


Département
d'Informatique

Introduction à GWT



Frédéric MOAL

Master 1 MIAGE

Année 2011/2012

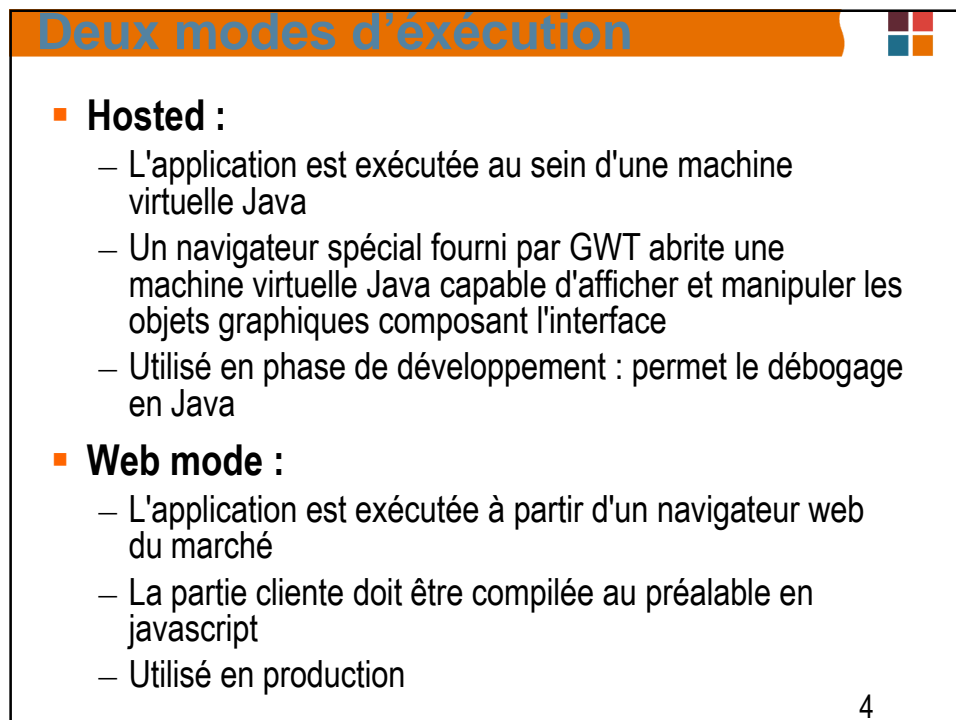
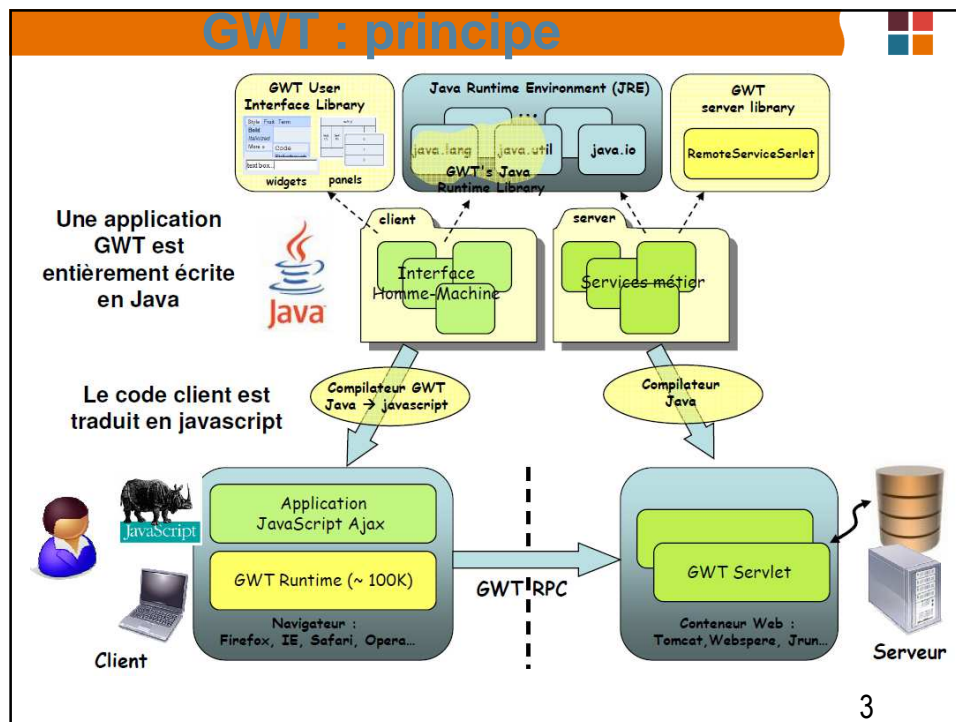
1

