De Java à GWT

David Roche

version 0.1 mai 2011

Notes de l'auteur :
Ce document est à l'origine destiné aux élèves du lycée G Fichet de Bonneville (Haute-Savoie) ayant choisi de suivre l'enseignement d'informatique proposé en seconde et en première. Il fait suite à deux autres documents : "Initiation à la programmation orientée objet avec Alice"[1] et "D'Alice à Java"[2].
Il n'a pas la prétention de faire des élèves des spécialistes du développement sous GWT mais uniquement de leur fournir des bases leur permettant de développer de petites applications (RIA). Si à la fin de la lecture de ce document certains élèves sont pris d'une "irrésistible" envie de se procurer des ouvrages beaucoup plus complets (mais aussi peut-être un peu plus complexes pour une première approche !), j'aurais atteint mon objectif.
Pour l'instant ce document aborde uniquement le côté "client" du développement. Le côté "serveur" pose quelques problèmes d'infrastructure pour une utilisation pédagogique (les serveurs de type Tomcat sont payants et les serveurs Google App Engine n'acceptent pas les bases de données de type MySQL). Malgré cela, la faisabilité d'un document "GWT côté serveur" est en cours d'étude.

N'hésitez pas à me contacter : projetalicejava@gmail.com

David Roche
Mai 2011

[1] voir
http://www.epi.asso.fr/revue/articles/a1006d.htm et http://www.animanum.com/index.php?page=initiations&rub=alice

[2] voir
http://www.epi.asso.fr/revue/articles/a1009b.htm

Chapitre I
Introduction

Avant d'entrer dans le vif du sujet, il est très important d'avoir quelques connaissances de base sur "qu'est-ce que le web et internet", "qu'est-ce qu'un client, qu'est-ce qu'un serveur». Il existe déjà sur internet d'excellents articles qui traitent du sujet, et comme j'essaye d'éviter "de réinventer en permanence la roue", je vous propose ici un cours de l'excellentissime site du zéro écrit par le non moins excellentissime Herby Cyrille alias Cysboy (ce cours est l'introduction d'un cours sur Java EE disponible sur http://www.siteduzero.com/tutoriel-3-112219-apprenez-a-creer-des-applications-web-dynamiques-avec-jee.html).

(début de l'introduction de Cysboy)

Internet : qui ? quoi ? qu'est-ce ?

Je ne vais pas vous faire un historique sur la naissance du web tel que nous le connaissons maintenant, je vais juste vous rappeler le fonctionnement de celui-ci.
Cependant, si certaines personnes souhaitent tout de même en savoir plus sur l'histoire d'Internet, elles peuvent suivre ce lien.

Pour faire court, ne confondez pas Internet avec le web !

Internet est un assemblage de multiples réseaux, tous connectés entre eux. Cet amas de câbles, de fibres optiques... de matériels, pour faire simple, constitue Internet, aussi appelé "le réseau des réseaux".

Le Web est un système de fichiers présent sur des machines (serveurs) transitant par un protocole particulier, consultable grâce à des navigateurs web et fonctionnant SUR Internet ! Le web est donc un système de fichiers que toute personne possédant un ordinateur (ou un téléphone, maintenant...)

connecté à Internet peut consulter.

En fait, consulter les fichiers présents sur le web est chose courante, surtout pour vous !
Eh oui ! Surfer sur le web, aller sur le Site du Zéro, consulter vos mails chez votre FAI... tout ceci est en fait de la consultation de fichiers présents sur Internet.

Vous n'êtes pas sans savoir que, dans la majeure partie des cas, on surfe sur le web avec un navigateur tel que Firefox, Internet Explorer, Safari... Ne vous êtes-vous jamais demandé comment les navigateurs savent aller au bon endroit ? Comme, par exemple, aller sur le SdZ (Site du Zéro) ?

Pour ceux qui ne le sauraient pas, tout ordinateur actuel possède une adresse sur un réseau : son adresse IP.
C'est grâce à cette adresse qu'un ordinateur, ou un serveur peuvent s'identifier sur un réseau. Voyez ça comme sa carte d'identité.
Par exemple, chez moi, je suis connecté à ma box (fournie par mon FAI) qui me donne accès à Internet.
Sur Internet, cette box a une adresse qui lui est propre et celle-ci ressemble à quelque chose comme ça "242.231.15.123" : on appelle ces adresses des "adresses IP".

Lorsque vous demandez une page web à votre navigateur, vous lui demandez, de façon tacite, d'aller chercher ce qui se trouve à l'adresse demandée !

Partez du principe que toute adresse de site internet pointe vers un serveur (ou plusieurs) qui a une adresse. Par exemple, taper "http://www.google.fr" dans votre navigateur revient à saisir "http://74.125.19.147" (adresse d'un serveur Google sur Internet) : essayez, vous verrez !
Vous êtes d'accord sur le fait que cette suite de nombres n'est pas des plus faciles à retenir...
Il est bien plus simple de mémoriser *google.fr*.
Je ne m'éterniserai pas sur le sujet, mais sachez qu'il y a une machine qui fait le lien entre les adresses de serveurs (suite de nombres) et les adresses littérales (google.fr) : les DNS. Voyez ces machines comme de gigantesques annuaires téléphoniques, mais pour les sites internet.

Et qu'est-ce que c'est que le "http://" ?

Si vous relisez bien ce que j'ai dit plus haut, vous devez voir que nous avons vu qu'avec l'URL que vous renseignez, vous spécifiez une machine à interroger, donc des fichiers à lire, il ne nous manque plus que le protocole.
Ici, il s'agit du protocole http.
C'est grâce à ce protocole que le navigateur envoie des "*requêtes*" (nous y reviendrons) aux serveurs que vous sollicitez. Il en existe d'autres comme le FTP, le SMTP...
Inutile de nous appesantir sur le sujet (c'est un tuto de programmation, pas de réseau, non, mais)...

Au final, une URL peut se décomposer comme suit :

Nous avons vu que, lorsque vous saisissez une URL dans votre navigateur, que vous validez cette dernière, votre navigateur envoie une "requête" au serveur concerné afin qu'il nous renvoie une page web.
Tout d'abord, on nomme vulgairement l'échange de données entre votre navigateur et le serveur qui fournit les pages web un échange client / serveur.
Le client représente votre navigateur et le serveur... Enfin, vous avez deviné.

Le moment est venu de vous en apprendre un peu plus.
Voici ce qu'il se passe :

- le client émet une requête http vers le serveur ciblé ;
- le serveur reçoit les éléments de la requête ;
- celui-ci les interprète ;
- il renvoie la page demandée en émettant une réponse http ;
- le client reçoit la page au format HTML ;
- le client affiche la page.

Nous pouvons résumer ce qu'il se passe avec ce schéma :

(fin de l'introduction de Cysboy)

Revenons sur "le client reçoit la page au format HTML":

Le HTML

HTML est un langage. A l'origine, c'est le seul langage compris par les navigateurs (aujourd'hui, il faut ajouter le JavaScript, nous verrons cela un peu plus loin).

Il est très important d'avoir quelques notions de HTML avant de pouvoir continuer, ici aussi, il existe un excellent cours sur le sujet, toujours sur le site du zéro; avant de poursuivre, allez jeter un oeil (enfin un peu plus qu'un oeil). Pendant que vous y êtes, étudiez aussi les fichiers CSS.

cours sur le XHTML (on considerera que XHTML=HTML) de M@teo21 (créateur du site du zéro)

http://www.siteduzero.com/tutoriel-3-13666-apprenez-a-creer-votre-site-web.html?variant=1

les sites web statiques et dynamiques

Un site web est dit "statique" quand il est uniquement constitué de fichiers textes contenant du HTML. Un site web statique est très largement suffisant si le but du concepteur est uniquement de fournir des informations à ses lecteurs.

En revanche si vous recherchez une certaine interaction entre vos lecteurs et votre site (par exemple, l'utilisateur doit entrer son adresse pour obtenir des informations sur les restaurants de sa région), il vous faudra alors concevoir un site dynamique.

Dans le cas d'un site dynamique, les pages HTML ne sont pas écrites à l'avance, c'est le serveur qui sera chargé de "créer" les pages HTML en réponse à une demande du client. Si vous habitez en Haute-Savoie, après avoir indiqué votre adresse, le serveur créera une page HTML qui comportera uniquement les restaurants Haut-Savoyard. Un autre exemple, encore plus simple : imaginer un site qui, en page d'accueil, vous demande votre prénom (toto). Une fois votre prénom rentré, une deuxième page avec le texte : Bonjour "votre prénom" (Bonjour toto) s'affiche. Ce site, même très simple, est un site dynamique. En effet, la deuxième page n'a pas été conçue par le créateur du site (il ne peut pas connaître votre prénom à l'avance !), mais sera générée par le serveur en temps voulu, plus précisément le serveur créera une page HTML qui affichera "Bonjour toto" (si votre prénom est toto !)

Il existe un certain nombre de langages qui permettent la programmation côté serveur, je n'en citerai que deux :

- PHP (le plus simple à utiliser). Si le sujet vous intéresse je ne saurai trop vous conseiller la lecture de l'excellent tutoriel du site du zéro (eh oui encore !) sur le couple PHP/MySQL, toujours écrit par M@teo21 :

http://www.siteduzero.com/tutoriel-3-197288-introduction-a-php.html#ss_part_1

- java (oui, le java que vous connaissez, enfin presque, on parle plutôt de JEE, Java Entreprise Edition), ici aussi, le site du zéro vous propose un tutoriel de cysboy :
http://www.siteduzero.com/tutoriel-3-112219-apprenez-a-creer-des-applications-web-dynamiques-avec-jee.html

Les sites dynamiques sont donc un vrai progrès et permettent l'interactivité entre l'utilisateur et le site. Mais il existe encore un problème, toute modification de la page (due à une éventuelle manipulation de l'utilisateur), entraînera : l'envoi d'informations du client vers le serveur, la génération d'une nouvelle page par le serveur, l'envoi de cette nouvelle page au client. Tout cela peut prendre du temps et ne va pas favoriser la "fluidité" du site. Heureusement, comme toujours,

une solution existe : faire gérer par le navigateur (c'est-à-dire côté client) une grosse partie du travail (sans avoir besoin d'échanger un gros flux de données avec le serveur).

Et Ajax fut....

La solution se nomme Ajax. Ajax n'est pas un langage, mais un acronyme : Asynchronous Javascript and XML.

