Sopra group

Google Web Toolkit

Sébastien Chassande-Barrioz

Principe technique d'AJAX / web 2.0

- **Web 1.0 : 1 click =**

  ➔ 1 requête Http

  ➔ rechargement de tout la page

  ➔ Une page web = données + présentation

- **AJAX / web 2.0:**
  - **Utilisation de Javascript**
  - **1 click ➔ des actions**
    - Requêtes Asynchrones
    - Mise à jour partiel de la page (Arbre DOM XML)

  ➔Respect des standards du web (W3C)
  ➔Une page HTML qui s'auto modifie avec du JS
  ➔Ergonomie d'une application lourde dans une application web

Le problème ?

- **Difficulté d'écriture du code JavaScript (Language au typage faible)**
- **Débug compliqué (alert !)**
- **Interprétation différente selon le navigateur**
- **Manque de compétence JavaScript**

Solution: Ne pas écrire de JS à la main !

- **Générer du code JavaScript à partir de code Java**

- **Pourquoi ?**
  - **Compétences nombreuses en développeut Java**
  - **L'environnement de développement Java est très riche (ex: IDE Eclipse, débug pas à pas)**
  - **Un seul code source Java pour tous les navigateurs ➔ génération de JS pour chaque navigateur cible**
  - **Le code Javascript est optimisé : compact, minimal, efficace**

GWT, c'est quoi ?
(Google Web Toolkit)

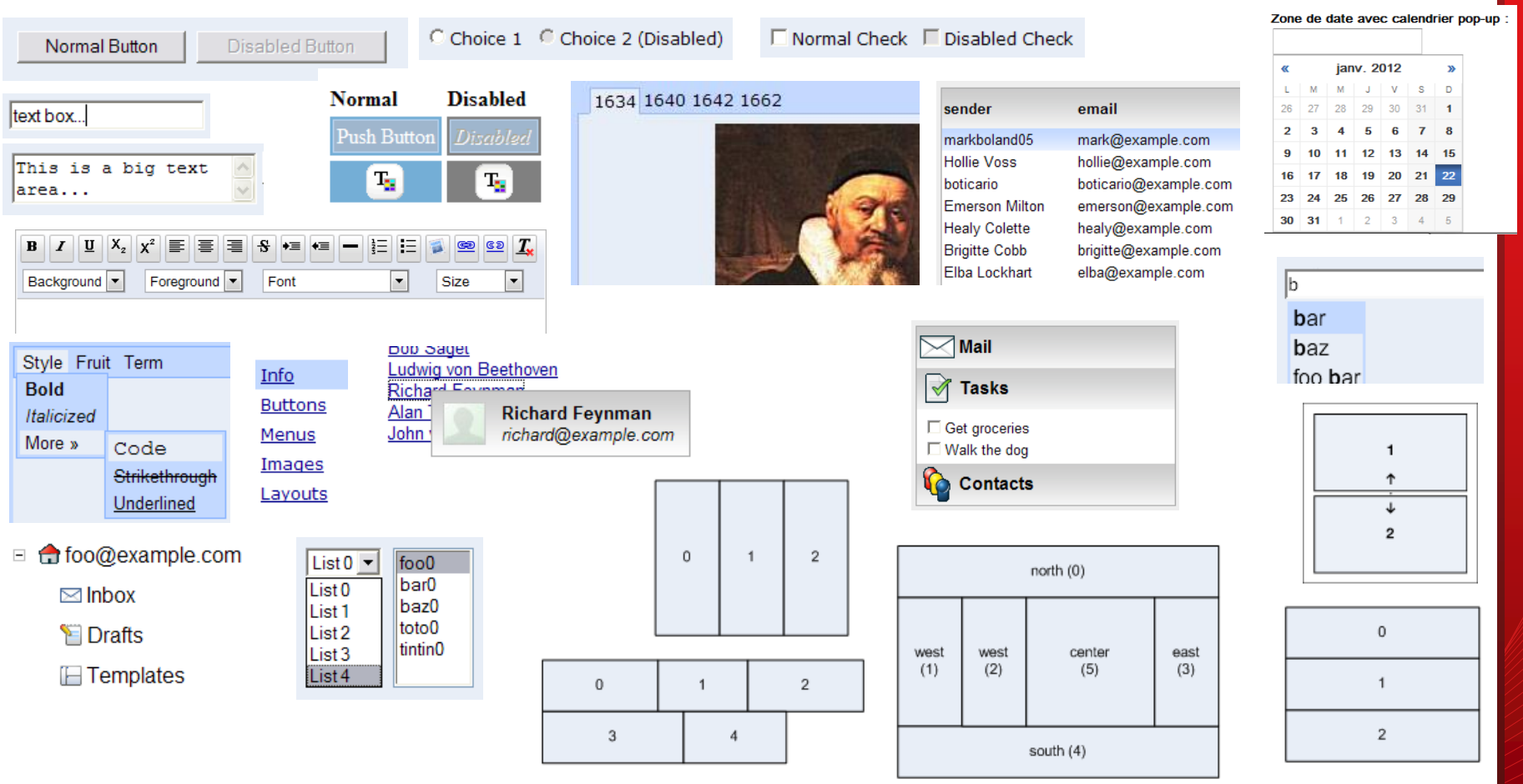
= Canevas logiciel pour la construction d'interface web 2.0