GWT : références



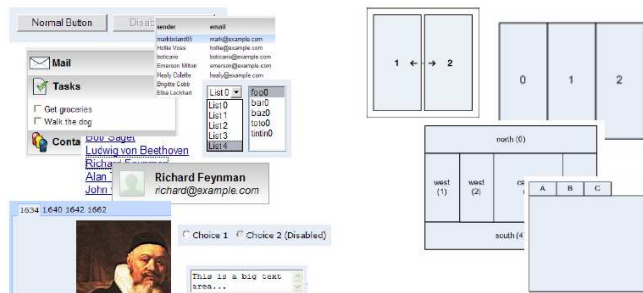
- [Developpez.com](http://developpez.com)
- javapassion.com
- Google : <http://code.google.com/intl/fr/webtoolkit/>
- Tutos :
<http://www.netbeans.org/kb/docs/web/quickstart-webapps-gwt.html>
<http://www.vogella.de/articles/GWT/article.html>
- GWT Plug-in GWT NetBeans plug-in flash demo from java.net
- GWT plugin pour Eclipse
- Plugin gwt-maven

2



GWT : approche

- Retour aux sources de la programmation d'IHM graphiques pour le développement de la partie client.
 - programmation similaire à ce qu'il se fait avec Swing, SWT ou Visual Basic
 - assembler des composants graphiques (widgets)
 - armer des gestionnaires sur les événements reçus par les widgets
- Possibilité de définir de nouveaux widgets ou d'intégrer des frameworks javascript (Dojo, jQuery, Yahoo UI...)



5

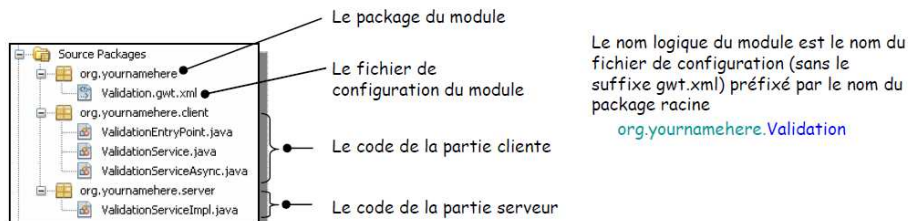
GWT : vocabulaire

- Module : une application GWT
 - Chaque module « monmodule » est décrit par un fichier de config « monmodule.gwt.xml »
- Entry Point : une classe qui définit un « main() »
 - Un ou plusieurs par module
- Page HTML : support pour un module
 - Exécute le code des modules
 - Balises div dans lesquelles GWT met ses widgets

6

Module GWT

- Module : composant d'IHM de haut niveau défini par GWT



Validation.gwt.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<module>
  <inherits name="com.google.gwt.user.User" />
  <entry-point class="org.yournamehere.client.ValidationEntryPoint" />
</module>
```

Annotations:

- `<inherits name="com.google.gwt.user.User" />`: Modules dont hérite ce module
- `<entry-point class="org.yournamehere.client.ValidationEntryPoint" />`: Points d'entrée de ce module : classe Java chargée et exécutée à démarrage du module

7

Création d'un module

- La partie cliente du module implémente l'interface `EntryPoint`

```
public class ValidationEntryPoint implements EntryPoint {

    public ValidationEntryPoint() {
    }

    public void onModuleLoad() {
        final Label userID = new Label("User ID : ");
        final HTML lblServerReply = new HTML();
        final TextBox txtUserInput = new TextBox();
        final Button button = new Button("Create Account");
        button.setEnabled(false);