Avant d'entrer dans les détails, il faut bien comprendre que le développement de site web en Ajax vous permet de vous affranchir en partie des problèmes exposés ci-dessus : le temps de latence dû au grand nombre d'échanges entre le client et le serveur. Grâce à Ajax, une partie du traitement se fera au niveau du client, la communication client-serveur pourra être réduite au strict minimum. Ajax a permis le développement de RIA (application internet riche) aussi appelée parfois (un peu pompeusement) web 2.0. L'idée est simple : l'utilisateur se trouve devant un site internet, mais il doit avoir l'impression qu'il utilise une application en local (sur son ordinateur). Google doc (http://www.google.com/google-d-s/intl/fr/tour1.html) est l'exemple typique de RIA, l'utilisateur a l'impression de se trouver devant une application de bureautique classique (comme Office de Microsoft ou Open Office la suite bureautique libre), mais, en fait, il se trouve devant un site internet !

Comment cela est-il possible ? C'est tout simplement le navigateur, en local, qui fait le plus gros du travail, les échanges client-serveur sont réduits au minimum, Google doc utilise une technologie de type Ajax.

Alors, qu'est-ce qu'Ajax ?

Tous les navigateurs modernes sont capables de "comprendre", en plus du HTML, un autre langage : le JavaScript (attention, le nom est trompeur, le JavaScript n'a pas grand-chose à voir avec le Java). La partie client (ce qui sera visible sur le navigateur) sera donc programmée en JavaScript.

En Ajax, les données sont échangées par l'intermédiaire de fichier XML. XML signifie eXtensible Markup Language (en français : langage extensible de balisage). XML n'est pas un langage de programmation, il n'y a pas de boucle for, de if, de while,..... .

Il est presque exclusivement utilisé pour stocker (ou transférer d'un programme à un autre) des données (du texte) de façon structurée.
C'est un langage qui utilise des balises (comme le HTML).

Les balises sont ouvrantes, <balise_ouvrante> ou fermantes, </balise_fermante>. Quand vous ouvrez une balise, vous devez à un moment ou un autre la refermer.

Il est possible de mettre du texte entre la balise ouvrante et la balise fermante :
<balise1> Voici le texte inséré </balise1>

Les balises peuvent être imbriquées : on peut insérer un ou plusieurs couples de balises (ouvrante et fermante) entre 2 balises (ouvrante+fermante)

voici un exemple permettant de stocker des informations sur votre cinémathèque :

```
<cinematheque>
	<film>
		<nom>Les deux tours</nom>
		<realisateur>P Jackson</realisateur>
		<annee_sortie>2002</annee_sortie>
	</film>
	<film>
		<nom>Bladerunner</nom>
		<realisateur>R Scott</realisateur>
		<annee_sortie>1982</annee_sortie>
	</film>
</cinematheque>
```

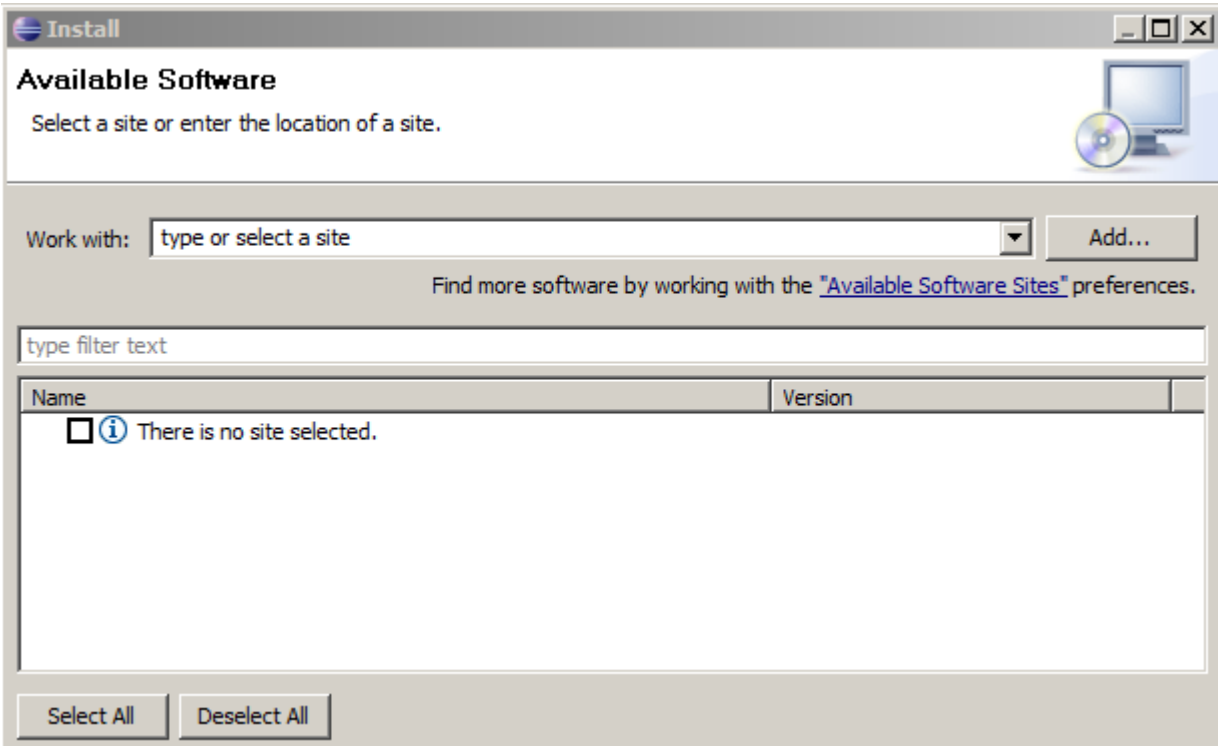
la balise <cinematheque> est appelée "élément racine", la balise <film> est le "premier enfant", etc. Enfin, l'essentiel est de retenir qu'XML va favoriser les échanges de données. Comme d'habitude, voici un tuto sur le site du zéro qui vous permettra d'approfondir la question : http://www.siteduzero.com/tutoriel-3-33440-le-point-sur-xml.html

Dernière partie de l'acronyme Ajax, "Asynchronous" : l'idée est simple, dans un échange synchrone entre le client et le serveur, quand le client (donc le navigateur web) effectue une requête, le client attend la "réponse" du serveur avant de faire quoi que ce soit d'autre (ce qui peut donner l'impression que le navigateur est "bloqué"). Dans un échange asynchrone, le client n'attend pas la réponse du serveur pour poursuivre les tâches en cours, ce qui évite donc le blocage du processus (si la réponse n'arrive jamais !). En Ajax, tous les échanges sont asynchrones, d'où le "Asynchronous".

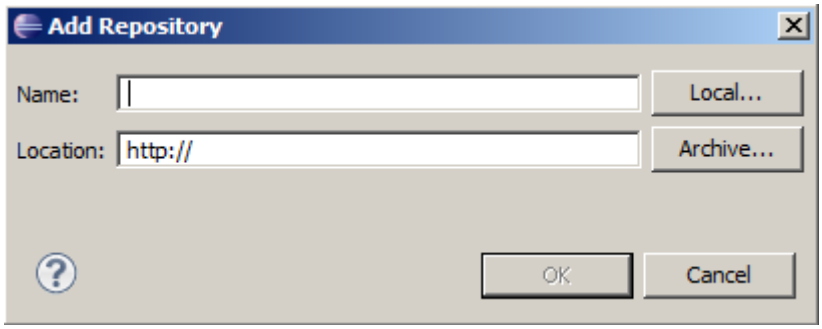
Problème, Javascript est un langage compliqué à maîtriser (pour moi l'utilisation du HttpRequest est.....difficile). Google a donc mis à la disposition des développeurs web GWT (Google Web Toolkit), qui permet de "faire" de l'Ajax en écrivant du Java à la place du Javascript. Toute la partie client est donc principalement programmée en Java (il faudra aussi parfois écrire un peu de HTML, d'où l'intérêt d'avoir quelques notions).

Chapitre II
Installation et premier programme

Pour développer nos projets en GWT, nous allons utiliser l'IDE Eclipse (voir le document "D'Alice à Java" pour plus d'informations sur Eclipse). Avant de passer à l'écriture de notre premier programme, nous allons devoir installer le plugin GWT pour Eclipse.
Dans le menu Help d'Eclipse, choisir Install New Software...
Vous devriez avoir la fenêtre suivante :



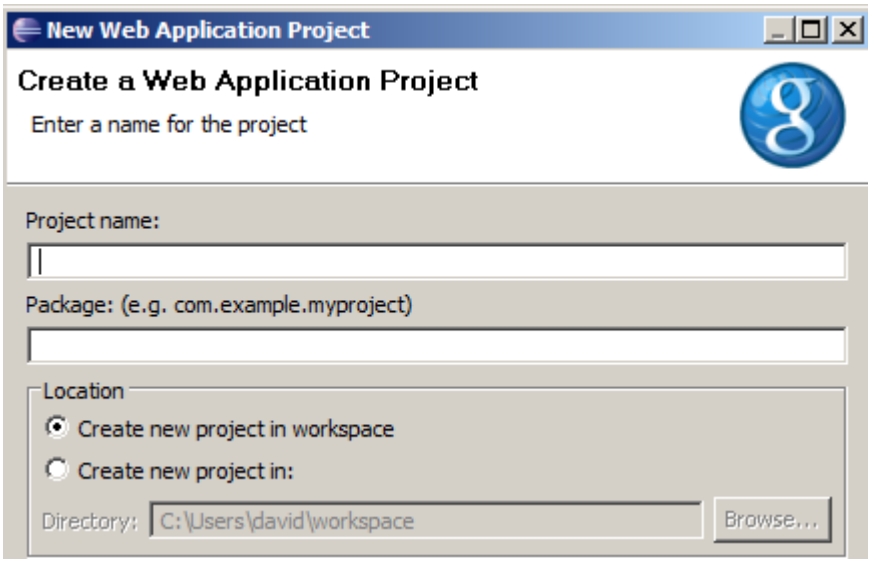
Cliquer sur le bouton Add...