- **Langage Java (java.util.*, java.lang.* …)**
- **GWT fournit une API de type Swing avec des widgets**
  Panel, Button, TextBox …
- **Compilateur du code java en Javascript**
- **Open source, multi navigateur, AJAX**

Qu'est que cela change ?

- **Pages webs = code javascript + html + css + …**
- **Javascript téléchargé et exécuté par le navigateur**
- **Style avec CSS**
- **Interaction serveur pour :**
  1. **Récupérer de données « non présentées »**
  2. **Lancer de calcul compliqué ou métier**
  3. **Transmettre des données au serveur (BDD)**

Widgets graphiques de GWT



Utiliser le showcase pour découvrir tous les widgets :

http://gwt.googleusercontent.com/samples/Showcase/Showcase.html

Les bibliothèque de widget

- **Projet de widget graphique en Java**
  - Ext GWT = GXT (commercial)
  - GwtQuery
  - GWanted (open source)
  - GWT-Widget (open source)

Performance

- **Projet encapsulant des librairies JS**
  - SmartGWT : SmartClient (open source)
  - GWT-EXT : ext ≤ 2.0.3 (open source)
  - Tatami : dojo (open source)

Richesse

Se créer ces propres composants

- **Comment : Par composition de widgets**

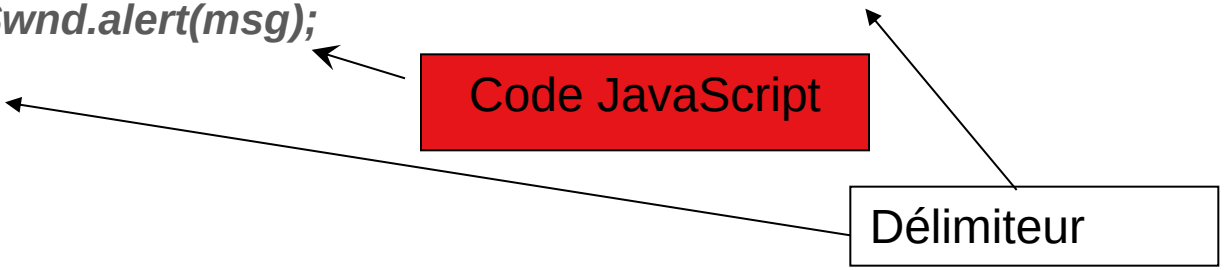
```
public class MyWidget extends Composite {

    HorizontalPanel rootWidget;

    public MyWidget(String label, String value) {
        rootWidget = new HorizontalPanel();
        rootWidget.add(new Label(label));
        rootWidget.add(new TextBox(label));
        initWidget(rootwidget);
    }
}
```

- **Intégration de code natif JavaScript**

```
public class Alert {
    public static native void alert(String msg) /*-{
        $wnd.alert(msg);
    }-*/;
}
```



Et les CSS ?