        HorizontalPanel panel1 = new HorizontalPanel();
        panel1.add(userID);
        panel1.add(txtUserInput);
        panel1.add(lblServerReply);
        VerticalPanel panel2 = new VerticalPanel();
        panel2.add(panel1);
        panel2.add(button);
        RootPanel.get("slot0").add(panel2);
    }
}
```

Annotations:

- `public ValidationEntryPoint() { }`: Constructeur sans paramètres
- `public void onModuleLoad() { }`: Méthode invoquée au chargement du module construit le contenu du module
- `new Label("User ID : ");`, `new HTML();`, `new TextBox();`, `new Button("Create Account");`: Création des composants (widgets GWT)
- `panel1.add(userID);`, `panel1.add(txtUserInput);`, `panel1.add(lblServerReply);`: Assemblage des composants
- `RootPanel.get("slot0").add(panel2);`: id de l'élément de la page HTML où sera "accroché" le `RootPanel` de ce module



8

Intégration dans une page HTML

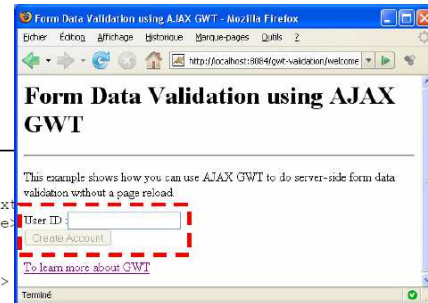
- Host page : page HTML qui contient l'invocation d'un module GWT

Validation.html

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html">
  <title>Form Data Validation using AJAX GWT</title>
</head>
<body>
  <h1>Form Data Validation using AJAX GWT</h1>
  <hr/>
  <p>
    This example shows how you can use AJAX GWT to do server-side
    form data validation without a page reload.
  </p>
  <script language="javascript" src="org.yournamehere.Validation.nocache.js"></script>
  <div id="slot0"></div>
  <p>
    <a href="http://code.google.com/webtoolkit/">To learn more about GWT</a>
  </p>
</body>
</html>
```

Element HTML auquel sera associé le RootPanel du module

Le nom complètement qualifié du module suivi de nocache.js

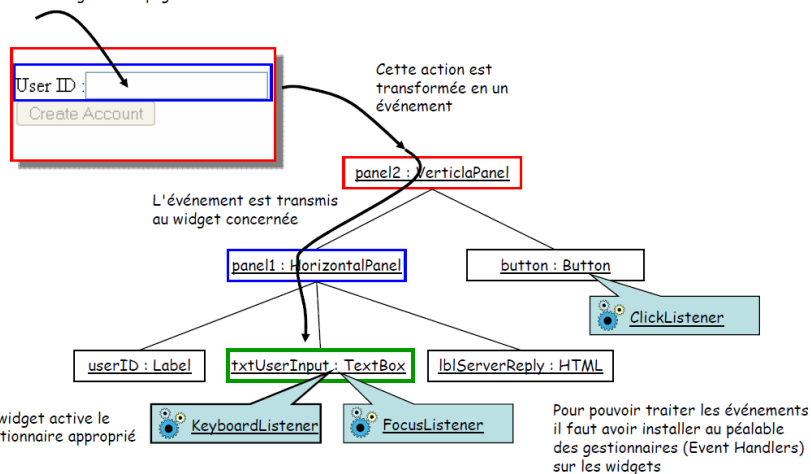


9

Gestion de l'interaction

- Modèle d'événement similaire à celui des frameworks d'IHM classique (Swing, SWT ...)

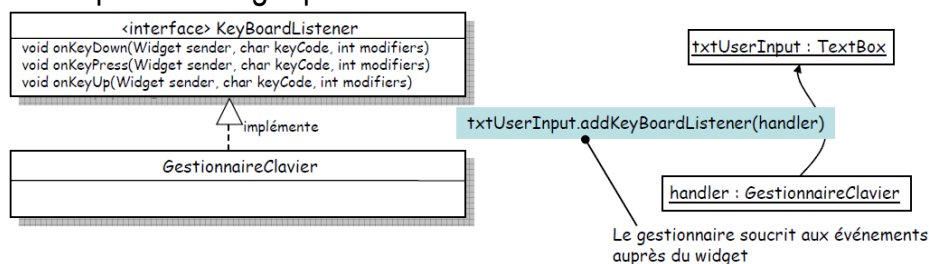
L'utilisateur agit sur la page



10

Gestion de l'interaction

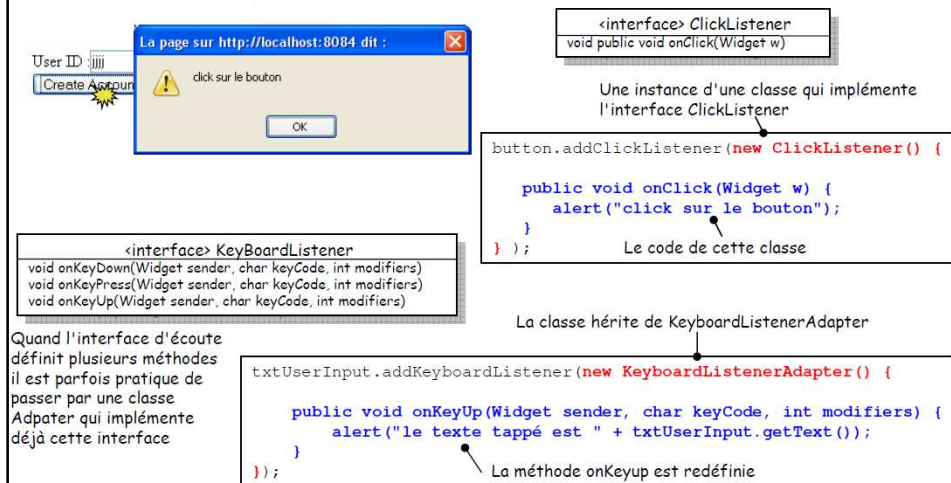
- Une interface d'écoute (listener interface) définit une ou plusieurs méthodes que le widget appelle pour annoncer un événement
- Un gestionnaire destiné à traiter un ou des événements d'un type particulier doit être défini par une classe qui implémente l'interface d'écoute associée.
- Le gestionnaire doit s'enregistrer (en passant sa référence) auprès du widget pour recevoir ces événements.



