalisation d'opérations de déplacement, renommage ou de suppression. Ils se présentent sous la forme de fichiers portant le même nom que le topic et ayant comme extension *.lock*. Les verrous sont gérés dans TWiki à partir du *WikiName* du client, qui est sauvegardé dans le verrou.

Un problème posé est qu'un même *WikiName* peut être partagé par plusieurs utilisateurs, par exemple guest si l'utilisateur ne s'est pas connecté. Deux utilisateurs invités peuvent mutuellement écraser leurs verrous par exemple. Autant cela n'est pas vraiment problématique dans le cadre de l'usage normal de TWiki, autant cela l'est d'avantage pour nos services. Nous ne devons pas permettre à un administrateur de déplacer un topic alors qu'il est cours d'édition WYSIWYG par exemple. Nos services doivent marcher sur des TWiki configurés sans authentification (clients tous guest), et doivent permettre plusieurs connexions avec un même login pour l'usage simultané de plusieurs logiciels clients du service.

Nous avons à notre disposition avec le système de connexions d'un identifiant plus « fin » que le login: la clef de connexion. L'objectif de cette section est de montrer comment nous gérons les verrous TWiki à partir de cette clef.

Principe

Nous avons pensé à deux systèmes possibles afin de répondre à ce besoin:

- Surcharger les verrous de TWiki en y mettant la clef comme information supplémentaire
- Maintenir une table d'associations verrou/clef

La première idée a été écartée à cause d'incompatibilités que cela aurait engendré dans TWiki. Après analyses, cela aurait posé des problèmes et il aurait été nécessaire de proposer un patch. La deuxième solution a été choisie, avec une différence:

Un chemin identifiant un verrou ne nous parait pas suffisant pour dire qu'un verrou appartient bien à un utilisateur. Ce verrou a très bien pu être posé par un autre utilisateur TWiki ayant le même *WikiName* (fortement possible avec *guest*), et si l'association est toujours présente, ce verrou appartiendra à l'utilisateur enregistré alors que ce n'est pas le cas. Nous avons ajouté la date de modification du verrou au chemin comme identifiant.

Représentation des données

Les associations sont sauvegardées dans un fichier texte comme pour les connexions. Ce fichier est nommé *locks* par défaut. Retrouver les associations consiste à lire ce fichier et en mettre à jour à l'écrire. Voici comment ces associations sont sauvegardées:

Key	Modification date	Filename
163208008	1100848409	/twiki/data/Sandbox/PageDeTest1.lock
16479493	1100848410	/twiki/data/Sandbox/PageDeTest2.lock

Voici un exemple de contenu de fichier:

```
163208008 1100848409 /twiki/data/Sandbox/PageDeTest1.lock
16479493 1100848410 /twiki/data/Sandbox/PageDeTest2.lock
```

Module

L'implémentation de ce système se trouve dans le module *Service::Locks*.

Verrou administratif ou comment gérer la concurrence entre applications de refactoring

Besoins

L'outil de refactoring a été prévu pour fonctionner en mode « déconnecté ». Les opérations peuvent être faites localement avant de les valider sur TWiki par service Web. Un utilisateur peut donc avoir une liste de modifications à valider ... en même temps qu'un autre. Si les modifications destinées à être apportés par chacun n'interfèrent pas, il n'y a aucun problème. Mais si un utilisateur veut par exemple déplacer un topic qu'un autre veut supprimer, il y a conflit. Si le premier utilisateur valide en premier, le deuxième utilisateur va voir sa liste de modifications en attente irréalisable. Il nous faut donc un moyen d'empêcher cela.

Principe

Nous avons choisi de proposer un mécanisme de prévention (il n'est pas imposé, le service peut fonctionner sans). Avec ceci, un administrateur doit prévenir les autres qu'il va réaliser des opérations de refactoring s'il veut les effectuer. Ceci se fait par le biais d'un « verrou administratif ». Il s'agit d'un verrou - fichier (comme les autres verrous dont nous avons parlé), qui contient la clef de l'utilisateur et sa date de création. Sa gestion est similaire à celle des verrous TWiki. Si le service est configuré pour le gérer, les opérations de refactoring échoueront s'il n'est pas posé.

Module

Le service *setAdminLock* permettant de poser un verrou administratif est implémenté dans le module *Service::Connection*. Tout ce qui nécessaire à sa gestion est implémenté dans *Service::AdminLock*.

Notifications

Il s'agit d'une amélioration qui n'était pas envisagée lors de la rédaction de notre cahier des charges. Les notifications sont des messages émis lorsqu'un événement se produit

afin d'informer les autres utilisateurs. Elles sont gérées uniquement au sein du service, pas dans TWiki en général car cela aurait nécessité de modifier son code (donc de faire un patch). Dans ce sens, ce système n'est pas terminé mais s'avère être un projet ouvert à d'autres développeurs.

Principe

Il faut savoir qu'à la date de réalisation de ce projet, les callbacks ne sont pas supportés par les services Web en standard. Seules quelques implémentations sont allées au-delà du standard afin de les proposer, comme .NET ou Java par le biais de JMS. Nous avons opté pour le concept d'abonnement/notifications, comme pour la gestion des événements en Java. L'abonnement est réalisé par service Web « habituel » par *SOAP::Lite*. Les notifications sont gérées aussi par service Web mais ce ne sont plus des messages SOAP gérés par *SOAP::Lite* mais des envois de données XML. Elles sont programmées à l'échelle des sockets et des messages.