- **CSS: Feuille de style**
- **Changer le style de la page en changeant la feuille de style**
- **Sur chaque widget on peut définir son style: class ou id**

```
myLabel.setStyleName('toto');
```

```
.toto {
    border: 2px solid #AAAAAA;
    background-color: white;
}
```

- **Dans la classe html qui charge le JS, inclusion de la feuille de style CSS**

- **Widget GWT ne sont pas très beau par défaut**

  ➔ **Nécessaire de faire du CSS pour personnaliser**

Un exemple

```
public class EditContactView extends Composite {

  public EditContactView() {
    VerticalPanel contentDetailsPanel = new VerticalPanel();
    contentDetailsPanel.setWidth("100%");

    initWidget(contentDetailsPanel);

    // Create the contacts list
    detailsTable = new FlexTable();
    detailsTable.setCellSpacing(0);
    detailsTable.setWidth("100%");
    detailsTable.addStyleName("contacts-ListContainer");
    detailsTable.getColumnFormatter().addStyleName(1, "add-contact-input");

    firstName = new TextBox();
    lastName = new TextBox();
    emailAddress = new TextBox();
    detailsTable.setWidget(0, 0, new Label("Firstname"));
    detailsTable.setWidget(0, 1, firstName);
    detailsTable.setWidget(1, 0, new Label("Lastname"));
    detailsTable.setWidget(1, 1, lastName);
    detailsTable.setWidget(2, 0, new Label("Email Address"));
    detailsTable.setWidget(2, 1, emailAddress);
    firstName.setFocus(true);
    contentDetailsPanel.add(detailsTable);

    HorizontalPanel menuPanel = new HorizontalPanel();
    saveButton = new Button("Save");
    cancelButton = new Button("Cancel");
    menuPanel.add(saveButton);
    menuPanel.add(cancelButton);
    contentDetailsPanel.add(menuPanel);
  }
```

Champs de la classe

```
private final TextBox firstName;
private final TextBox lastName;
private final TextBox emailAddress;
private final FlexTable detailsTable;
private final Button saveButton;
private final Button cancelButton;
```

Création d'une grille

Remplissage avec un formulaire

Boutons du formulaire

Mais ou est mon html ?

- **Un seul fichier HTML simple**
  - *Référence le fichier JS initial*
  - *Définit le cadre d'action du code GWT*