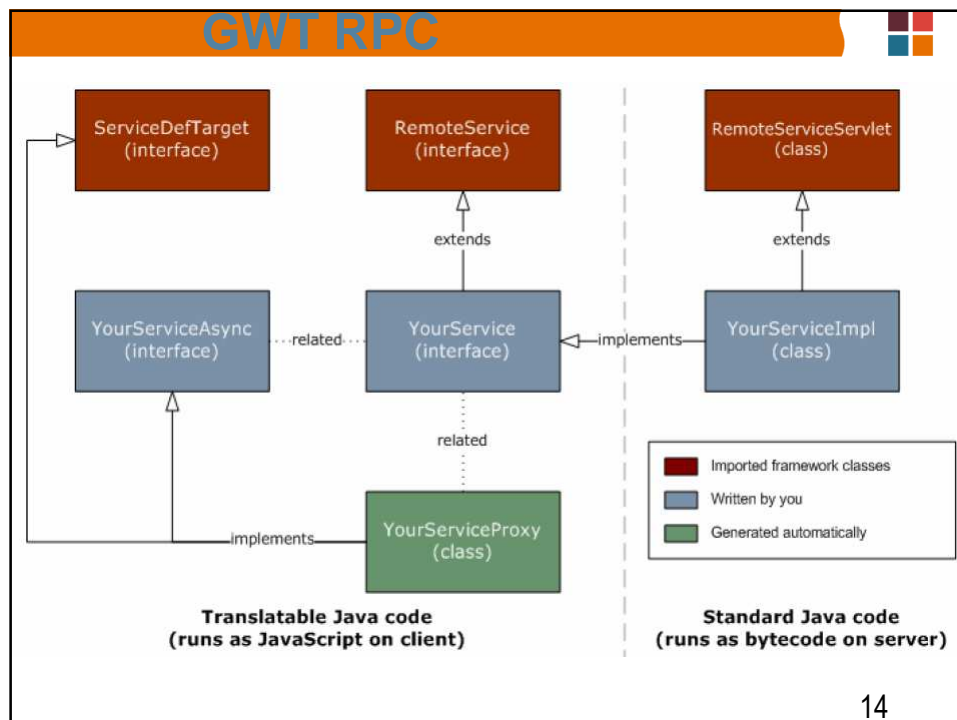
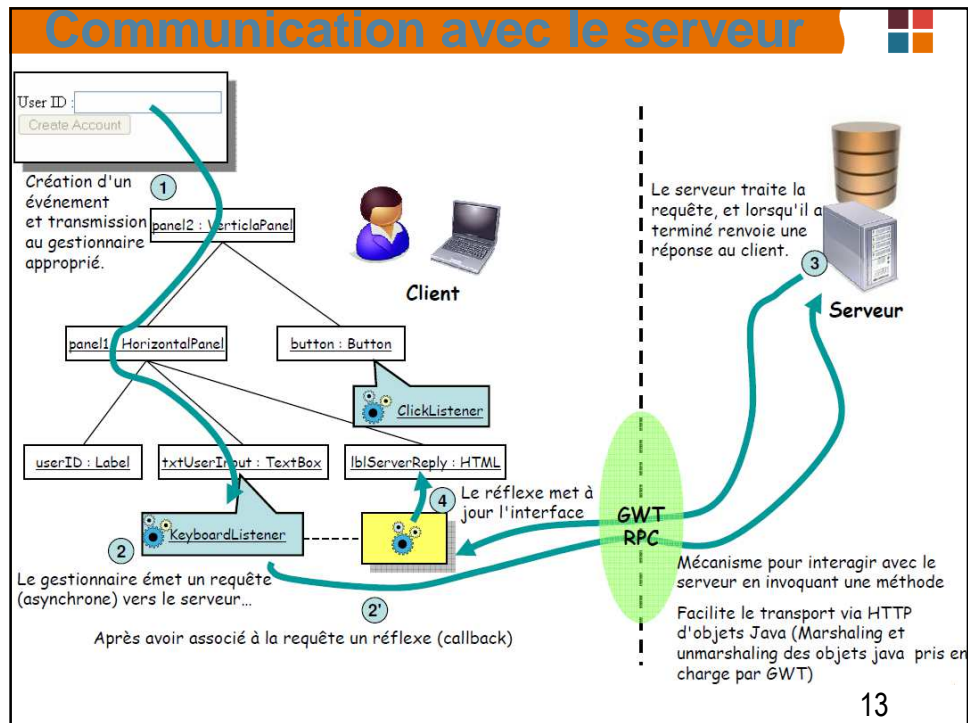
11