Services d'abonnement

Deux services Web *subscribe* et *removeSubscription* permettent de s'abonner et de résilier un abonnement à des notifications. On demande en outre lors de la souscription le port vers lequel envoyer le message au client. Ces services consistent à ajouter ou enlever un utilisateur d'une liste de contacts, sauvegardés dans un fichier comme pour les connexions. Chaque entrée de ce fichier indique une clef de connexion, une adresse IP et un port sur lequel on doit établir une connexion pour envoyer le message.

Envoi de messages

Si le service est configuré pour, un message est envoyé à chaque opération de :

- refactoring
- (dé)verrouillage de verrou administratif
- modification de topic
- mise à jour des liens d'un topic
- (dé)connexion

Ces envois consistent à produire un événement, à se connecter chez chaque client abonné (en dehors de l'émetteur du message), et à envoyer un message XML contenant l'événement sérialisé.

Message

Un message contient quatre « éléments » :

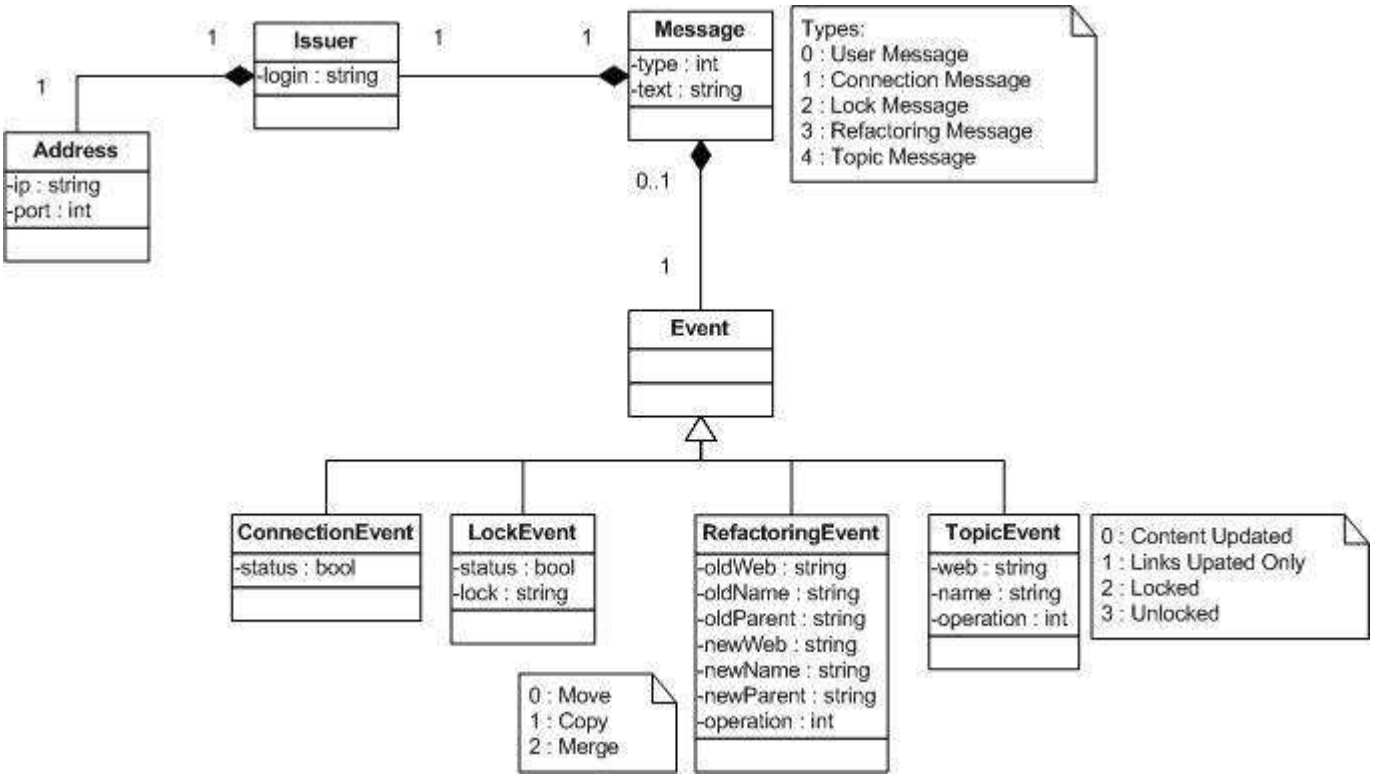
- un type
- une origine
- un événement (facultatif)
- un texte

Le type indique quel est le type du message, s'il concerne par exemple les connexions ou les opérations de refactoring. L'origine indique l'utilisateur émetteur du message, c'est-à-dire son login et s'il est inscrit au service, son adresse IP et le port sur lequel il écoute (afin éventuellement de permettre à quelqu'un qui reçoit le message de le contacter). Le texte est le contenu même du message. Un événement peut aussi y être

attaché afin de préciser ce qu'il s'est passé. Nous avons développé quatre objets Perl différents pour les modéliser :

- *ConnectionEvent* : événements de connexion
- *LockEvent* : événement lié à un (dé)verrouillage « général » (pas sur topic). Seul les opérations sur verrou administratif produisent ce type d'événement actuellement.
- *RefactoringEvent* : événement de refactoring sur topic
- *TopicEvent* : événement sur topic, c'est-à-dire modification, (dé)verrouillage ou mise à jour des liens

Voici un diagramme de classes de type UML spécifiant ce qu'est un message:



Sérialisation XML

Un message (ainsi que l'événement attaché) est sérialisé en XML afin d'être transmit au client. La sérialisation se fait par le biais de la librairie Perl *XML::TreeBuilder*. Le message suit un XML Schéma précis qui est disponible dans les ressources du projet: *messages.xsd*.