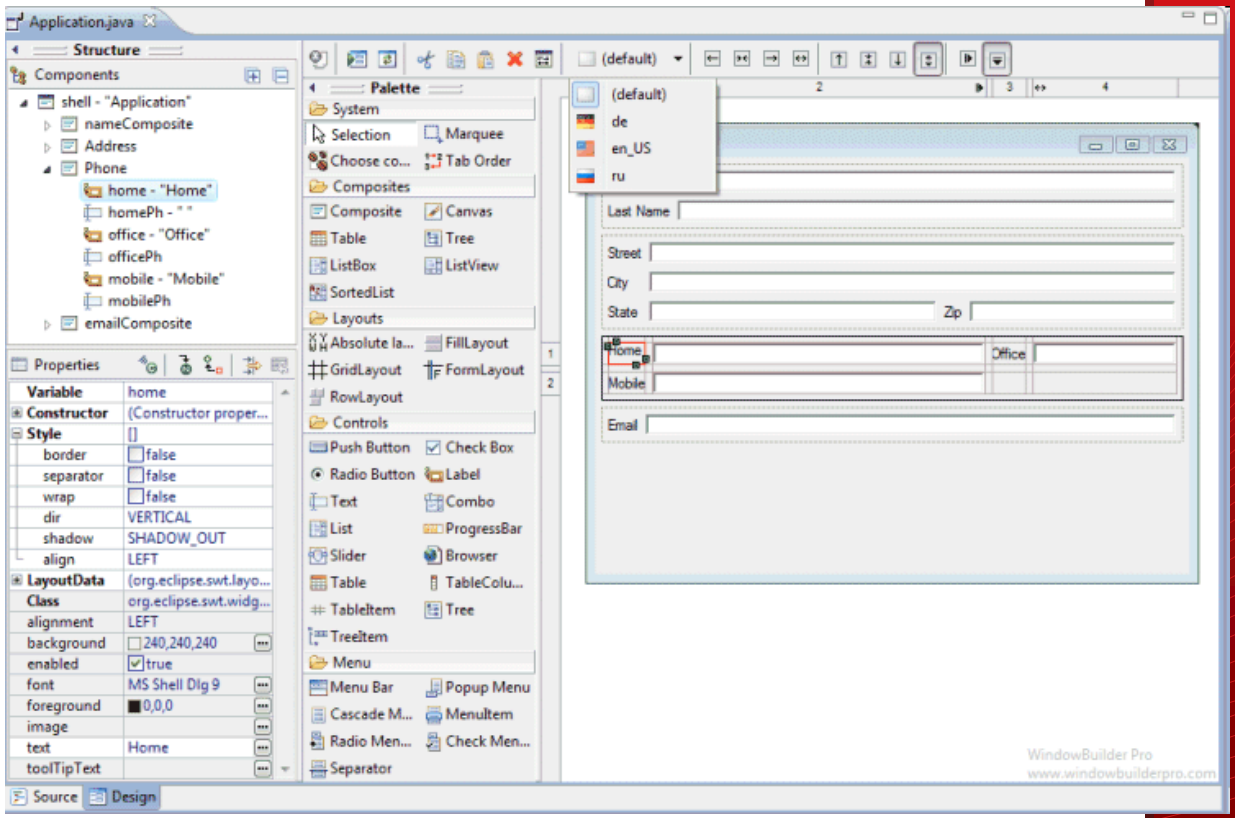
index.html

```
<!doctype html><html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <link type="text/css" rel="stylesheet" href="ImageViewer.css">
    <title>Wrapper HTML for ImageViewer</title>
    <script language="javascript" src="com.mycompany.project.ImageViewer/
                                       com.mycompany.project.ImageViewer.nocache.js">
    </script>
</head>
<body>
    <iframe id="__gwt_historyFrame" style="width:0;height:0;border:0"></iframe>
</body>
</html>
```

GWT designer

- Click droit sur le fichier « sample_gwt.java »
  - Ouvrir avec ➔ « GWT Designer »

- Editeur WYSIWYG
- Bidirectionnel avec le code
- I18N
- Custom Component
- Gestion des événements
- Editeur de menu



Composer du HTML et du GWT

- **Exemple précédent**
  - **Page vierge**
  - **Construction de toute la page par du GWT**
- **Autre voie possible : Injecter des portions de GWT dans des boites html**
  - **Dans le html: identification de la zone par l'id html**

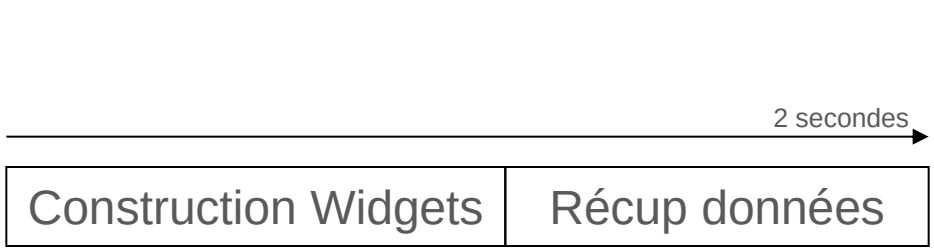
```
<div id="my_panel_1"/>
```

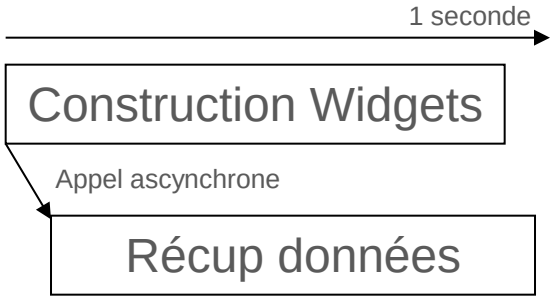
  - **Dans le code Java: Récupération comme un panel via une méthode static**

```
RootPanel rootPanel = RootPanel.get("my_panel_1");
```