Dans le champ Name, taper GWT
Dans le champ Location, entrer : `http://dl.google.com/eclipse/plugin/3.6`
Attention, cette adresse peut évoluer dans le temps (en fonction de la version d'Eclipse), pour être sûr que la procédure d'installation soit toujours à jour, consulter le site officiel :
http://code.google.com/intl/fr-FR/webtoolkit/usingeclipse.html

Cliquer ensuite sur Select All puis sur Next 2 fois de suite. Accepter les termes de la licence puis cliquer sur Finish. Il faut ensuite patienter un peu. Si un message vous demande (en anglais !) si vous voulez continuer l'installation, cliquez sur OK.
Une fois l'installation terminée, une fenêtre vous demandera de redémarrer Eclipse, cliquez sur Restart Now.
Vous devriez maintenant avoir des icônes supplémentaires dans Eclipse

Pour créer un nouveau projet, cliquez sur l'icône ronde bleue avec un g à l'intérieur. Vous devriez alors obtenir cette fenêtre

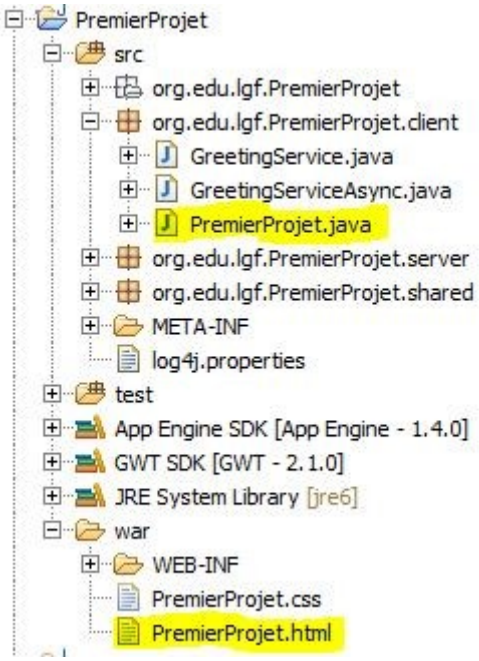


"Project name" : vous devez tout simplement entrer ici le nom de votre projet !

"Package" : ici, c'est un peu plus compliqué. Imaginez que vous décidiez d'écrire une application qui porte le nom de "Toto", tellement fier de votre oeuvre, vous publiez votre application. Mais à l'autre bout du monde (en Australie par exemple, mais bon, cela n'a aucune importance !) Mr Smith,

décide lui aussi de publier une application "Toto" (mais qui, à part le nom, n'a aucun rapport avec la vôtre) : Problème ?
Pas vraiment, car, en théorie, vos 2 applications n'auront pas le même "Package name". Un package name a, en général, la forme suivante :
org.education.lgf.atelier
Oui, votre première impression est la bonne, cela ressemble beaucoup a une adresse internet à l'envers. A la place de "org", vous auriez pu mettre "fr" ou "com". Notre package name indique que notre application a un rapport avec l'éducation, avec le lycée G Fichet (lgf) et avec l'atelier scientifique.
Il y a très peu de chances pour que notre ami australien, Mr Smith, ait le même package name. Nos applications pourront donc être distinguées.
Laisser les autres paramètres tels quels (pour l'instant)

Voilà, votre premier projet est créé.
Par défaut, le plugin met en place une arborescence de fichiers et de dossiers assez complexes.
Voici l'arborescence d'un projet "PremierProjet" :



Nous allons uniquement nous intéresser aux fichiers PremierProjet.java et PremierProjet.html

Eclipse vous propose un projet "exemple" d'où les nombreuses lignes déjà présentes dans les 2 fichiers. Vous, vous devez partir d'un projet "vierge".

Supprimer la plupart des lignes à l'exception de quelques-unes, le but étant de vous retrouver avec ceci :
fichier java client

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;

public class Original implements EntryPoint {
      public void onModuleLoad() {
      }
}
```

NB : le nom du package et le nom de la classe peuvent-être différents, selon vos choix. La méthode

"onModuleLoad" est l'équivalent de la méthode "main" en java.

fichier html dossier "war" (dans notre exemple : PremierProjet.html)

```
<html>
  <head>
    <link type="text/css" rel="stylesheet" href="Original.css">
    <title>Projet GWT LGF</title>
    <script type="text/JavaScript" language="JavaScript"
src="original/original.nocache.js"></script>
  </head>
  <body>
    <noscript>
      <div style="width: 22em; position: absolute; left: 50%; margin-left:
-11em; color: red; background-color: white; border: 1px solid red; padding: 4px;
font-family: sans-serif">
        Your web browser must have JavaScript enabled
        in order for this application to display correctly.
      </div>
    </noscript>

    <h1>Projet GWT LGF</h1>
  </body>
</html>
```

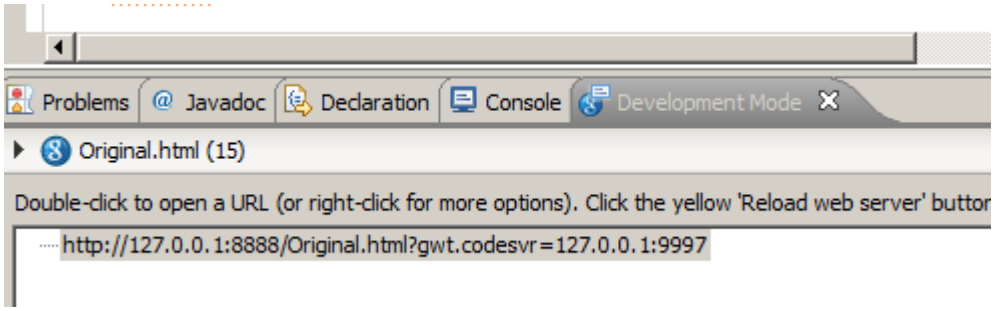
À partir de ce "squelette", nous allons placer notre premier widget : un bouton.

Un widget doit être placé dans un panel (un support), le panel de base (celui qui "couvre" toute votre page HTML) s'appelle `RootLayoutPanel`, il est créé en utilisant la méthode static "get". Nous allons "déposer" un bouton dans le `RootLayoutPanel`. Un bouton est objet au sens POO du terme.

ex 1

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.RootLayoutPanel;

public class Original implements EntryPoint {
      public void onModuleLoad() {
            final Button monBouton = new Button ();
            monBouton.setText("Cliquez Ici");
            RootLayoutPanel.get().add(monBouton);
      }
}
```

La méthode setText appliquée à l'instance monBouton que nous venons de créer, permet d'écrire un texte dans le bouton.
Notez aussi le **`final`** devant **`Button.`** Notre bouton (monBouton) ne sera pas modifié en cours d'exécution, le final permet d'optimiser le code (prends moins de place en mémoire).
Pour tester votre programme, cliquer sur la même touche qu'en Java (Run As....), attendre quelques secondes (le cas échéant, choisir "Web Application"). Après une attente plus ou moins longue, une adresse devrait apparaître dans l'onglet "Development Mode".



Faites un clic droit sur cette adresse et choisir "Open". Vous pouvez aussi copier-coller cette adresse dans votre navigateur préféré. Sans trop entrer dans les détails, un serveur virtuel est créé, cette adresse est donc l'adresse de ce serveur virtuel. Votre navigateur devrait vous demander d'installer un plugin supplémentaire: acceptez.

En testant votre programme , vous pouvez vous rendre compte que le bouton "prend" toute la place disponible, c'est-à-dire toute la page.

Pour avoir un "beau" bouton, vous devez créer un autre panel qui contiendra votre bouton. Ce panel étant lui-même ajouté à notre panel `RootLayoutPanel`.
Nous allons utiliser un "VerticalPanel" (vous comprendrez un peu plus loin le "Vertical"). Nous devons créer une instance de type "VerticalPanel" (monPanel), ajouter "monBouton" à "monPanel" (à l'aide de la méthode "add"), puis enfin ajouter "monPanel" à "RootLayoutPanel".

ex2

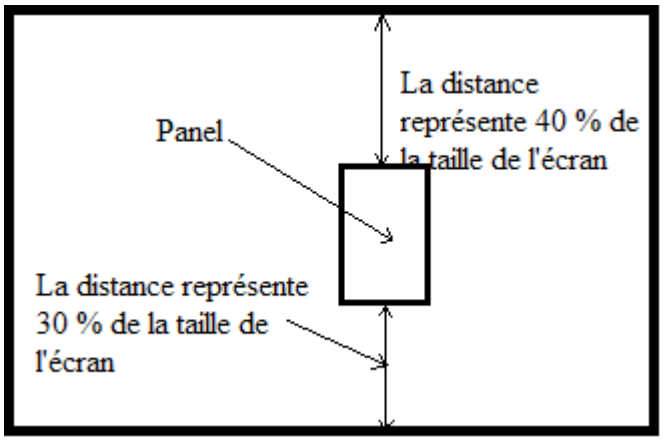
```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.RootLayoutPanel;
import com.google.gwt.user.client.ui.VerticalPanel;

public class Original implements EntryPoint {
	public void onModuleLoad() {
		final Button monBouton = new Button ();
		monBouton.setText("Cliquez Ici");
		final VerticalPanel monPanel = new VerticalPanel ();
		monPanel.add(monBouton);
		RootLayoutPanel.get().add(monPanel);
	}
}
```

Il est bien sûr possible de "déplacer" le VerticalPanel dans la page (pour l'instant il se trouve en haut et à gauche). Nous allons utiliser les méthodes :
`setWidgetLeftRight` et `setWidgetTopBottom`

La méthode `setWidgetLeftRight` vous permet de spécifier la distance entre les bords verticaux de la page et votre widget (pour nous ce widget sera un VerticalPanel).
La méthode `setWidgetTopBottom` vous permet de spécifier la distance entre les bords horizontaux de la page et votre widget.

Si l'on a :

```
setWidgetTopBottom (40, Unit.PCT,30, Unit.PCT)
```



Il est possible de choisir différents types d'unités, ici, nous utilisons "le pourcentage de la taille totale de l'écran" (d'où le Unit.PCT). Il faudra au maximum éviter d'utiliser des tailles en "pixel".