Modules

Le module *Service::Notification* propose les services d'abonnement (*subscribe*) et de résiliation d'abonnement (*removeSubscription*), ainsi que la sérialisation du message même. Un répertoire *Notification* contient les objets (modules particuliers) Perl modélisant les événements.

Package client TWikiListener

Un package Java a été développé afin de faciliter l'interception de messages côté client. Ce package contient deux types de classes:

Messages

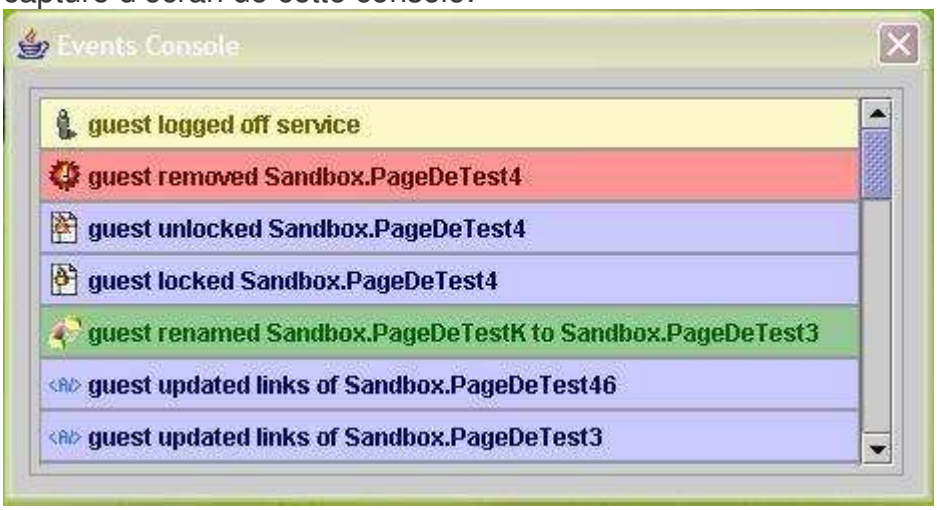
Des classes « mappées » sur les structures XML des messages attendus. Elles ont été générées à l'aide d'un « binding compiler » du projet Open Source *Castor*, à partir du XML Schema *messages.xsd*. Ce sont des instances de ces classes qui sont renvoyées au client.

Ecouteur : TWikiListener

Une classe d'objet qu'il faut instancier afin d'écouter le port que l'on a donné à l'abonnement. L'objet instancié fonctionnera selon le design-pattern observers/observable. A chaque interception de message, il enverra à ses observers le message reçu. La désérialisation (unmarshalling) se fait avec l'aide des bibliothèques de *Castor* et de *Xerces*. Ces librairies sont requises pour pouvoir utiliser *TWikiListener*.

Console Java

Une « console » Java a été développée afin de montrer l'utilisation du package *TWikiListener* et du service de notification développé. L'abonnement n'est pas géré par ce programme. Si cette console doit être utilisée dans une application, c'est cette dernière qui doit faire l'appel au service. Pour tester la console indépendamment, il faut passer par un programme tel que *Capescience NetTool* pour envoyer manuellement les requêtes SOAP. Certaines sont disponibles dans les ressources du projet. Voici une capture d'écran de cette console:



Seules les sources sont disponibles dans les ressources du projet. Cette console n'est pas destinée à être une application standalone mais plutôt à être incluse dans d'autres programmes telles que l'outil de refactoring.

Aspects fonctionnels

Hiérarchie des topics

Récupération des propriétés d'un topic

Ce service permet de récupérer les propriétés d'un topic à partir de son chemin. Ses propriétés sont récupérées directement dans les données meta du topic et sont envoyées par l'intermédiaire d'un tableau de chaînes de caractères. Les valeurs représentent respectivement dans l'ordre la taille, le nom du Web, le chemin, le topic

parent, l'auteur, la date de création, la date de modification, la version, le format et le droit. Pour ce dernier, 0 traduit le fait que l'utilisateur n'a aucun droit d'écriture, 1 qu'il a le droit de modification, 2 pour le droit de renomage et 3 pour le renomage et la modification. Ces droits sont calculés à partir de la clé donnée en paramètre, représentant l'utilisateur au niveau serveur.

Récupération des propriétés d'un Web

Ce service permet de récupérer les propriétés d'un Web, c'est-à-dire sa taille. Cette dernière n'est pas calculée récursivement, c'est-à-dire qu'elle fait la somme de toutes les tailles des topics et ajoute une constante pour chaque Web (elle ne calcule pas la taille des sous webs). Cette fonction utilise la commande *stat* d'Unix.