Appel serveur ascynchrone

- **Navigateur mono thread :**
  - **Le moteur JavaScript se déconnecterait pendant l'attente d'une réponse synchrone du serveur.**
  - **Le navigateur alerte l'utilisateur quand le JavaScript dure longtemps.**
- **Doit pouvoir supporter une indisponibilité temporaire du serveur sans bloquer l'interface indéfiniment.**
- **Exécution paralléle coté serveur ➔ temps de réponse + court**
- **Appel possible sur plusieurs serveur en parallèle.**





Interaction avec le serveur

- ➔ **Nouveau RPC par GWT**
- ➔ **Ascynchrone !**
- **Définition de services coté serveur**
  - **Interface Java de définition du service (méthodes)**
  - **Implémentation du service par une Servlet**
- **Appel des services par le code client**
  - **RPC Asynchrone sur http**
  - **Interface Java de réception du retour d'appel (CallBack)**

RPC GWT : Exemple

- **Définition de l'interface du service**

```
public interface MyService extends RemoteService {

        Info getInfo(String s);
}

public class Info implements java.io.Serializable {

        public String field1;
        public String field2;
        public Info(String f1, String f2) { field1 = f1; field2 = f2; }
}
```

- **Implémentation de l'interface coté Serveur**

```
public class MyServiceImpl extends RemoteServiceServlet implements MyService {

        public Info getInfo(String s) {

                return new Info(s, s + "." + s);
        }
}
```

RPC GWT : Exemple (suite)

- **Définition de l'interface de callback**

```
public interface MyServiceAsync {

        void getInfo(String s, AsyncCallback<Info> a);
}
```

- **Appel du service depuis du code coté navigateur**

```
MyServiceAsync myService = (MyServiceAsync) GWT.create(MyService.class);

((ServiceDefTarget) myService).setServiceEntryPoint(GWT.getModuleBaseURL() + "myservice");

AsyncCallback<String> callback = new AsyncCallback<String>() {
        public void onSuccess(Info result) {
            /*do some UI stuff to show success*/
        }
        public void onFailure(Throwable caught) {
            /*do some UI stuff to show failure*/
        }
};

myService.getInfo("toto", callback);
```

GWT & Internationalisation

- **Objectifs :**
  - **Supporter plusieurs langues (fr, en, …)**
  - **Permettre le choix des terminologies par un non développeur (ex: Marketing)**
  - **Séparer le code du texte**
  - **Support de messages construits (avec paramètres)**

- **Solution = Internationalisation GWT**
  - **Externaliser les textes dans un fichier .properties**
  - **Utilisation simple via une interface Java + Annotations**

GWT & Internationalisation : Exemple

TotoMessages.properties

```
my.prop.name=bla bli bla blo
another.prop.name=foo bar
```

TotoMessages_fr.properties

…

TotoMessages_en.properties

TotoMessages.java

```
import com.google.gwt.i18n.client.Constant;
public interface TotoMessages extends Constant {
    @Key("my.prop.name");
    String getMyPropName();

    @Key("another.prop.name ");
    String getAnotherPropName();
}
```