Nous avons le même principe pour la méthode `setWidgetLeftRight` avec la gauche et la droite de l'écran.

ex3

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
i :mport com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.RootLayoutPanel;
import com.google.gwt.user.client.ui.VerticalPanel;

public class Original implements EntryPoint {
	public void onModuleLoad() {
		final Button monBouton = new Button ();
		monBouton.setText("Cliquez Ici");
		final VerticalPanel monPanel = new VerticalPanel ();
		monPanel.add(monBouton);
		RootLayoutPanel.get().add(monPanel);
		RootLayoutPanel.get().setWidgetTopBottom (monPanel,40,Unit.PCT,40,Unit.PCT);
		RootLayoutPanel.get().setWidgetLeftRight (monPanel,10,Unit.PCT,50,Unit.PCT);
	}
}
```

Il existe d'autres méthodes pour placer votre Panel dans le RootLayoutPanel : `setWidgetLeftWidth` et `setWidgetTopHeight`
Leur utilisation est relativement simple.

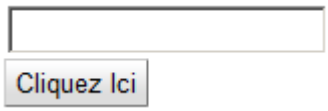
```
setWidgetLeftWidth(monPanel,40,Unit.PCT,10,Unit.PCT)
```

La distance entre le bord gauche de la page et le panel "monPanel" représente 40 % de la largeur de la page. La largeur de "monPanel" représente 10 % de la largeur de la page.
La méthode `setWidgetTopHeight` fonctionne sur le même principe (distance entre le haut et le panel, hauteur du panel).
Personnellement je trouve ces méthodes plus pratiques que les précédentes, j'aurai l'occasion de les utiliser dans les prochains chapitres.
N'hésitez pas à "balader" "monPanel" sur l'écran pour vous familiariser avec tout cela.

Nous allons maintenant utiliser 2 widgets dans un même panel
ex4

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.RootLayoutPanel;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.VerticalPanel;

public class Original implements EntryPoint {
	public void onModuleLoad() {
		final Button monBouton = new Button ();
		final TextBox monTextBox = new TextBox();
		monBouton.setText("Cliquez Ici");
		final VerticalPanel monPanel = new VerticalPanel ();
		monPanel.add (monTextBox);
		monPanel.add(monBouton);
		RootLayoutPanel.get().add(monPanel);
		RootLayoutPanel.get().setWidgetTopBottom (monPanel,40,Unit.PCT,40,Unit.PCT);
		RootLayoutPanel.get().setWidgetLeftRight (monPanel,10,Unit.PCT,50,Unit.PCT);
	}
}
```

résultat :



Le même programme avec `HorizontalPanel` :

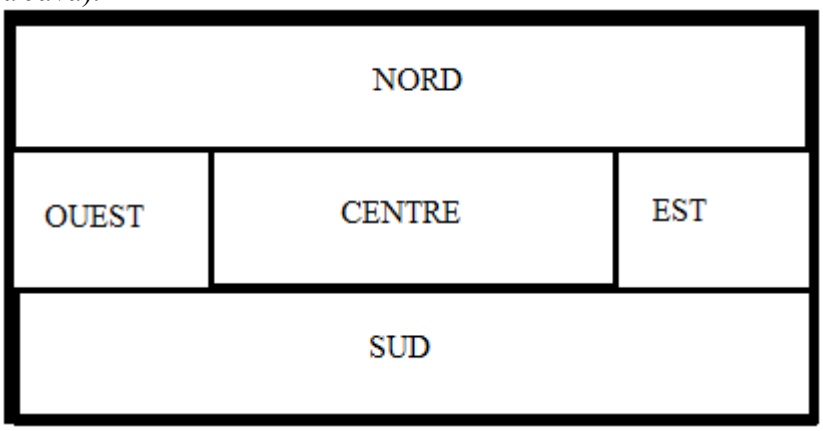
ex5

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.HorizontalPanel;
import com.google.gwt.user.client.ui.RootLayoutPanel;
import com.google.gwt.user.client.ui.TextBox;

public class Original implements EntryPoint {
	public void onModuleLoad() {
		final Button monBouton = new Button ();
		final TextBox monTextBox = new TextBox();
		monBouton.setText("Cliquez Ici");
		final HorizontalPanel monPanel = new HorizontalPanel ();
		monPanel.add (monTextBox);
		monPanel.add(monBouton);
		RootLayoutPanel.get().add(monPanel);
		RootLayoutPanel.get().setWidgetTopBottom (monPanel,40,Unit.PCT,40,Unit.PCT);
		RootLayoutPanel.get().setWidgetLeftRight (monPanel,10,Unit.PCT,10,Unit.PCT);
	}
}
```

résultat :

Je pense qu'un long discours est inutile, vous avez compris la différence entre les 2 types de Panel.

Il existe un grand nombre de Panels dans GWT (en plus des HorizontalPanel et VerticalPanel), nous allons en voir un (pour les autres consulter la doc Google) :
Le Dockpanel :
Dans le cours sur Java, nous avons vu l'équivalent swing du Dockpanel, le BorderLayout (voir le document d'Alice à Java).



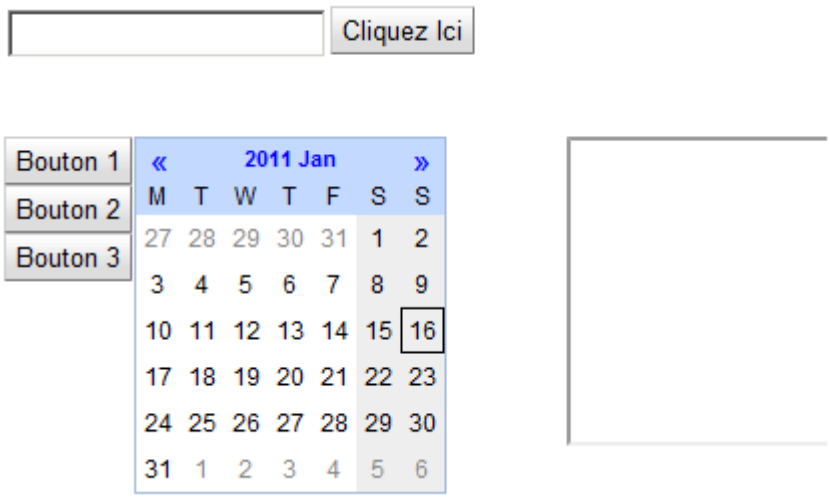
Il est possible de disposer des panels dans des panels :

ex 6

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.DockLayoutPanel;
import com.google.gwt.user.client.ui.HorizontalPanel;
import com.google.gwt.user.client.ui.RichTextArea;
import com.google.gwt.user.client.ui.RootLayoutPanel;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.VerticalPanel;
import com.google.gwt.user.datepicker.client.DatePicker;

public class Original implements EntryPoint {
	public void onModuleLoad() {
		final TextBox monTextBox = new TextBox();
		final DatePicker calendrier= new DatePicker ();
		final RichTextArea zoneTexte = new RichTextArea ();
		final Button monBouton_1 = new Button ();
		final Button monBouton_2=new Button ();
		final Button monBouton_3=new Button ();
		final Button monBouton_4=new Button ();
		monBouton_1.setText("Cliquez Ici");
		monBouton_2.setText("Bouton 1");
		monBouton_3.setText("Bouton 2");
		monBouton_4.setText("Bouton 3");
		final HorizontalPanel monPanel_1 = new HorizontalPanel ();
		final VerticalPanel monPanel_2 = new VerticalPanel ();
		final DockLayoutPanel monPanel_3 = new DockLayoutPanel (Unit.PCT);
		monPanel_1.add (monTextBox);
		monPanel_1.add(monBouton_1);
		monPanel_2.add (monBouton_2);
		monPanel_2.add(monBouton_3);
		monPanel_2.add (monBouton_4);
		monPanel_3.addNorth(monPanel_1,50);
```

```
                monPanel_3.addEast(monPanel_2, 30);
                monPanel_3.addWest(calendrier, 25);
                monPanel_3.add(zoneTexte);
                RootLayoutPanel.get().add(monPanel_3);
                RootLayoutPanel.get().setWidgetTopBottom (monPanel_3,30,Unit.PCT,5,Unit.PCT);
                RootLayoutPanel.get().setWidgetLeftRight (monPanel_3,5,Unit.PCT,5,Unit.PCT);
        }
}
```

Résultat :



Analysez bien cet exemple et modifiez les différents paramètres.

Chapitre III
Les Handlers

Les handlers sont un peu l'équivalent des listeners en Java Swing (voir "D'Alice à Java").

Dans un premier temps, nous allons apprendre à gérer les clics de souris sur les boutons grâce à la méthode onClick et à la classe anonyme ClickHandler (n'hésitez pas à revoir le cours sur Java).

ex 1

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.RootLayoutPanel;

public class Original implements EntryPoint {
	public void onModuleLoad() {
		final Button bouton_1=new Button();
		bouton_1.setText("Bouton");
		bouton_1.addClickHandler(new ClickHandler (){
			public void onClick (ClickEvent event) {
				Window.alert("Bonjour bouton 1");
			}
		});
		RootLayoutPanel.get().add(bouton_1);
		RootLayoutPanel.get().setWidgetLeftWidth (bouton_1,40,Unit.PCT,15,Unit.PCT);
		RootLayoutPanel.get().setWidgetTopHeight (bouton_1,45,Unit.PCT,10,Unit.PCT);

	}
}
```

Quelques commentaires sur l'interface graphique :

Notre bouton (bouton_1) est directement placé dans un RootLayoutPanel. Nous utilisons les 2 méthodes `setWidgetLeftWidth` et `setWidgetTopHeight` pour placer notre bouton sur l'écran. Nous travaillons en "pourcentage" de la taille totale de l'écran (Unit.PCT) pour les 2 méthodes.

Passons maintenant à notre classe anonyme ClickHandler.
Notre classe possède une seule méthode : onClick. En cas de clic sur "bouton_1", une fenêtre d'alerte s'affiche (`Window.alert("Bonjour bouton 1");`), simple non ?

Comme pour Java Swing, l'utilisation de la classe anonyme n'est pas la seule possibilité (parfois, il vaut mieux l'éviter).
Nous pouvons aussi utiliser une instance de notre classe "Original" pour surveiller notre bouton (toujours comme avec Java Swing)

ex 2

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.RootLayoutPanel;

public class Original implements EntryPoint, ClickHandler {
	public void onModuleLoad() {
		final Button bouton_1=new Button();
		bouton_1.setText("Bouton");
		bouton_1.addClickHandler(this);
		RootLayoutPanel.get().add(bouton_1);
		RootLayoutPanel.get().setWidgetLeftWidth (bouton_1,40,Unit.PCT,15,Unit.PCT);
		RootLayoutPanel.get().setWidgetTopHeight (bouton_1,45,Unit.PCT,10,Unit.PCT);
	}
	public void onClick (ClickEvent event){
		Window.alert("Bonjour bouton 1");
	}
}
```

Pas grand chose à dire, il ne faut pas oublier d'implémenter le ClickHandler et d'ajouter la méthode onClick.