Représentation de la hiérarchie des topics

Le service *sameTopicParent* est le cœur de l'application de refactoring pour la représentation arborescente des topics. Il prend trois paramètres : le nom du topic parent, le nom du web pour définir d'où commence la recherche et la clé pour identifier l'utilisateur au niveau serveur. Suivant les paramètres, ce service réagit de façons différentes. Je vais énumérer les cas les plus intéressants :

- Si le nom du topic parent est «» , cette fonction renverra la liste des topics parent présents dans le Web donné en paramètre mais aussi dans ses sous webs. De plus il renverra la liste des topics qui ne sont ni des enfants, ni des parents. Cela à pour but de ne pas tout envoyer au client. Si un topic parent est lui-même enfant, il ne sera visible que lorsqu'on aura déployé son père.
- Si le nom du web est « » alors la recherche se fait dans toute la hiérarchie data de TWiki. En effet, un topic peu avoir son père dans un web qui ne se trouve pas dans sa même hiérarchie. Dans l'exemple ci-dessous, titi peut être le père de toto.
  Web A
  |____Web B___toto.txt
  |____Web C___ titi.txt

- Si le nom du web est spécifié, la recherché se faire suivant Nomtopic et Web.Nomtopic

Cette fonction est très gourmande en ressource, par le simple fait qu'elle doit constamment parser tous les topics du serveur pour récupérer les infos dont elle a besoin.

Récupération d'un contenu d'un Web

Ce service permet de connaître la liste des sous webs, topics présent dans le Web donné en paramètre. Il correspond à la commande « ls » du système Unix.

Opérations de refactoring

Toutes les fonctions dédiées aux opérations de refactoring suivantes ont été implémentées dans le module *Service::Refactoring*.

Renommage

Le renommage se fait au sein d'un même web. Le service *renameTopic* demande en paramètre quel topic renommer et quel nouveau nom lui donner. Après différents tests (existence, verrouillage, autorisations, verrou administratif, nouveau nom WikiWord etc.), il fait appel à la fonction générale *TWiki::Store::rename* de TWiki, qui lui sert au renommage mais aussi aux déplacements. La mise à jour des liens est proposée en option.

Déplacement

On peut distinguer deux types de déplacements dans TWiki. Le premier est un déplacement réel qui consiste à changer les fichiers du topic d'emplacement dans le file system. Par « les fichiers », nous voulons dire le fichier texte qui constitue le topic, sa pile RCS pour gérer les versions et éventuellement ses attachements. Le deuxième déplacement, qui n'en est pas un, consiste simplement à changer une référence dans les « meta-informations » du topic. Le service de déplacement moveTopic demande en arguments le web et le nom du topic à déplacer, le web de destination et le topic parent auquel le rattacher. Le déplacement inter-webs sera lancé si le web source est différent du web destination. Les deux types de déplacement peuvent être combinées en fonction des arguments. La mise à jour des liens est proposée en option.

Déplacement inter-webs

Le déplacement inter-webs s'apparente à un renommage étant donné que la fonction « interne » à TWiki pour faire cela est la même (*TWiki::Store::rename*). Le nouveau nom est identique au précédent, mais les web source et destination sont différents.

Changement de parent

Le changement de parent consiste à changer un lien placé dans l'entête du topic. TWiki nous offre la possibilité d'ouvrir un topic et de récupérer de manière distincte ses meta-informations (modélisées par une table de hachage) et son contenu. Mettre à jour une meta-information consiste à changer une entrée de la table et à sauvegarder le topic.

Suppression

Plusieurs types de suppression sont proposés par le service *removeTopic*. La suppression « classique » de TWiki consiste à déplacer un topic dans le web « Trash ». Si un topic dans ce web a le même nom, la suppression échoue. Mais TWiki propose de donner un nouveau nom pour éviter cela. Quatre options de suppression sont proposées, certaines peuvent être désactivées depuis le fichier de configuration:

- Déplacer un topic dans le web « Trash » en renommant avec un nom demandé en paramètre
- Déplacer un topic dans le web « Trash » et générer automatiquement un nouveau nom si le topic existe déjà.
- Déplacer un topic dans le web « Trash » et supprimer celui qui y est éventuellement présent.
- Supprimer réellement un topic.

Les attachements sont gérés de manière « similaire », c'est-à-dire qu'une suppression « réelle » impliquera la suppression des attachements et un déplacement, un déplacement des attachements.

Fusion

La fusion est une concaténation de topics et/ou des attachements. Le service *mergeTopics* demande en arguments le topic cible et le second topic à partir duquel faire la fusion. Géré les attachements est proposé en option. Au cas où ce serait demandé et dans le cas peu probable ou deux attachements auraient le même nom sur les topics, un renommage est automatiquement fait. Ce service propose en outre de faire de la fusion un rapport indiquant explicitement à quel topic appartient un contenu.

Copie

La copie de topics peut s'apparenter à un déplacement dans l'idée. Concernant l'implémentation, ils sont très différents car les fonctions proposées par TWiki ne concernent que les déplacements, autant en ce qui concerne les topics eux-mêmes que les attachements. Là où moveTopic réutilisait des fonctions de haut niveau de TWiki, *copyTopic* gère les topics à un niveau plus bas (à l'échelle de la création, de la recopie de contenu et de la sauvegarde) et les attachements au niveau quasiment le plus bas (cf. fonction *Service::Topics::copyAttachmentFile*).