Gestion de l'interaction

- Souvent écriture des gestionnaire d'événement à l'aide de classes anonymes internes (inner classes)



12



GWT RPC

1 Ecriture des interfaces de service

Source Packages

- org.yournamehere
 - Validation.gwt.xml
- org.yournamehere.client
 - ValidationEntryPoint.java
 - ValidationService.java
 - ValidationServiceAsync.java
- org.yournamehere.server
 - ValidationServiceImpl.java

ValidationService.java

```
package org.yournamehere.client;

import com.google.gwt.user.client.rpc.RemoteService;

public interface ValidationService extends RemoteService {
    public Boolean validate(String s);
    public Boolean createAccount(String s);
}
```

ValidationServiceAsync.java

```
import com.google.gwt.user.client.rpc.AsyncCallback;

public interface ValidationServiceAsync {
    public void validate(String s, AsyncCallback callback);
    public void createAccount(String s, AsyncCallback callback);
}
```

Même nom que l'interface synchrone mais on ajoute le suffixe Async

Ajout d'un paramètre de type AsyncCallback

```
void onFailure(java.lang.Throwable caught);
void onSuccess(java.lang.Object result);
```

Le type de retour est toujours void

Translatable Java code (runs as JavaScript on client)

Standard Java code (runs as bytecode on server)

15

GWT RPC

2 Implémentation du service

Source Packages

- org.yournamehere
 - Validation.gwt.xml
- org.yournamehere.client
 - ValidationEntryPoint.java
 - ValidationService.java
 - ValidationServiceAsync.java
- org.yournamehere.server
 - ValidationServiceImpl.java

ValidationServiceImpl.java

```
public class ValidationServiceImpl extends RemoteServiceServlet
    implements ValidationService {

    ...

    public Boolean validate(String s) {
        if (!accounts.containsKey(s.trim())) {
            return Boolean.TRUE;
        } else {
            return Boolean.FALSE;
        }
    }

    public Boolean createAccount(String s) {
        if ((s != null) && !accounts.containsKey(s.trim())) {
            accounts.put(s.trim(), "account data");
            return Boolean.TRUE;
        } else {
            return Boolean.FALSE;
        }
    }
}
```

Vérifie qu'il n'existe pas de compte au nom de s sur le serveur

Création du compte sur le serveur renvoie True si la création a réussi False sinon

Translatable Java code (runs as JavaScript on client)

Standard Java code (runs as bytecode on server)

GWT RPC

3 Configuration du service

Configuration Files

- MANIFEST.MF
- context.xml
- gwt.properties
- web.xml

Server Resources

- org.yournamehere
- Validation.gwt.xml
- org.yournamehere.client
- ValidationEntryPoint.java

Validation.gwt.xml

```
<module>
<inherits name="com.google.gwt.user.User"/>
<entry-point class="org.yournamehere.client.ValidationEntryPoint"/>
<servlet path="/validationService" class="org.yournamehere.server.ValidationServiceImpl"/>
</module>
```

Web.xml

```
<servlet>
<servlet-name>ValidationService</servlet-name>
<servlet-class>org.yournamehere.server.ValidationServiceImpl</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>ValidationService</servlet-name>
<url-pattern>/validationService</url-pattern>
</servlet-mapping>
```

Dans le fichier de configuration du module pour le test en phase de développement (hosted mode)

Dans le fichier web.xml pour le déploiement en production (web mode)