3e méthode, l'écriture d'une classe "chargée" de gérer les handlers
ex3
classe MonHandler

```
package org.educ.lgf.atelier.client;

import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.Window;

public class MonHandler implements ClickHandler {
	public void onClick (ClickEvent event){
		Window.alert("Bonjour bouton 1");
	}

}
```

classe Original

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.RootLayoutPanel;

public class Original implements EntryPoint {
	public void onModuleLoad() {
		final Button bouton_1=new Button();
		MonHandler myHandler = new MonHandler ();
		bouton_1.setText("Bouton");
		bouton_1.addClickHandler(myHandler);
		RootLayoutPanel.get().add(bouton_1);
		RootLayoutPanel.get().setWidgetLeftWidth (bouton_1,40,Unit.PCT,15,Unit.PCT);
		RootLayoutPanel.get().setWidgetTopHeight (bouton_1,45,Unit.PCT,10,Unit.PCT);
	}
}
```

Nous créons une classe MonHandler qui implémente ClickHandler et doit donc contenir la méthode onClick(). Sinon rien de spécial.
En fonction du problème à traiter, il sera plus judicieux d'utiliser une méthode plutôt qu'une autre, cela sera à vous de voir.

Voici un nouvel exemple avec 2 boutons à gérer :

ex4

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.HorizontalPanel;
import com.google.gwt.user.client.ui.RootLayoutPanel;

public class Original implements EntryPoint, ClickHandler {
	final Button bouton_1=new Button();
	final Button bouton_2=new Button();
	public void onModuleLoad() {
		bouton_1.setText("Bouton 1");
		bouton_1.addClickHandler(this);
		bouton_2.setText("Bouton 2");
		bouton_2.addClickHandler(this);
		HorizontalPanel monPanel = new HorizontalPanel();
		monPanel.add(bouton_1);
		monPanel.add(bouton_2);
		RootLayoutPanel.get().add(monPanel);
		RootLayoutPanel.get().setWidgetLeftWidth (monPanel,40,Unit.PCT,15,Unit.PCT);
		RootLayoutPanel.get().setWidgetTopHeight (monPanel,45,Unit.PCT,10,Unit.PCT);
	}
	public void onClick (ClickEvent event){
		if (event.getSource()==bouton_1){
			Window.alert("Bonjour bouton 1");
		}
		else {
			Window.alert("Bonjour bouton 2");
		}

	}
}
```

Je vous laisse analyser cet exemple

Nous allons maintenant utiliser un widget "TextBox" et un widget "Label".
ex 5

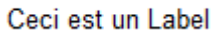
```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.HorizontalPanel;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootLayoutPanel;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.VerticalPanel;

public class Original implements EntryPoint, ClickHandler {
	final Button bouton_1=new Button();
	final TextBox nom = new TextBox();
	final Label labelNom =new Label("Nom");
	public void onModuleLoad() {
		bouton_1.setText("Valider");
		bouton_1.addClickHandler(this);
		VerticalPanel monPanelV = new VerticalPanel();
		HorizontalPanel monPanelH = new HorizontalPanel();
		monPanelH.add(labelNom);
		monPanelH.add(nom);
		monPanelV.add(monPanelH);
		monPanelV.add(bouton_1);
		monPanelV.setSpacing(5);
		monPanelH.setSpacing(5);
		RootLayoutPanel.get().add(monPanelV);
		RootLayoutPanel.get().setWidgetLeftWidth (monPanelV,40,Unit.PCT,15,Unit.PCT);
		RootLayoutPanel.get().setWidgetTopHeight (monPanelV,45,Unit.PCT,15,Unit.PCT);
	}
	public void onClick (ClickEvent event){
		if (nom.getValue()==""){
			Window.alert("Entrer votre nom SVP");
		}
		else {
			Window.alert("Bonjour "+nom.getValue());
		}
	}
}
```

L'organisation des Panels (il faut tout de même noter l'utilisation de la méthode "setSpacing" qui permet d'écarter les panels les uns des autres) ne devrait pas vous poser de problème.
Ce qui va surtout nous intéresser dans cet exemple, c'est la méthode "onClick" qui permet d'afficher le texte contenu dans la TextBox "nom" (la méthode "getValue" retourne la chaîne de caractères qui se trouve dans la TextBox au moment du clic sur le bouton) à l'aide d'une fenêtre. Si la TextBox est vide ("") une fenêtre vous demandant d'entrer votre nom apparaît.

Chapitre IV
Exemples de Widgets et Handlers

Voici une liste non exhaustive des widgets utilisables avec GWT

Label



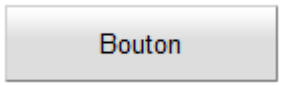
Classe : Label ("votre texte")

Label vous permet d'afficher un texte à l'écran

exemple

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootLayoutPanel;


public class Original implements EntryPoint {
	final Label labelNom =new Label("Ceci est un Label");
	public void onModuleLoad() {
		RootLayoutPanel.get().add(labelNom);
		RootLayoutPanel.get().setWidgetLeftWidth (labelNom,40,Unit.PCT,15,Unit.PCT);
		RootLayoutPanel.get().setWidgetTopHeight (labelNom,45,Unit.PCT,15,Unit.PCT);
	}
}
```

Button

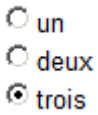


Classe : Button ( )

exemple :

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.RootLayoutPanel;

public class Original implements EntryPoint {
	public void onModuleLoad() {
		final Button bouton_1=new Button();
		bouton_1.setText("Bouton");
		bouton_1.addClickHandler(new ClickHandler (){
			public void onClick (ClickEvent event) {
				Window.alert("Bonjour bouton 1");
			}
		});
		RootLayoutPanel.get().add(bouton_1);
		RootLayoutPanel.get().setWidgetLeftWidth (bouton_1,40,Unit.PCT,10,Unit.PCT);
		RootLayoutPanel.get().setWidgetTopHeight (bouton_1,45,Unit.PCT,6,Unit.PCT);

	}
}
```

RadioButton



Classe : RadioButton ("nom du groupe","valeur")

exemple

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.user.client.ui.RadioButton;
import com.google.gwt.user.client.ui.RootLayoutPanel;
import com.google.gwt.user.client.ui.VerticalPanel;

public class Original implements EntryPoint {
	public void onModuleLoad() {
		final RadioButton r1 = new RadioButton ("rG","un" );
		final RadioButton r2 = new RadioButton ("rG","deux");
		final RadioButton r3 = new RadioButton ("rG","trois");
		final VerticalPanel monPanel = new VerticalPanel ();
		monPanel.add (r1);
		monPanel.add (r2);
		monPanel.add (r3);
		RootLayoutPanel.get().add(monPanel);
		RootLayoutPanel.get().setWidgetTopBottom (monPanel,40,Unit.PCT,40,Unit.PCT);
		RootLayoutPanel.get().setWidgetLeftRight (monPanel,10,Unit.PCT,50,Unit.PCT);
	}
}
```

Un seul bouton par "groupe" peut être coché (ici tous les boutons appartiennent au groupe "rG").

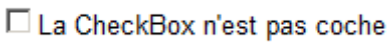
Une interface ("ValueChangeHandler") permet de suivre votre choix.

exemple

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.event.logical.shared.ValueChangeEvent;
import com.google.gwt.event.logical.shared.ValueChangeHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.RadioButton;
import com.google.gwt.user.client.ui.RootLayoutPanel;
import com.google.gwt.user.client.ui.VerticalPanel;

public class Original implements EntryPoint, ValueChangeHandler<Boolean> {
	final RadioButton r1 = new RadioButton ("rG","un" );
	final RadioButton r2 = new RadioButton ("rG","deux");
	final RadioButton r3 = new RadioButton ("rG","trois");
	public void onModuleLoad() {
		final VerticalPanel monPanel = new VerticalPanel ();
		monPanel.add (r1);
		monPanel.add (r2);
		monPanel.add (r3);
		r1.addValueChangeHandler(this);
		r2.addValueChangeHandler(this);
		r3.addValueChangeHandler(this);
		RootLayoutPanel.get().add(monPanel);
		RootLayoutPanel.get().setWidgetTopBottom (monPanel,40,Unit.PCT,40,Unit.PCT);
		RootLayoutPanel.get().setWidgetLeftRight (monPanel,40,Unit.PCT,50,Unit.PCT);
	}
	public void onValueChange (ValueChangeEvent<Boolean> event){
		if (event.getSource()==r1){
		Window.alert("hello Un");
		}
		if (event.getSource()==r2){
			Window.alert("hello Deux");
		}
		if (event.getSource()==r3){
			Window.alert("hello Trois");
		}

	}

}
```

L'interface "ValueChangeHandler" doit être suivie par le type de valeur qui va changer (ici <Boolean>, le radioButton est coché (true) ou n'est pas coché (false)). La méthode getSource( ) appliquée à l'instance "event" (`event.getSource()`) renvoie le nom de l'instance qui vient de changer. Cette interface ValueChangeHandler est souvent utilisée, il faut donc bien maîtriser cet exemple.

CheckBox



CheckBox est le "cousin" de RadioButton : c'est une case à cocher pouvant prendre 2 états (coché ou décoché).