Mise à jour des liens

La mise à jour des liens se déroule en deux temps: rechercher les topics concernés par la mise à jour et la réaliser. La recherche se fait via la commande *egrep*, à qui on passe une expression régulière formatée selon le web et le nom du topic qui a changé. Tous les topics ayant un lien, que ce soit dans le contenu ou dans les « meta-informations » (donc les topics enfants) seront retournés. La recherche va se faire dans des emplacements particuliers, en fonction des paramètres passés. Cette première phase retournera un tableau de locations de topics (syntaxe *web.topic*). La procédure de mise à jour va prendre ce tableau en paramètre. Elle va tenter d'ouvrir chaque topic (s'il est verrouillé: échec et on passe au suivant), récupérer les « meta-informations » et mettre à jour le lien vers le topic parent si c'est nécessaire, effectuer du pattern-matching via expression régulière sur le contenu et le sauvegarder.

Rapatriement de topics vers l'éditeur

Création de topic

Le service *getTopic* permet de générer un objet *topic* à partir du topic donné en paramètre (par l'intermédiaire de son chemin). Il va directement chercher les informations écrites en dur dans les données meta du topic. Avant de renvoyer le résultat, les données sont formatées pour traduire les caractères spéciaux en Unicode afin d'être validées par tout parser XML. La clé donnée en deuxième paramètre permet de déterminer si l'utilisateur possède les droits nécessaire et si le topic n'a pas déjà de verrou administratif. (Un autre utilisateur fait des modifications dessus).

Création d'attachement

Le service *getAttach* permet de générer un objet *Attachment* à partir du topic et du nom de l'attachement donné en paramètre. Les propriétés de cet objet sont récupérées grâce aux attributs meta de l'attachement écrit en dur dans le topic. Il n'y a pas de vérification des droits, car on suppose que pour qu'un client possède le topic et le nom de l'attachement, il a du forcement appelé un service lui demandant sa clé, et donc vérifié ses droits. Il n'est donc pas nécessaire de vérifier de nouveau.

Récupération des attachements

Le service *getListAttach* permet de récupérer la liste des attachements pour un topic donné. Pour ce faire, il récupère la liste des noms de fichiers directement dans les données meta écrite en dur dans le topic. Le clé donnée en deuxième paramètre permet de vérifier si l'utilisateur possède bien le droit de lecture sur ce topic.

Listing

Le service *getAllTopic* permet de renvoyer tous les topics précèdent par leur web (web.topic) visible par l'utilisateur (identifié par la clé donnée en paramètre).

Sauvegarde de topics

Le service *setTopic* prend en argument un objet *topic*. Celui-ci dispose d'un nom, d'une location (le web), de propriétés, d'un contenu textuel et d'un tableau d'objet *attachment*; un objet *attachment* disposant lui-même de propriétés et d'un contenu, binaire celui-là. Côté client Java, les attachements et les topics sont réellement modélisés par des instances de classes. En XML, ce sont des structures complexes. Côté Perl, ils sont traduis en tables de hachage automatiquement. La sauvegarde de topic en elle-même est très simple étant donné que des fonctions de haut niveau sont disponibles dans TWiki. La difficulté réside dans l'upload d'attachements, gérés dans TWiki par le serveur Apache. Avant de parler de ceci, voici le « protocole » sur lequel se base la sauvegarde de topics à la réception :

- Un attachement du topic contient un contenu binaire : on ajoute/mets à jour l'attachement.
- Un attachement du topic est retourné mais ne contient pas de données binaires : cela ne sert que s'il existe déjà dans le topic sinon on ne fait rien. Si l'option *doKeep* est à false, on le garde. Sinon, si *doKeep* est à *vrai*, on ne fait rien.
- Le topic ne contient pas d'attachements : si *doKeep* est à *vrai*, on ne change rien, sinon on les supprime tous.

L'option *doKeep* à *false* permet de faire un comparaison sur les attachements présents et ceux retournés. Cela permet de les gérer. A l'heure où ces lignes sont écrites, *doKeep* est automatiquement mis à vrai car le rapatriement de topics ne gère pas les attachements. Concernant l'upload, les données binaires sont stockées en base64 dans les messages SOAP. A la réception, on récupère ces données et les écrits dans un fichier temporaire portant un nom lié à la clef de connexion (upload_*clef*). Ceci permet d'éviter des problèmes de concurrence entre uploads simultanés. Une fois ce fichier temporaire écrit, une fonction de haut niveau de TWiki permet de faire la sauvegarde à partir de celui-ci.

Génération de stubs client

Comme toute technologie liée aux applications réparties, les services Web requièrent des stubs afin que les clients les invoquent. Nous utilisons une procédure de génération automatique de stubs à partir d'un fichier de description (comme RPCGEN avec un fichier .x). Le fichier de description est une WSDL qui décrit les services Web en XML. Le générateur est *WSDL2Java* du framework *Apache WSIF*. A l'issu de cette génération, nous obtenons des stubs cablés vers l'adresse du serveur SOAP décrit dans la WSDL. Afin d'avoir des stubs dont l'adresse est paramétrable, une petite modification doit être

faite dans le fichier *ServiceLocator.java*. Au lieu d'avoir un attribut adresse *final*, nous devons faire un constructeur qui la prenne en paramètre. L'application, au lancement, devra la lui passer. Dans notre cas, nous la lui passons via un argument Java Web Start.

Annexe 5 : TWiki WYSIWYG Editor - Manuel de maintenance

Introduction

On dispose :

- du traducteur TWiki ML <-> HTML de F. Luddeni
- de l'API Java qui met en oeuvre, au travers de *SWING*, le modèle MVC (modèle vue contrôleur). Dans notre cas, les classes intervenant sont :
- **JEditorPane** pour la vue
- **HTMLEditorKit** pour le contrôleur
- **HTMLDocument** pour le modèle

Un document TWiki peut être traduit en un document HTML, cependant ce document HTML ne comporte qu'un sous-ensemble limité de balises HTML.