17

GWT RPC

4 Invoquer le service depuis le client

Source Packages

- org.yournamehere
- Validation.gwt.xml
- org.yournamehere.client
- ValidationEntryPoint.java
- ValidationService.java
- ValidationServiceAsync.java
- org.yournamehere.server
- ValidationServiceImpl.java

a) Instancier un proxy client (objet de type ValidationServiceAsync) en utilisant GWT.create()

b) Spécifier l'URL point d'entrée pour le proxy en utilisant ServiceDefTarget

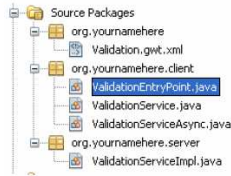
c) Créer un objet callback asynchrone qui sera notifié lorsque le RPC sera terminé

d) Faire l'appel depuis le client

18

GWT RPC

4 Invoquer le service depuis le client



- Instancier un proxy client (objet de type ValidationServiceAsync)
- Spécifier l'URL point d'entrée pour le proxy en utilisant ServiceDefTarget
- Créer un objet callback asynchrone notifié lorsque le RPC sera terminé
- Faire l'appel depuis le client

ValidationEntryPoint.java

```
public class ValidationEntryPoint implements EntryPoint {
    ...
    public static ValidationServiceAsync getService() {
        ...
        (a) ValidationServiceAsync service = (ValidationServiceAsync) GWT.create(ValidationService.class);
        (b) {
            ServiceDefTarget endpoint = (ServiceDefTarget) service;
            String moduleRelativeURL = GWT.getModuleBaseURL() + "validationService";
            endpoint.setServiceEntryPoint(moduleRelativeURL);
            return service;
        }
    }
}
```

GWT génère le code du proxy pour le service.

URL à laquelle l'implémentation du service s'exécute.

19

GWT RPC

4 Invoquer le service depuis le client



- Instancier un proxy client (objet de type ValidationServiceAsync)
- Spécifier l'URL point d'entrée pour le proxy en utilisant ServiceDefTarget
- Créer un objet callback asynchrone notifié lorsque le RPC sera terminé

```
public class ValidationEntryPoint implements EntryPoint {
    ...
    public void onModuleLoad() {
        final Label userID = new Label("User ID : ");
        final HTML lblServerReply = new HTML();
        final TextBox txtUserInput = new TextBox();
        final Button button = new Button("Create Account");

        final AsyncCallback callback = new AsyncCallback() {
            public void onSuccess(Object result) {
                boolean res = ((Boolean) result).booleanValue();
                if (res) {
                    lblServerReply.setHTML("<div style='color:green'>Identité valide ! </div>");
                    button.setEnabled(true);
                } else {
                    lblServerReply.setHTML("<div style='color:red'>Identité invalide ! </div>");
                    button.setEnabled(false);
                }
            }
            public void onFailure(Throwable caught) {
                lblServerReply.setText("Communication failed");
            }
        };
    }
}
```

Classe interne, ce qui lui permet d'accéder aux variables locales de la méthode onModuleLoad pour la mise à jour de l'interface

20

GWT RPC

4 Invoquer le service depuis le client

a) Instancier un proxy client (objet de type ValidationServiceAsync)

b) Spécifier l'URL point d'entrée pour le proxy en utilisant ServiceDefTarget

c) Créer un objet callback asynchrone notifié lorsque le RPC sera terminé

d) Faire l'appel depuis le client

```

public class ValidationEntryPoint implements EntryPoint {
    ...
    public void onModuleLoad() {
        final Label userID = new Label("User ID : ");
        final HTML lblServerReply = new HTML();
        final TextBox txtUserInput = new TextBox();
        final Button button = new Button("Create Account");

        final AsyncCallback callback = new AsyncCallback() {
            public void onSuccess(Object result) { ... }
            public void onFailure(Throwable caught) { ... }
        };

        txtUserInput.addKeyboardListener(new KeyboardListenerAdapter() {
            public void onKeyUp(Widget sender, char keyCode, int modifiers) {
                (d) getService().validate(txtUserInput.getText(), callback);
            }
        });
    }
    ...
  
```

Gestionnaire d'événements clavier associé au champ de saisie de l'identifiant de l'utilisateur

Appel du serveur à distance. Cet appel est asynchrone, le flot de contrôle continuera immédiatement et plus tard le callback sera invoqué quand le service aura été exécuté.

21

GWT et exceptions

- Les appels RPC peuvent provoquer de nombreuses erreurs
 - Problème réseau, panne du serveur, erreur lors d'un traitement d'une requête
- GWT permet de traiter ce type de problèmes à l'aide d'exceptions java.
- Les méthodes d'une interface de service peuvent définir des clauses throws
- Les exceptions ainsi déclarées doivent être traitées dans la méthode onFailure(Throwable) de l'objet callback
- Si un appel de service distant ne peut aboutir (réseau coupé, problème de DNS, arrêt du serveur HTTP) une exception de type InvocationException est passé à la méthode onFailure.