Classe : CheckBox ("Label de la CheckBox")

exemple

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.event.logical.shared.ValueChangeEvent;
import com.google.gwt.event.logical.shared.ValueChangeHandler;
import com.google.gwt.user.client.ui.CheckBox;
import com.google.gwt.user.client.ui.RootLayoutPanel;


public class Original implements EntryPoint, ValueChangeHandler<Boolean> {
	final CheckBox maBox = new CheckBox ("La CheckBox n'est pas coche");
	String etat;
	public void onModuleLoad() {
		etat="La CheckBox n'est pas coche";
		maBox.addValueChangeHandler(this);
		RootLayoutPanel.get().add(maBox);
		RootLayoutPanel.get().setWidgetTopHeight (maBox,40,Unit.PCT,20,Unit.PCT);
		RootLayoutPanel.get().setWidgetLeftWidth (maBox,40,Unit.PCT,15,Unit.PCT);
	}
	public void onValueChange(ValueChangeEvent<Boolean> event){
		if (event.getValue()==true){
			etat="La CheckBox est coche";
		}
		else{
			etat="La CheckBox n'est pas coche";
		}
		maBox.setText(etat);
	}
}
```

Nous utilisons cette fois-ci la méthode getValue qui renvoie true si la case est cochée et false si la case est décochée.

ToggleButton

Cette fois nous avons affaire au "frère" de CheckBox. ToggleButton est un bouton qui possède 2 états : "enfoncé" ou "non enfoncé"
exemple

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.event.logical.shared.ValueChangeEvent;
import com.google.gwt.event.logical.shared.ValueChangeHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.RootLayoutPanel;
import com.google.gwt.user.client.ui.ToggleButton;


public class Original implements EntryPoint, ValueChangeHandler<Boolean> {
	final ToggleButton tBouton= new ToggleButton ("NON","OUI");
	public void onModuleLoad() {
		tBouton.addValueChangeHandler(this);
		RootLayoutPanel.get().add(tBouton);
		RootLayoutPanel.get().setWidgetTopHeight (tBouton,40,Unit.PCT,4,Unit.PCT);
		RootLayoutPanel.get().setWidgetLeftWidth (tBouton,40,Unit.PCT,3,Unit.PCT);
	}
	public void onValueChange(ValueChangeEvent<Boolean> event){
		if (event.getValue()==true){
			Window.alert("Le bouton est enfonce");
		}
		else{
			Window.alert("Le bouton n'est pas enfonce");
		}
	}
}
```

Remarquez le "`new ToggleButton ("NON","OUI")`", il y a donc la possibilité de changer le texte du bouton selon son état (ici "NON" pour non enfoncé et "OUI" pour enfoncé)

TextBox

TextBox est un champ de saisie.
Nous allons traiter un exemple qui va nous permettre d'utiliser un nouvel Handler : KeyUpHandler

exemple :

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.event.dom.client.KeyUpEvent;
import com.google.gwt.event.dom.client.KeyUpHandler;
import com.google.gwt.user.client.ui.HorizontalPanel;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootLayoutPanel;
import com.google.gwt.user.client.ui.TextBox;


public class Original implements EntryPoint, KeyUpHandler {
	final TextBox tBox= new TextBox ();
	final Label nbr=new Label();
	final HorizontalPanel monHPanel=new HorizontalPanel();
	public void onModuleLoad() {
		tBox.addKeyUpHandler(this);
		tBox.setText("Taper votre texte ici");
		monHPanel.add(tBox);
		monHPanel.add(nbr);
		RootLayoutPanel.get().add(monHPanel);
		RootLayoutPanel.get().setWidgetTopHeight (monHPanel,40,Unit.PCT,4,Unit.PCT);
		RootLayoutPanel.get().setWidgetLeftWidth (monHPanel,40,Unit.PCT,30,Unit.PCT);
	}
	public void onKeyUp(KeyUpEvent event){
		String nbrT=tBox.getText();
		nbr.setText(" Vous avez tape "+nbrT.length()+" caractères");

	}
}
```

Nous utilisons ici l'interface KeyUpHandler avec sa méthode associée onKeyUp. Cette méthode est appelée après chaque appui sur une touche du clavier (il existe aussi les interfaces KeyDownHandler (quand une touche est relâchée, la méthode onKeyDown est appelée) et KeyPressHandler (quand une touche est pressée puis relâchée, la méthode onKeyPress est appelée)).
Notez aussi l'utilisation de la méthode "length ( )" qui compte le nombre de caractères dans une chaîne.
Il existe aussi la classe PasswordText qui fonctionne exactement comme TextBox (les caractères sont masqués, c'est la seule différence).

DatePicker

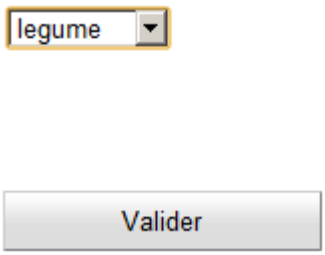
DatePicker vous fournit un calendrier qui va vous permettre de sélectionner une date (il existe même une classe Date)

exemple

```
package org.educ.lgf.atelier.client;

import java.util.Date;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.event.logical.shared.ValueChangeEvent;
import com.google.gwt.event.logical.shared.ValueChangeHandler;
import com.google.gwt.i18n.client.DateTimeFormat;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootLayoutPanel;
import com.google.gwt.user.client.ui.VerticalPanel;
import com.google.gwt.user.datepicker.client.DatePicker;

public class Original implements EntryPoint, ValueChangeHandler<Date> {
	final DatePicker calendrier=new DatePicker();
	final Label dateLabel = new Label();
	final DateTimeFormat formatDate= DateTimeFormat.getFormat("dd/MM/yyyy");
	final VerticalPanel vPanel=new VerticalPanel();

	public void onModuleLoad() {
		calendrier.addValueChangeHandler(this);
		vPanel.add(calendrier);
		vPanel.add(dateLabel);
		RootLayoutPanel.get().add(vPanel);
		RootLayoutPanel.get().setWidgetTopBottom (vPanel,20,Unit.PCT,40,Unit.PCT);
		RootLayoutPanel.get().setWidgetLeftRight (vPanel,20,Unit.PCT,60,Unit.PCT);
	}
	public void onValueChange (ValueChangeEvent<Date> event){
		Date choixDate=event.getValue();
		dateLabel.setText("Vous avez choisi le "+formatDate.format(choixDate));
	}

}
```

Normalement cet exemple ne devrait pas vous poser de problème.

ListBox



La listeBox vous permet de choisir entre plusieurs possibilités.

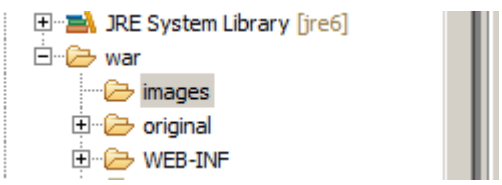
exemple

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.ListBox;
import com.google.gwt.user.client.ui.RootLayoutPanel;

public class Original implements EntryPoint,ClickHandler {
	final Button valider=new Button();
	final ListBox listB=new ListBox();
	public void onModuleLoad() {
		valider.setText("Valider");
		valider.addClickHandler(this);
		listB.addItem("fruit");
		listB.addItem("viande");
		listB.addItem("légume");
		RootLayoutPanel.get().add(valider);
		RootLayoutPanel.get().add(listB);
		RootLayoutPanel.get().setWidgetLeftWidth (valider,10,Unit.PCT,15,Unit.PCT);
		RootLayoutPanel.get().setWidgetTopHeight (valider,25,Unit.PCT,5,Unit.PCT);
		RootLayoutPanel.get().setWidgetLeftWidth (listB,10,Unit.PCT,8,Unit.PCT);
		RootLayoutPanel.get().setWidgetTopHeight (listB,10,Unit.PCT,4,Unit.PCT);
	}

	public void onClick (ClickEvent event) {
		if (listB.isItemSelected(0)){
			Window.alert("fruit");
		}
		if (listB.isItemSelected(1)){
			Window.alert("viande");
		}
		if (listB.isItemSelected(2)){
			Window.alert("légume");
		}
	}
}
```

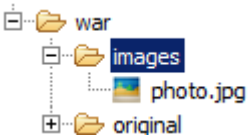
Une instance de la classe ListBox (listB) est créée. Nous utilisons la méthode "addItem" pour "remplir" la listbox avec des chaînes de caractères ("fruit", "viande" et "légume").
Le bouton permet de valider notre choix. La méthode "isItemSelected (0)" renvoie "true" si vous avez sélectionné "fruit" (premier élément de la liste) et "false" dans le cas contraire. Même principe pour "isItemSelected (1)" (qui renvoie "true" si "viande" a été sélectionné) et pour "isItemSelected (2)" (qui renvoie "true" si "légume" a été sélectionné). Prenez garde, le premier élément de la liste correspond toujours au chiffre 0. Le reste ne devrait pas vous poser de problème.

Image

Il est possible d'inclure des images dans une page directement en HTML, mais il est aussi possible (et même conseillé) de créer une instance de la classe Image, cette instance pourra être utilisée comme n'importe quel widget.
Vos images (format jpeg) devront être sauvegardées dans un dossier images qu'il faudra créer dans le dossier war



Pour copier votre photo dans le dossier images, il suffit de "cliquer", "déplacer" et "déposer"



Au moment de la création de votre instance image, il faut indiquer le chemin de votre image (ici, "images/photo.jpg")

exemple :

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootLayoutPanel;
import com.google.gwt.user.client.ui.VerticalPanel;

public class Original implements EntryPoint {
	public void onModuleLoad() {
		final Image monImage = new Image ("images/photo.jpg");
		final Label monTexte = new Label ("Voici une photo");
		final VerticalPanel monPanel = new VerticalPanel ();
		monPanel.add(monTexte);
		monPanel.add(monImage);
		RootLayoutPanel.get().add(monPanel);
		RootLayoutPanel.get().setWidgetTopHeight (monPanel,0,Unit.PCT,100,Unit.PCT);
		RootLayoutPanel.get().setWidgetLeftWidth (monPanel,0O,Unit.PCT,100,Unit.PCT);
	}
}
```

DialogBox

Vous avez toujours rêvé de faire surgir de nulle part une magnifique fenêtre ? La classe DialogBox est faite pour vous !
Votre DialogBox pourra contenir n'importe quel widget, il suffira de créer une instance (DialogBox maBox= new DialogBox()), de lui donner un titre à l'aide de la méthode "setText" et d'ajouter les widgets (ou panels) désirés à l'aide de la méthode "setWidget(votreWidget)". La méthode "show()" permet de faire apparaître votre DialogBox (la méthode "center()" permet aussi de faire apparaître votre fenêtre, mais en la plaçant au centre de votre écran).
Il est aussi possible d'utiliser la méthode "setAnimationEnabled" avec "true" comme paramètre pour que l'apparition et la disparition de la fenêtre se fassent avec une petite animation.
La méthode "setGlassEnabled" (avec le paramètre "true") permet de rendre inactif tout l'écran, à l'exception de la DialogBox.
La méthode "hide" fait disparaître la DialogBox et enfin la méthode "clear" "vide" la DialogBox de tous ses widgets (ou panels).

exemple

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.DialogBox;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootLayoutPanel;
import com.google.gwt.user.client.ui.VerticalPanel;


public class Original implements EntryPoint, ClickHandler {
	final DialogBox maBox =new DialogBox();
	final Button monBout1=new Button("Cliquez ici");
	final Button monBout2=new Button ("OK");
	final Label monLabel = new Label("Voici votre DialogBox");
	final VerticalPanel vPanelBox= new VerticalPanel();

	public void onModuleLoad() {
		RootLayoutPanel.get().add(monBout1);
		RootLayoutPanel.get().setWidgetLeftWidth (monBout1,40,Unit.PCT,15,Unit.PCT);
		RootLayoutPanel.get().setWidgetTopHeight (monBout1,45,Unit.PCT,15,Unit.PCT);
		vPanelBox.add(monLabel);
		vPanelBox.add(monBout2);
		maBox.setText("Ma DialogBox");
		maBox.setAnimationEnabled(true);
		maBox.setGlassEnabled(true);
		maBox.add(vPanelBox);
		monBout1.addClickHandler(this);
	}
	public void onClick (ClickEvent event) {
		maBox.center();
		monBout2.addClickHandler (new ClickHandler (){
			public void onClick (ClickEvent event) {
				maBox.hide();
			}
		});

	}
}}
```

Il existe d'autres widgets, n'hésitez pas à consulter la doc officielle.