L'idée du programme est de réaliser un éditeur HTML en Java dont le champ d'action correspond au **sous-ensemble HTML en bijection avec les éléments de syntaxe TWiki**. On trouvera plus de détail sur ce sous-ensemble dans les annexes du cahier des charges. L'éditeur WYSIWYG est complètement dépendant du traducteur aussi bien dans le sens entrant (TWiki ML -> HTML) que sortant (HTML -> TWiki ML). On s'efforcera donc de ne pas y toucher pour garantir l'évolutivité de l'application. Par rapport à un éditeur HTML classique, Un éditeur dédié à TWiki présente les intérêts suivants :

- Ajout de fonctionnalités dynamiques (Insertion de Wiki Word?, de pièces jointes, ...)
- Gestion spécifique d'élément de syntaxe TWiki : par exemple, réalisé un mapping entre smileys TWiki et image gif correspondante dans l'éditeur HTML
- Evolution des services dynamiques en relation avec le web service

Architecture de l'application

Packages

- twe.actions.misc : contient les actions diverses telles que l'ouverture, la sauvegarde de documents locaux, la complétion,... .
- twe.actions.twiki : contient les actions d'édition de texte (Titres, Gras, Insertion de tableau, ...).
- twe.gui : contient les élements d'interface utilisateur.
- twe.translator : contient le traducteur HTML <-> TWiki ML. Il est préférable de ne pas faire de modifications sur ce traducteur pour rendre l'éditeur indépendant d'éventuelles évolutions du traducteur.
- twe.util contient diverses classes utiles à l'application. Concerne essentiellement HTMLDocument et Element

Fonctionnement

La classe principale est MainFrame. Cette classe est chargée de mettre en place les composants Swing de l'interface et de leur donnée vie au travers d'écouteurs et d'actions. MainFrame dispose d'un ensemble d'actions qui se trouvent dans twe.actions. La plupart de ces actions implémentent TWEAction qui est une interface décrivant une action comme possédant un bouton un contrôleur et une icône. Cela facilite le montage de l'interface graphique.

La plupart des actions se contente d'hériter d'actions déjà existantes dans l'API Java. C'est le cas par exemple des actions pour le copier/coller ou pour le gras et l'italique. D'autres actions comme l'insertion de tableau ou de liste à puces n'existent pas dans l'API et doivent être codée en manipulant le document HTML.

Extensions de l'application

Ajouter une fonctionnalité

C'est très simple, il suffit de :

- créer une classe héritant de Action et implémentant TWEAction
- dans MainFrame, modifier la fonction buildMiscBar() ou buildFormatBar() en instantiant l'action nouvellement crée.

NB : Le code effectif de l'action nouvellement crée se trouve dans la fonction héritée actionPerformed. Une action disposant généralement d'un contrôleur (JEditorPane, MainFrame,...) il n'est pas nécessaire d'utiliser l'objet *évenement* passé en argument de actionPerformed pour récupérer la source de l'action. Il suffit d'utiliser directement la référence au contrôleur.

Modifier le traducteur (externe) TWiki ML <-> HTML

Il suffit de remplacer les classes du traducteur par les nouvelles. Du point de vue de l'éditeur, le traducteur est considéré comme un composant. Le package concerné est twe.translator.

Modifier les stubs d'appel au web service

Tout comme le traducteur, il suffit de remplacer les anciennes classes par les nouvelles. Cependant si la signature des fonctions d'appel au web service change, il faudra également modifier les appels à ces fonctions. Le package concerné est fr.unice.twikiservice.

Annexe 6 : Topic du projet sur `twiki.org`

A TWiki WYSIWYG Editor

The project got registered on BerliOS on 06 Mar 2004,
http://developer.berlios.de/projects/twysiwyg/

We are planning the project. Development will start in May 2004.

Developers : MaximeLamure, MarioDiMiceli, DamienMandrioli, RomainRaugi

-- DamienMandrioli - 11 Mar 2004

ColasNahaboo tells me this is a student project in Nice University (France), where he is an external advisor from ILOG. AFAIK the goal is a full WYSIWYG editor built using Java WebStart? technology. This means it will necessarily be Java1.4 and - I think - require the Java Plugin. However it should be significantly quicker to start up than any of the older applet editors.

Damien, you should look seriously at EKit.

-- CrawfordCurrie - 11 Mar 2004

There are many existing WYSIWYG editors that work with HTML. They can be written as a Java Applet, JavaScript + DHTML, or Flash. One of these editors could be integrated into TWiki but you'd need to convert TWikiML >> HTML >> TWikiML. Since it is challenging to do this roundtrip 100%, one would confuse the revisions of TWiki pages.

I think a better approach is to create an editor that works natively with the TWikiML. If HTML is in a page, it could be shown verbatim, `like this <u>example</u>`. This has the nice side-effect that (a) users are encourraged to use only TWiki ML, and (b) TWikiApplication authors could use the editor to write complex forms.

A good fit for the TWikiMission is an editor that is cross-platform; works across major browsers; does not require a client side install; has low system requirements; and falls back to editing the conventional way for users/systems that do not have the required environment (like PDAs, text browsers, older browsers). You might have different priorities.

Also worth pointing out is Crawford's PowerEditPlugin.