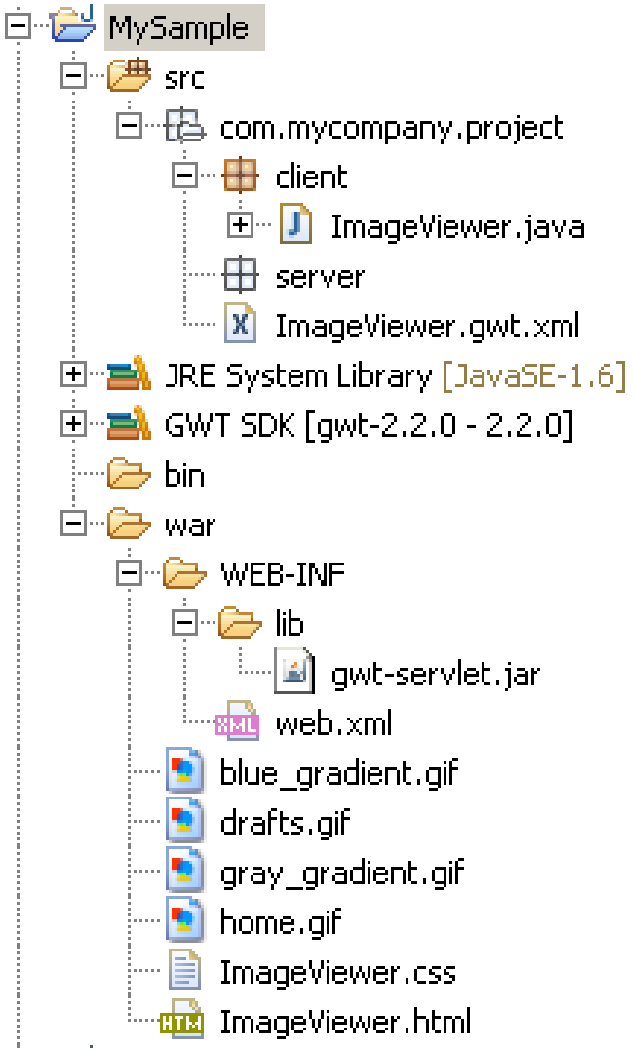
```
TotoMessages msg = (TotoMessages) GWT.create(TotoMessages.class);
Button b = new Button();
b.setText( msg.getMyPropName() );
```

A mettre dans le fichier du module .gwt.xml :

<inherits name="com.google.gwt.i18n.I18N"/>

Organisation d'un projet GWT

- **Séparation Java/html**
  - **Répertoire « src » contient le code java**
  - **Répertoire « war » contient**
    - les ressources web (html, css, jpg…)
    - Les librairies serveurs dans WEB-INF/lib
- **Séparation client/server**
  - **Package « client » contient le code java transformé en JS**
  - **Package « server » contient le code Java qui s'exécutera sur le server**
- **Fichier xml descripteur**



Le concept de Module

- **Définition: Unité ré-utilisable entre les projets**
- **Est utilisé pour la compilation vers le Javascript**
- **Est décrit par un fichier xml**
  - **Ou sont les sources (client, server) ?**
  - **Les dépendances (Module, CSS, JS Script …)**
  - **Les services distants (RPC)**
  - **Le point d'entrée**
  - **Paramètres de compilation (Langage, navigateurs)**
  - **Règles de remplacement**

```
<module>
        <inherits name="com.google.gwt.user.User"/>

        <inherits name="com.google.gwt.user.theme.standard.Standard"/>

        <entry-point class="com.mycompany.project.client.ImageViewer"/>
</module>
```

Le concept de point d'entrée

- **Classe implémentant l'interface EntryPoint**

  ➔ **Implémente la méthode** *void onModuleLoad()*

- **Définit un point de démarrage d'une l'application**

```
public class ImageViewer implements EntryPoint {
   private Button clickMeButton;

   public void onModuleLoad() {
      RootPanel rootPanel = RootPanel.get();

      clickMeButton = new Button();
      rootPanel.add(clickMeButton);
      clickMeButton.setText("Click me!");
      clickMeButton.addClickHandler(new ClickHandler(){
         public void onClick(ClickEvent event) {
            Window.alert("Hello, GWT World!");
         }
      });
   }
}
```