Chapitre V
Exemple : site de réservation d'un hôtel

Pour terminer, nous allons créer un site de réservation pour un hôtel fictif. Nous nous occuperons uniquement de la partie "client" (voir "notes de l'auteur"). La création de ce site est un bon moyen pour, non seulement réinvestir toutes les notions vues dans les chapitres précédents, mais aussi avoir un premier exemple de la mise en place d'un projet plus "complexe".
Ce site est visible à l'adresse suivante : http://www.beau-sejour.appspot.com/

Nous allons voir ensemble comment obtenir ce résultat (attention, ce site n'a qu'un intérêt pédagogique, l'aspect esthétique a été totalement négligé, un "vrai" site demanderait un travail beaucoup plus approfondi dans ce domaine !)

Si à votre tour vous désirez créer un site accessible depuis internet, il faudra vous créer un compte Gmail pour pouvoir vous créer un compte Google App Engine sur :
http://code.google.com/intl/fr-FR/appengine/

Il suffit ensuite d'utiliser l'icône  dans Eclipse pour déployer votre site.

Encore un petit avertissement, je ne suis pas développeur professionnel, il doit exister d'autres méthodes de programmation, beaucoup plus judicieuses !

Comme vous pouvez le constater, l'écran est "découpé" en plusieurs morceaux :
le titre et la description :

Hotel Beau Rivage

L'hotel Beau Rivage vous accueille dans un cadre magnifique à 20 m d'une plage de sable fin

les tarifs :

Tarifs au 12/02/11

Chambre double: 90 euros par nuit
Petit déjeuner : 8 euros par personne

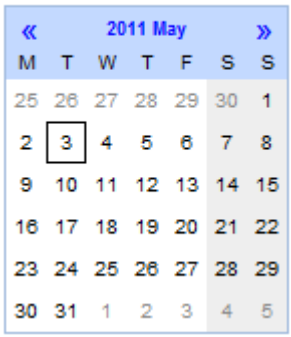
Ces 2 éléments sont écrits dans le fichier HTML du dossier war (voir le tuto sur le HTML)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
  <head>
    <link type="text/css" rel="stylesheet" href="Original.css">
    <title>Projet GWT LGF</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    <script type="text/JavaScript" language="JavaScript"
src="original/original.nocache.js"></script>
  </head>
  <body>
    <noscript>
      <div style="width: 22em; position: absolute; left: 50%; margin-left: -11em; color: red;
background-color: white; border: 1px solid red; padding: 4px; font-family: sans-serif">
        Your web browser must have JavaScript enabled
        in order for this application to display correctly.
      </div>
    </noscript>

    <h1>Hotel Beau Rivage</h1>
    <div id="presentation">
    <p>L'hôtel Beau Rivage vous accueille dans un cadre magnifique à 20 m d'une plage de sable
fin</p>
    </div>
    <div id="tarif">
    <p><h3>Tarifs au 12/02/11</h3><br />Chambre double: 90 euros par nuit<br />
    Petit déjeuner : 8 euros par personne </p>
    </div>
```

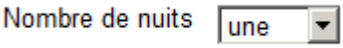
Le reste est uniquement écrit en Java :

La date d'arrivée

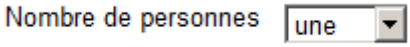
Entrer votre date d'arrivée



Le nombre de nuits

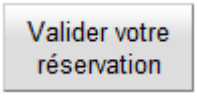


Le nombre de personnes



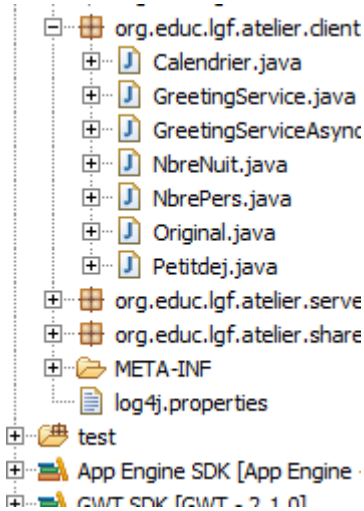
Le choix pour le petit déjeuner

Le bouton de validation



et l'image centrale

L'idée de base est simple, j'ai choisi de créer une nouvelle classe pour chaque élément (sauf pour l'image et le bouton de validation qui sont gérés au niveau de la classe de base qui se nomme ici "Original.java"). Pour créer une nouvelle classe, après avoir sélectionné le package "client" (voir ci-après), choisir File puis New et enfin Class.



Pour illustrer cela, étudions la classe "Calendrier"

Calendrier.java

```
package org.educ.lgf.atelier.client;

import java.util.Date;

import com.google.gwt.event.logical.shared.ValueChangeEvent;
import com.google.gwt.event.logical.shared.ValueChangeHandler;
import com.google.gwt.i18n.client.DateTimeFormat;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.VerticalPanel;
import com.google.gwt.user.datepicker.client.DatePicker;

public class Calendrier implements ValueChangeHandler<Date> {
	DatePicker calen=new DatePicker();
	final Label dateLabel = new Label("Entrer votre date d'arrivée");
	final DateTimeFormat formatDate= DateTimeFormat.getFormat("dd/MM/yyyy");
	String dateArr="";
	public VerticalPanel afficheCalen () {
		calen.addValueChangeHandler(this);
		VerticalPanel vPanelCalen = new VerticalPanel();
		vPanelCalen.add(dateLabel);
		vPanelCalen.add(calen);
		vPanelCalen.setSpacing(10);
		return vPanelCalen;

	}
	public void onValueChange (ValueChangeEvent<Date> event){
		Date choixDate=event.getValue();
		if (choixDate.after(new Date())){
		dateArr="Votre date d'arrivée : "+formatDate.format(choixDate);
		dateLabel.setText(dateArr);
		}
	}
	public String dateGetter (){
		if (dateArr==""){
			Window.alert("Entrer une date SVP");
			return null;
		}
		else {
			return dateArr;
		}

	}

}
```

Je ne vais pas revenir sur l'utilisation du widget DatePicker (voir chapitre précédent). Nous avons 3 méthodes : afficheCalen, onValueChange et dateGetter.

La méthode afficheCalen renvoie un VerticalPanel, ce VerticalPanel est constitué de 2 widgets un DatePicker et un Label (la méthode setSpacing permet de séparer ces 2 widgets).

La méthode onValueChange est appelée lorsque la date sélectionnée sur le DatePicker change. Cette méthode crée une chaîne (dateArr) "Votre date d'arrivée : "+la date sélectionnée sur le DatePicker. Cette chaîne est ensuite utilisée pour "remplir" le Label. Vous noterez aussi la présence de la condition : `if (choixDate.after(new Date())`, cette condition permet d'empêcher le choix d'une date d'arrivée antérieure à la date du jour.

La méthode dateGetter vérifie si une date a bien été sélectionné (dans le cas contraire, une fenêtre "Entrer une date SVP" apparaît), si c'est bien le cas, la méthode renvoie la chaîne dateArr (voir la méthode onValueChange).

Les autres classes sont toutes basées sur le même principe : une méthode qui renvoie un Panel (VerticalPanel ou HorizontalPanel selon les cas) constitué de widgets.

NbreNuit.java

```
package org.educ.lgf.atelier.client;

import com.google.gwt.user.client.ui.HorizontalPanel;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.ListBox;

public class NbreNuit {
	final Label nbreJoLabel=new Label("Nombre de nuits ");
	final ListBox listBoxNbreJo = new ListBox();
	final HorizontalPanel hPanelNJ= new HorizontalPanel ();
	public HorizontalPanel hPanelNbreJ () {
		listBoxNbreJo.addItem("une");
		listBoxNbreJo.addItem("deux");
		listBoxNbreJo.addItem("trois");
		listBoxNbreJo.addItem("quatre");
		listBoxNbreJo.addItem("cinq");
		listBoxNbreJo.addItem("six");
		listBoxNbreJo.addItem("sept");
		listBoxNbreJo.addItem("huit");
		hPanelNJ.add(nbreJoLabel);
		hPanelNJ.add(listBoxNbreJo);
		hPanelNJ.setSpacing(10);
		return hPanelNJ;

	}
	public int nbreJoGetter(){
		int nbreJ=1;
		if (listBoxNbreJo.isItemSelected(0)){
			nbreJ=1;
		}
		if (listBoxNbreJo.isItemSelected(1)){
			nbreJ=2;
		}
		if (listBoxNbreJo.isItemSelected(2)){
			nbreJ=3;
		}
		if (listBoxNbreJo.isItemSelected(3)){
			nbreJ=4;
		}
		if (listBoxNbreJo.isItemSelected(4)){
			nbreJ=5;
		}
		if (listBoxNbreJo.isItemSelected(5)){
			nbreJ=6;
		}
```

```
                if (listBoxNbreJo.isItemSelected(6)){
                        nbreJ=7;
                }
                if (listBoxNbreJo.isItemSelected(7)){
                        nbreJ=8;
                }
                return nbreJ;
        }

}
```

La méthode hPanelNbreJ renvoie un Panel constitué d'une ListBox et d'un Label. La méthode nbreJoGetter renvoie le nombre de jours sélectionnés par l'utilisateur (variable nbreJ).

NbrePers.java

```
package org.educ.lgf.atelier.client;

import com.google.gwt.user.client.ui.HorizontalPanel;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.ListBox;

public class NbrePers {
        final Label nbrePerLabel=new Label("Nombre de personnes ");
        final ListBox listBoxNbrePer = new ListBox();
        final HorizontalPanel hPanelNP= new HorizontalPanel ();
        public HorizontalPanel hPanelNbreP () {
                listBoxNbrePer.addItem("une");
                listBoxNbrePer.addItem("deux");
                listBoxNbrePer.addItem("trois");
                listBoxNbrePer.addItem("quatre");
                listBoxNbrePer.addItem("cinq");
                listBoxNbrePer.addItem("six");
                listBoxNbrePer.addItem("sept");
                listBoxNbrePer.addItem("huit");
                hPanelNP.add(nbrePerLabel);
                hPanelNP.add(listBoxNbrePer);
                hPanelNP.setSpacing(10);
                return hPanelNP;

