Cahier de reprise
Documentation système

0. Configuration d'un poste de développement
1. Importer le projet
2. Organisation du code
3. Compilation
4. Test en local
5. Deploy to app engine
6. Mettre en place son propre serveur d'application
7. Javadoc

0. Configuration d'un poste de développement

Pour utiliser le Google Web Toolkit, on peut utiliser l'IDE Eclipse disponible au téléchargement à cette adresse :

http://www.eclipse.org/downloads/

La version que nous avons utilisée est "Eclipse Helios (3.6.2)".

Ensuite, il est nécessaire d'installer le plugin GWT pour Eclipse. Il est disponible sur google code :

http://code.google.com/webtoolkit/download.html

Nous avons utilisé la version 1.4.3 du SDK de l'AppEngine, et la version 2.2.0 du SDK GWT (3).

Si la version proposée sur google code n'est pas la bonne, il faut soit :

- la chercher sur le net
- utiliser le dossier d'éclipse fourni (qui contient déjà l'IDE configuré avec les bonnes librairies)

Si vous souhaitez utiliser un serveur SVN, un plugin pour Eclipse est disponible. Il permet d'obtenir le code source de l'application, de créer de nouvelles versions et de continuer le développement. Il est disponible à cette adresse :

http://www.eclipse.org/subversive/

1. Importer le projet

Dans Eclipse, pour créer un projet à partir de l'archive des sources il suffit de faire fichier > importer. Puis choisir général > fichier d'archive.

Pour récupérer le code à partir d'un SVN (faire un Checkout), il suffit de créer un nouveau projet dans Eclipse, en choisissant "Projet depuis SVN". On entre ensuite les coordonnées du serveur SVN et on valide.

2. Organisation du code

Le code est organisé en packages, qui représente les modules, puis dans chaque package, en trois sous-packages qui sont respectivement :
- le code de la partie client qui contient la partie de code qui sera exécuté par le navigateur de l'utilisateur. (Partie compilée en JavaScript)
- le code de la partie serveur qui sera exécuté par le serveur d'application, par une machine virtuelle Java.
- le code partagé entre le client et le serveur. Ces classes sont instanciables autant sur le serveur que sur le client. Les instances peuvent être transférées du client au serveur et inversement à partir d'appels asynchrones.

L'application comprend divers modules qui ont chacun un package dédié :
- Calendrier (gestion des calendriers)
- Chat (gestion du chat)
- Groupe (gestion des groupes)
- Parseur (parseur iCal)
- Post-It (gestion des post-it)
- CCalendar (package principal liant l'interface et comprenant la gestion du login)

3. Compilation

Il est nécessaire, avant de déployer l'application, de compiler en JavaScript la partie du code qui sera exécutée par le navigateur et de compiler en bytecode Java la partie serveur. Les classes partagées entre le serveur et le client sont compilées des deux cotés. Pour ce faire, il suffit (dans Eclipse) de cliquer sur "Compile GWT".

4. Test en local

Pour déployer l'application en local, il faut l'avoir compilé au préalable. Ensuite, clic droit sur le dossier principal, On clique ensuite sur "Run as" et "web application". Par défaut, le serveur est lancé sur le port 8888. On accède donc à son application en local via l'adresse :

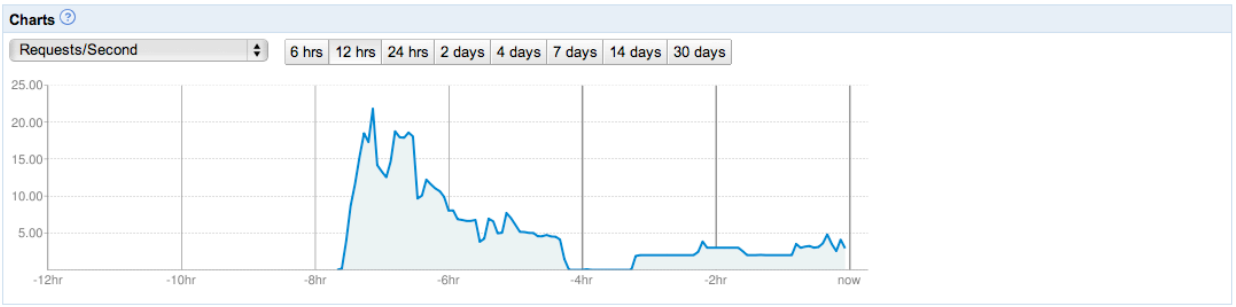
http://localhost:8888

5. Deploy to app engine

Avant le déploiement il est nécessaire de réserver un appspots. Cette opération est gratuite et peut se faire à l'adresse suivante :

https://appengine.google.com/

Pour déployer l'application sur un appspot, le plugin GWT pour Eclipse comprend une mode de déploiement Google. Pour déployer, il suffit de faire une clique droit sur le projet, menu Google, puis "Deploy to l'App engine". Les serveurs d'application de Google comprennent une interface de gestion. Elle permet de visionner les statistiques de l'application, les logs d'erreurs, les quotas, et les tables du datastore. En voici un exemple :



Instances

Number of Instances - Details	Average QPS	Average Latency	Average Memory
4 total	0.496	37.0 ms	73.6 MBytes

Billing Status: Free - Settings — Quotas reset every 24 hours. Next reset: 9 hrs

Resource	Usage		
CPU Time		44%	2.87 of 6.50 CPU hours
Outgoing Bandwidth		21%	0.21 of 1.00 GBytes
Incoming Bandwidth		11%	0.11 of 1.00 GBytes
Total Stored Data		0%	0.00 of 1.00 GBytes
High Replication Data		0%	0.00 of 0.50 GBytes
Recipients Emailed		0%	0 of 2,000
Backend Usage		0%	$0.00 of $0.72

Les détails sur les quotas et limitations du Google AppEngine sont disponibles à cette adresse :

http://code.google.com/appengine/docs/quotas.html

6. Mettre en place son propre serveur d'application

Pour déployer l'application sur un serveur autre que ceux du Google AppEngine, un serveur d'application Jetty (ou Tomcat) avec support JDO.
Installation de Jetty :

http://jetty.codehaus.org/jetty/

Installation de Tomcat :

http://tomcat.apache.org/

Documentation de JDO :

http://db.apache.org/jdo/javadoc.html

7. Javadoc

Javadoc est un outil développé par Sun Microsystems permettant de créer une documentation d'API en format HTML depuis les commentaires présents dans un code source en Java. Javadoc est le standard industriel pour la documentation des classes Java. Nous avons formatés nos commentaires dans le code pour pouvoir être exploité par un outil de génération automatique de javadoc. Le produit de cette génération est fourni avec les fichiers sources.