-- PeterThoeny - 12 Mar 2004

See PowerEditPluginDev for an applet editor that works natively on TWikiML and displays formatted text.

-- CrawfordCurrie - 12 Mar 2004

Some background info on this:

We have had many requests for a WYSIWYG editor for TWiki. However, all of these requests come from non-technical users, and to us that means IE on Windows exclusively. So having a cross-browser solution would be nice, but is not required in this context.

I am in a private company, ILOG. I am the intranet architect there, and have close ties with professors in the local Universities, especially Michel Buffa, so we set up some student projects on this.

We had a student project that was divided in 3 parts:

- have server side converters (in java) between TML and HTML, both ways.
- have the HTML edited by client-side standard HTML editors:
    - one team looked at a .NET solution: they found it impossible (no documentation, too complex), so they used the javascript editor htmlarea, very nice (works on IE and mozilla), and cleanly coded
    - one team used a java solution, and choose Ekit, which is an abysmal pile of crap, and did not manage to get anything of value out of it

So, from this project, the offline converter TML<->HTML is the thing that works. The sudent who did it, Fredeerci Luddeni liked it, and should be uploading it here and maintaining it.

Another project will start to try other things, namely use standalone client editors: either directly editing TML syntax (the direction that the students seem to like best), or I would like to try a solution based on a standalone app made out of mozilla editor + XUL.

The technical difficulty is to handle Wiki links, attachments (you want to drag and drop to attach them), %VARS, Plugin syntaxes,... and be able to allow people edit wiki pages with both the editor and the old browser way.

-- ColasNahaboo - 15 Mar 2004

Colas, from what you say you really should look at PowerEditPluginDev. I have frozen further development of it because of lack of interest from the community. It just didn't seem worthwhile pursuing something that raised so little interest when it was posted! Note that the code for PowerEditPlugin in CVS is the code for the multi-window editor.

-- CrawfordCurrie - 15 Mar 2004

MediaWiki WYSIWYG_editor outlines a process for the EpozEditor which sounds promising. Epoz uses just DHTML and Javascript and works in IE and Gecko (Mozilla) based browsers. Output is well formed xml:

"Epoz only allows structural markup (no font tags etc), so this is mainly a str_replace(). With the appropriate configuration tidy will strip the complex things first. This setup combines the advantages of both worlds by providing two views switchable back and forth without reloading the page:

- WYSIWYG view: uses the default css and html editing
- SOURCE view: WIKI source"

It is unclear whether if a wikimedia-epoz editor is under active development or if it is still/just in the planning stage.

-- MattWilkie - 28 Mar 2004

There are some changes on our project. We met Colas Nahaboo Wednesday March 31 and we decided to re-orient the project on something more general than a WYSIWYG editor : an administrative console.

WYSIWYG part would not be excluded, but would be more a basis of an editor, something simple which could be used by others developpers. The main part now would be implementations of a visual representation of TWiki hierarchy (decorated tree highlighting authorizations or dates on which topics were modified) and manipulations it is possible to do on it (like refactoring).

Concerning technologies, the console would be implemented as a standalone Java application and the communication with TWiki established via XML Web services (Perl implementation on server side).

-- RomainRaugi - 01 Apr 2004

I am aware of two WYSIWYG Editors used in the CMS area:

- The Bitflux Editor - The Wysiwyg XML Editor "Bitflux Editor is a browser based Wysiwyg XML Editor ? and that changes everything! You can edit now your content semantically and at the same time display it to your users and editors in its final form." The Apache/Lenya CMS uses Bitflux.

- Kupu Kuppu is the successor of Epoz. It is now an OSCOM-Project. For an introduction see Kupu 1.0.3 released.

-- HansruediHaenni - 05 Apr 2004

http://c2.com/cgi/wiki?WysiwygWikiContest -- CrawfordCurrie - 06 Apr 2004

Our project is running. Main project page : here (in french)

The page of the part concerning wysiwyg editor : here (in french)

Download first version of wisywig editor for test only : Download

Features set will be extended in the next days.

Source code can be found on : http://developer.berlios.de/projects/twysiwyg/

DamienMandrioli - 20 May 2004

Would sombody (maybe someone who is currently studying the language) care to practice their translations skills on these topics? if so, im sure there wouldnt be a problem useing a topic such as TWikiwysiwygEditorEnglishTranslation? page.

-- TravisBarker - 21 May 2004

Because it is a students project (limited in time), we can't do this translation. We deployed the Java Web Start Editor and the Web Service for communications on http://miageprojet.unice.fr/twikitestnice/bin/view/Main/WebHome You can test the editor by clicking on "Edit with WYSIWYG Editor". For authentification, press Cancel and you will be able

to edit the page as guest. To save the topic on server, click the green up arrow. Note that the editor is not finished.

-- DamienMandrioli - 31 May 2004

DamienMandrioli, thanks for the efforts; it is understandable that at times during student projects there is no spare time to provide a manual in a foreign language. However, it would be very helpful to have at least rudimentary installation instructions.

For example, above it says to download the first version of wisywig editor for test only. This download is a jar file. But when experimenting at your test site, the editor is invoked via Java Webstart 1.2 (on Windows, there are two executables invoked, javaw.exe and javaws.exe). Looking at your page, all this happens magically when clicking on the link which does `/twikitestnice/bin/jnlplauncher/Main/WebHome?jnlp=TWEOnFly`. So clearly there is more to testing this application than just downloading a jar file. I then went into the CVS archive, where we find three modules, `proto`, `service`, and `twe`. The latter is the code for the editor. Browsing the service directory I find the `jnlplauncher` script and some SOAP interface, which leads me to believe that these should be installed also. It is unclear whether the `proto` directory is needed.