        }
        public int nbrePersGetter(){
                int nbreP=1;
                if (listBoxNbrePer.isItemSelected(0)){
                        nbreP=1;
                }
                if (listBoxNbrePer.isItemSelected(1)){
                        nbreP=2;
                }
                if (listBoxNbrePer.isItemSelected(2)){
                        nbreP=3;
                }
                if (listBoxNbrePer.isItemSelected(3)){
                        nbreP=4;
                }
                if (listBoxNbrePer.isItemSelected(4)){
                        nbreP=5;
                }
                if (listBoxNbrePer.isItemSelected(5)){
                        nbreP=6;
                }
                if (listBoxNbrePer.isItemSelected(6)){
                        nbreP=7;
                }
                if (listBoxNbrePer.isItemSelected(7)){
                        nbreP=8;
                }
                return nbreP;
        }

}
```

Cette classe ressemble beaucoup à la classe NbreNuit, je ne vais donc pas m'étendre dessus.

Petitdej.java

```
package org.educ.lgf.atelier.client;

import com.google.gwt.event.logical.shared.ValueChangeEvent;
import com.google.gwt.event.logical.shared.ValueChangeHandler;
import com.google.gwt.user.client.ui.HorizontalPanel;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RadioButton;

public class Petitdej implements ValueChangeHandler<Boolean> {
	RadioButton ptiDejRgO = new RadioButton ("rGpetidej","OUI");
	RadioButton ptiDejRgN = new RadioButton ("rGpetidej","NON");
	Label ptiDejLabel = new Label ("Petit déjeuner :");
	Boolean ptiDejO=false;
	public HorizontalPanel affichePtiDej(){
		ptiDejRgO.addValueChangeHandler(this);
		HorizontalPanel hPanelPtiDej=new HorizontalPanel();
		hPanelPtiDej.add(ptiDejLabel);
		hPanelPtiDej.add(ptiDejRgO);
		hPanelPtiDej.add(ptiDejRgN);
		hPanelPtiDej.setSpacing(10);
		return hPanelPtiDej;
	}

	public void onValueChange(ValueChangeEvent<Boolean> event){
		if (event.getValue()==false){
			ptiDejO=false;
		}
		else {
			ptiDejO=true;
		}
	}

	public String ptitDejGetter(){
		String ptitDejS;
		if (ptiDejO==false){
			ptitDejS = "Vous ne prendrez pas le petit déjeuner";
		}
		else {
			ptitDejS = "Vous prendrez le petit déjeuner";
		}
		return ptitDejS;
	}

}
```

La méthode affichePtiDej crée un panel et le retourne. La méthode onValueChange modifie la variable ptiDejO (booléen). Enfin, la méthode ptitDejGetter retourne une chaîne de caractères (ptitDejS).

Pour terminer, la classe Original gère l'affichage des panels, le calcul du nombre de chambres (en fonction du nombre de personnes (2 personnes par chambre)), le prix du séjour, le montant de l'accompte et la création de la fenêtre de résumé.
Je vous laisse étudier le code (en vous aidant des commentaires)

Original.java

```
package org.educ.lgf.atelier.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.dom.client.Style.Unit;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.DialogBox;
import com.google.gwt.user.client.ui.HorizontalPanel;
import com.google.gwt.user.client.ui.Image;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootLayoutPanel;
import com.google.gwt.user.client.ui.VerticalPanel;
```

```
public class Original implements EntryPoint,ClickHandler {
	final Button boutonVal=new Button("Valider votre réservation");
	//définition des différentes instances
	Calendrier calen = new Calendrier();
	Petitdej ptiDej = new Petitdej ();
	NbreNuit nbreN = new NbreNuit ();
	NbrePers nbreP = new NbrePers ();
	//définition des Labels du résumé
	final Label resumCal=new Label ();
	final Label resumPtiDej=new Label ();
	final Label resumNbreJ=new Label ();
	final Label resumNbreP=new Label ();
	final Label resumPrix=new Label ();
	final Label resumAc=new Label ();
	//définition du bouton valider
	final Button resumValid=new Button("Valider");
	//définition de l'image de fond
	final Image photoFond = new Image ("images/image.jpg");
	//création d'un panel à partir de la méthode afficheCalen de la classe Calendrier
	VerticalPanel panelCalen = calen.afficheCalen() ;
	//création d'un panel à partir de la méthode affichePtiDej de la classe Petitdej
	HorizontalPanel panelPtiDej = ptiDej.affichePtiDej();
	//création d'un panel à partir de la méthode hPanelNbreJ de la classe NbreNuit
	HorizontalPanel panelNbreN = nbreN.hPanelNbreJ();
	//création d'un panel à partir de la méthode hPanelNbreP de la classe NbrePers
	HorizontalPanel panelNbreP = nbreP.hPanelNbreP();
	public void onModuleLoad() {
		//mise sous surveillance du bouton valider
		boutonVal.addClickHandler(this);
		//Mise en place des Panels (18 lignes suivantes)
		RootLayoutPanel.get().add(panelCalen);
		RootLayoutPanel.get().setWidgetLeftWidth (panelCalen,3,Unit.PCT,20,Unit.PCT);
		RootLayoutPanel.get().setWidgetTopHeight (panelCalen,10,Unit.PCT,40,Unit.PCT);
		RootLayoutPanel.get().add(panelPtiDej);
		RootLayoutPanel.get().setWidgetLeftWidth (panelPtiDej,3,Unit.PCT,20,Unit.PCT);
		RootLayoutPanel.get().setWidgetTopHeight (panelPtiDej,70,Unit.PCT,10,Unit.PCT);
		RootLayoutPanel.get().add(panelNbreN);
		RootLayoutPanel.get().setWidgetLeftWidth (panelNbreN,3,Unit.PCT,20,Unit.PCT);
		RootLayoutPanel.get().setWidgetTopHeight (panelNbreN,50,Unit.PCT,10,Unit.PCT);
		RootLayoutPanel.get().add(panelNbreP);
		RootLayoutPanel.get().setWidgetLeftWidth (panelNbreP,3,Unit.PCT,20,Unit.PCT);
		RootLayoutPanel.get().setWidgetTopHeight (panelNbreP,60,Unit.PCT,10,Unit.PCT);
		RootLayoutPanel.get().add(boutonVal);
		RootLayoutPanel.get().setWidgetLeftWidth (boutonVal,90,Unit.PCT,7,Unit.PCT);
		RootLayoutPanel.get().setWidgetTopHeight (boutonVal,85,Unit.PCT,7,Unit.PCT);
		RootLayoutPanel.get().add(photoFond);
		RootLayoutPanel.get().setWidgetLeftWidth (photoFond,27,Unit.PCT,50,Unit.PCT);
		RootLayoutPanel.get().setWidgetTopHeight (photoFond,30,Unit.PCT,65,Unit.PCT);
	}
	//La méthode onClick est appelée en cas de validation de la réservation
	public void onClick(ClickEvent event){
		//vérification qu'une date a été choisi
e
		if (calen.dateGetter()!= null ){
			//création des coefs déjeuner,chambre et prix
			int coefPtiDej = 0;
			int coefCham=1;
			int prixSej=0;
			//mise en place des labels du résumé
			resumCal.setText(calen.dateGetter());
			resumPtiDej.setText(ptiDej.ptitDejGetter());
			resumNbreJ.setText("Vous avez réservé "+nbreN.nbreJoGetter()+" nuit(s)");
			resumNbreP.setText("Pour "+nbreP.nbrePersGetter()+" Personne(s)");
			//test si petit déjeuner
			if (ptiDej.ptitDejGetter()=="Vous prendrez le petit déjeuner"){
				coefPtiDej=1;
			}
			//les lignes suivantes permettent de déterminer le nombre de chambres
			if (nbreP.nbrePersGetter()==3 | nbreP.nbrePersGetter()==4 ){
				coefCham=2;
			}
			if (nbreP.nbrePersGetter()==5 | nbreP.nbrePersGetter()==6 ){
				coefCham=3;
			}
			if (nbreP.nbrePersGetter()==7 | nbreP.nbrePersGetter()==8 ){
				coefCham=4;
			}
```

```
                        //Calcul du prix du séjour
                        prixSej=((90*coefCham*nbreN.nbreJoGetter())+
(coefPtiDej*8*nbreN.nbreJoGetter()*nbreP.nbrePersGetter()));
                        //Calcul du montant de l'accompte
                        double prixAcDouble=prixSej*0.2;
                        int prixAc=(int)prixAcDouble;
                        resumPrix.setText("Le prix de votre séjour s'élève à "+prixSej+" euros");
                        resumAc.setText("Accompte à verser : "+prixAc+" euros (20% du montant
total)");
                        //création du Panel "résumé"
                        VerticalPanel vPanelResum=new VerticalPanel();
                        vPanelResum.add(resumCal);
                        vPanelResum.add(resumNbreJ);
                        vPanelResum.add(resumNbreP);
                        vPanelResum.add(resumPtiDej);
                        vPanelResum.add(resumPrix);
                        vPanelResum.add(resumAc);
                        vPanelResum.add(resumValid);
                        vPanelResum.setSpacing(10);
                        //création de la fenêtre "résumé"
                        final DialogBox boxResume=new DialogBox ();
                        boxResume.setGlassEnabled(true);
                        boxResume.setAnimationEnabled(true);
                        boxResume.setText("Résumé de votre réservation");
                    boxResume.setWidget(vPanelResum);
                    boxResume.center();
                    resumValid.addClickHandler(new ClickHandler () {
                              public void onClick (ClickEvent event) {
                                     boxResume.hide();
                                     //appel de la page fin.html
                                     Window.Location.assign("http://www.beau-
sejour.appspot.com/fin.html");
                               }
                        });
                }

        }
}
```

Voilà, cette introduction à GWT côté client est maintenant terminée, pour ceux qui voudraient approfondir la question, je vous propose ci-dessous deux ouvrages à consulter :

GWT Créer des applications web interactives avec Google Web Toolkit (versions 1.7 et 2.0) d'Olivier Gérardin

GWT (Google Web Toolkit) - Développez des Applications Internet Riches (RIA) en Java de Damien Picard