22

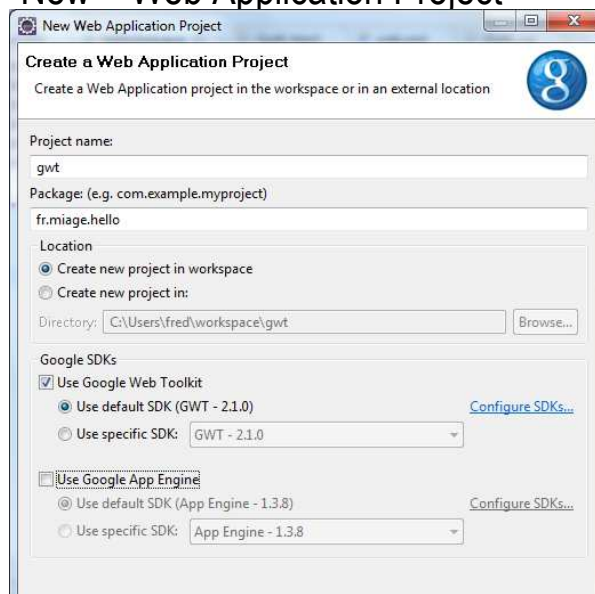
GWT et Eclipse

- Plugins pour Google pour Eclipse :
 - GWT
 - Google App Engine = Cloud
- Help -> Marketplace... récupère plugin Eclipse + les 2 SDK
- Pour créer un projet : File > New > Web Application Project, puis cocher "Use Google Web Toolkit"

23

Exemple mini

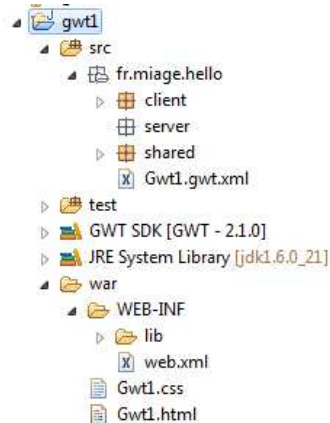
- File > New > Web Application Project



24

Exemple mini

- Génère la structure de base du projet
- Supprimer les classes client, server générées



25

Exemple mini

- Ajouter une classe client, EntryPoint :

```
package fr.miage.hello.client;
public class HelloGwt implements EntryPoint
{
    @Override
    public void onModuleLoad() {
        Label label = new Label("Hello GWT !!!");
        Button button = new Button("Say
something");
        button.addClickHandler(new ClickHandler()
        {
            public void onClick(ClickEvent event) {
                Window.alert("Hello, again");
            }
        });
        RootPanel.get().add(label);
        RootPanel.get().add(button);
    }
}
```

26

Exemple mini

- Définition de l'EntryPoint, dans Gwt1.gwt.xml :

```
<?xml version="1.0" encoding="UTF-8"?>
<module rename-to='gwt1'>
  <inherits name='com.google.gwt.user.User' />
  <inherits name='com.google.gwt.user.theme.standard.Standard' />
  <!-- Specify the app entry point class -->
  <entry-point class='fr.miage.hello.client.HelloGwt' />

  <source path='client' />
  <source path='shared' />
</module>
```

27

Exemple mini

- Création de la page HTML (dans war/): Gwt1.html

```
<!doctype html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=UTF-8">
    <link type="text/css" rel="stylesheet" href="Gwt1.css">
    <title>Premier projet GWT</title>
    <script type="text/javascript" language="javascript"
    src="gwt1/gwt1.nocache.js"></script>
  </head>

  ...
```

28

Exemple mini

- Création de la page HTML (dans war/): Gwt1.html

```
<body>
  <!-- OPTIONAL: include this if you want history support -->
  <iframe src="javascript:''" id="__gwt_historyFrame"
tabIndex='-1'
style="position:absolute;width:0;height:0;border:0"></iframe>
  <!-- RECOMMENDED if your web app will not function without
JavaScript enabled -->
  <noscript>
    <div style="font-family: sans-serif">
      Your web browser must have JavaScript enabled
      in order for this application to display correctly.
    </div>
  </noscript>
  <h1>Hello world !</h1>
</body>
</html>
```

29

Exemple mini

- Modification du css : Gwt1.css

```
h1 {
  font-size: 2em;
  font-weight: bold;
  color: #777777;
  margin: 40px 0px 70px;
  text-align: center;
}
.gwt-Label {
  color: #DF0101;
  font: normal 12px tahoma, arial, helvetica, sans-serif;
  border: 1px solid #99bbe8;
  padding: 14px;
}
.gwt-Button {
  height: 5.7em;
  margin-bottom: 5px;
  padding-bottom: 3px;
  font-size: 12px;
  font-family: arial, sans-serif;
}
```