It would be wonderful if at least there would be a step by step instruction on what someone have to do to install the editor on one's machine. Please keep in mind not everybody understands how to run Java programs and how to connect them to TWiki.

By the way, when pressing `cancel` on the log in one is denied access, so I could not really try the editor at your site.

-- ThomasWeigert - 31 May 2004

The editor is not yet ready to be installed on another TWiki. It's linked for the moment to the site whose Damien gave you the address. Here are some détails on how it works :

- Launching

The editor is in a Jar file but requires a jnlp descriptor to be launched with Web Start. Thanks to it, we can pass arguments to the application (i.e. the web and the name of the topic to edit). The script called "jnlplauncher" is just a way to write dynamic arguments in the jnlp from twiki's templates.

- Communication with TWiki

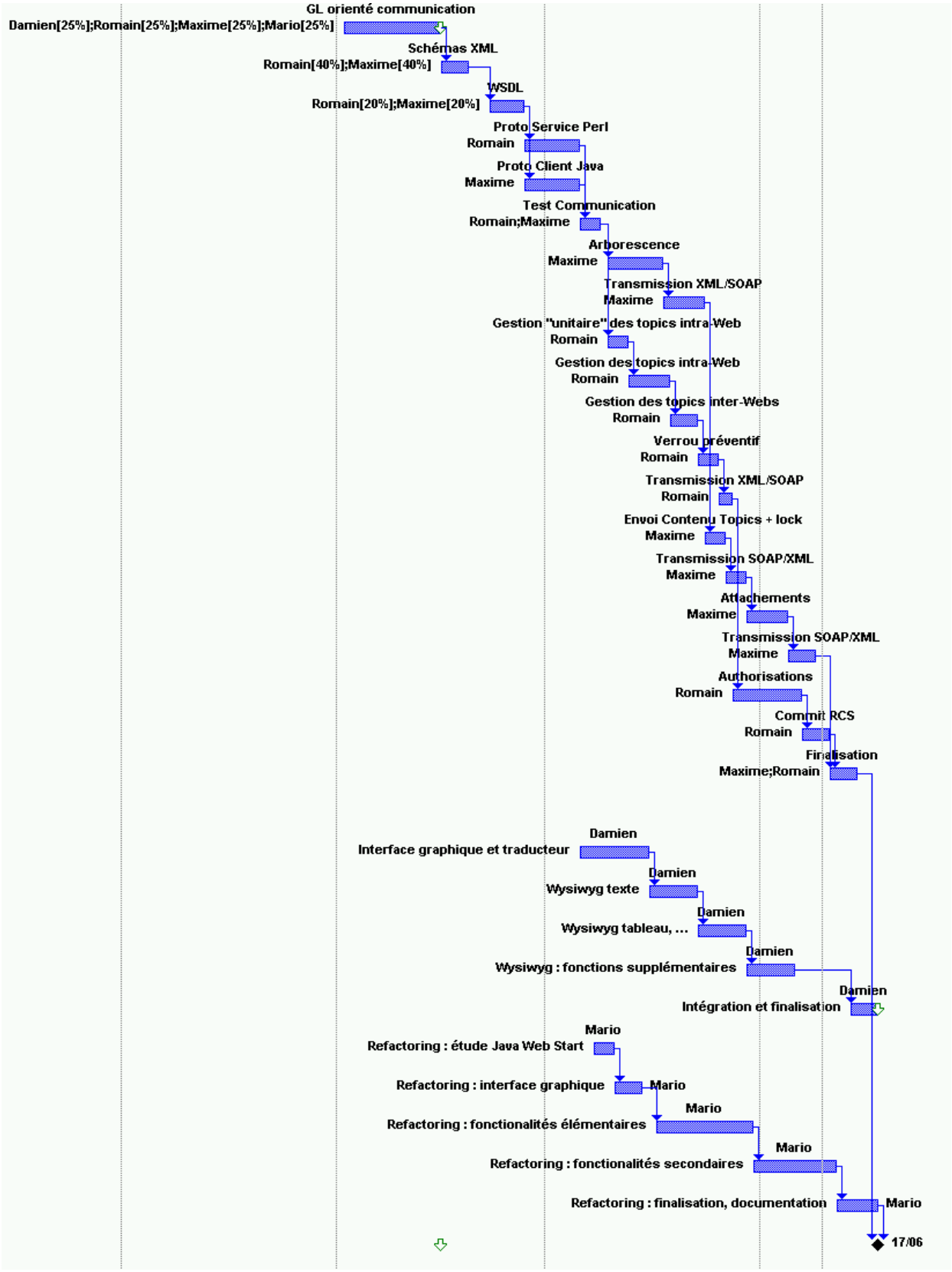
The editor rests on soap interfaces to communicate with TWiki. These soap interfaces are not only for wysiwyg usage but also for a refactoring tool that another member of our project is making. For the wysiwyg tool, communications consists essentially in retrieving and saving a topic.

For the moment, stubs required by these two applications to invoke services are linked to this particular site. We will prepare some installation instructions as soon as this is fixed.

Concerning cvs files, the module called "proto" will soon be removed.

-- RomainRaugi - 01 Jun 2004

Planning Prévisionnel



Planning Effectif