30

Exemple mini

- Connection dans web.xml comme toujours

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
    2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>

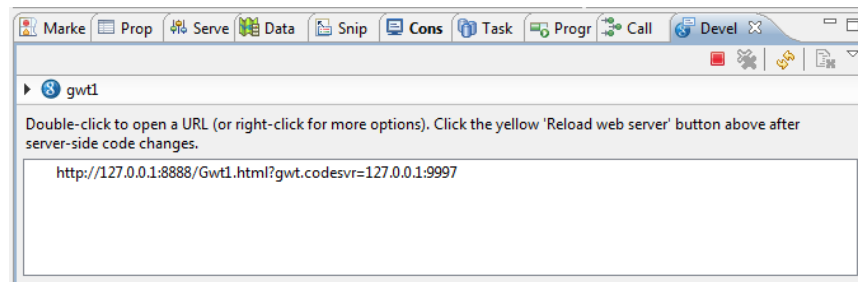
    <!-- Default page to serve -->
    <welcome-file-list>
        <welcome-file>Gwt1.html</welcome-file>
    </welcome-file-list>

</web-app>
```

31

Exemple mini

- Run de l'application : Run As -> (G) Web application
- Ouvre une nouvelle vue :



- Double-clic sur l'URL pour lancer l'application dans le navigateur par défaut

32

Exemple mini

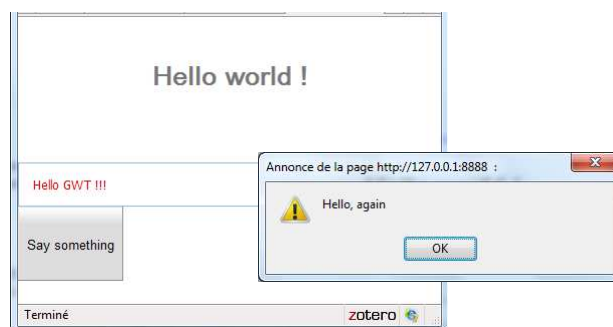
- Premier run : installation du plugin pour le navigateur



33

Exemple mini

- Premier run : installation du plugin pour le navigateur



34

GWT : conclusion

- Les atouts de GWT :
 - Simplicité de mise en œuvre
 - Utilise un paradigme de programmation connu
 - Unifie les technologies nécessaires au développement d'applications Web. Il ne vient pas s'ajouter à la pile des technologies Web/Java (servlets, JSP, JSTL, Struts...) il les remplace.
 - Pas besoin d'apprendre/utiliser javascript, pas besoin de gérer les incompatibilités entre navigateurs (GWT le fait pour vous)
 - Intégration à la plateforme Java
 - Le contrôle statique des types du langage Java réduit les erreurs de programmation et améliore la productivité. Des erreurs JavaScript communes (typos, incompatibilités de types) sont détectées à la compilation
 - Le mode hôte facilite la mise au point
 - Utilisation des environnements de développement Java (Eclipse, Netbeans...) Plugin GWTDesigner sous Eclipse + MAVEN ³⁵

GWT : conclusion

- Les atouts de GWT :
 - Ouverture
 - Facilement extensible
 - De nombreux frameworks opensource complète la palette de composants de base de GWT : GWT ext, MyGWT, GWidget, GWT components
 - Capacité d'intégrer des frameworks javascript externes (JSNI)
 - Ex : projet Tatami ObjetDirect – France Telecom pour encapsulation de DOJO dans des composants GWT
 - Robustesse et stabilité
 - Google prend son temps avant de retirer le Tag beta à une API
 - Pérennité
 - Projet opensource (licence apache 2.0) avec un géant comme principal sponsor
 - Une communauté très active

GWT : conclusion

- Mais encore quelques problèmes :
 - Programmation asynchrone peut être déroutante
 - Tout java n'est pas traductible en javascript
 - Partiellement java.lang et java.util, pas de support pour Java 5(...)
 - Echanges des données entre le client et le serveur limitée
 - Lenteur des build/run

37

GWT : conclusion

- Mais encore quelques problèmes :
 - Manque de maturité des librairies annexes nécessaires, plugins (wrapper JS, maven, Spring security...)
 - Intégration de javascript peut se révéler délicate (wrapper de librairies JS)
 - Intégration avec Spring/Hibernate « un peu » délicate et restrictive (perte mode hosted) mais possible (services du serveur)
 - Nécessité d'une réflexion sur les architectures des applications GWT
 - Problème plus général à Ajax et RIA

38