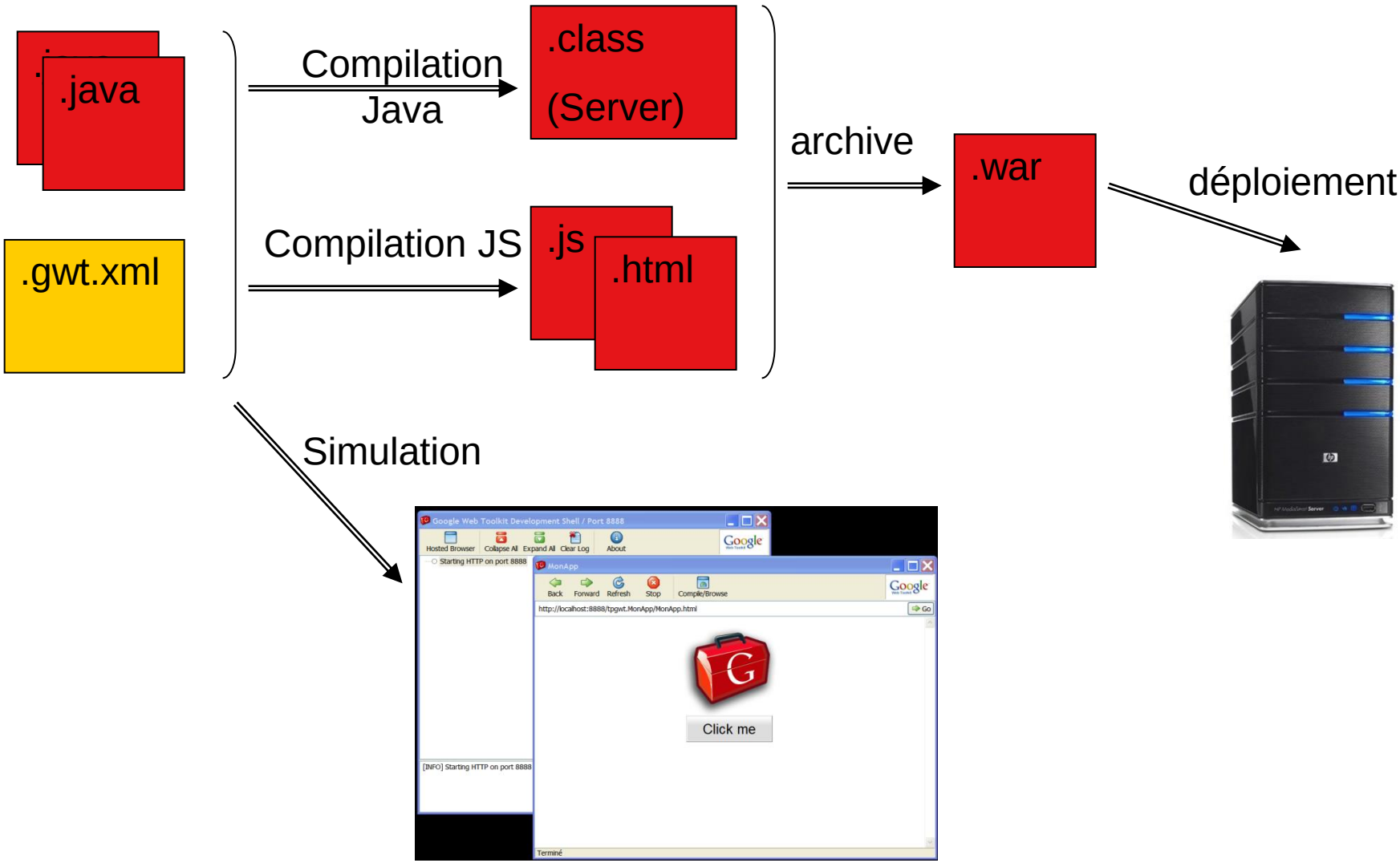
3 modes d'utilisation

- **Developpement ("Hosted") : application exécutée en tant que bytecode Java :**
  - **Permet de faciliter le code-compile-test-**debug
  - **Debug pas à pas, points d'arrêt**
  - **IDE Eclipse**

- **Web : Le navigateur lit simplement le code JS généré par le compilateur GWT et l'interprète naturellement**

- **Superdev mode**

Processus de développement



4 stratégies de test d'une application GWT

- **Test manuel de l'IHM**
  - On clique de partout dans l'application compilée et déployée
  - Pour chaque navigateur ☹
  - Couteux en temps de réalisation ☹
- **Test unitaire (composant simple uniquement)**
  - Exécution du test en Java avec le simulateur GWT (lent)
  - Utilisation obligatoire de JUnit avec la classe GWTTestCase
- **Test de la logique métier SANS la partie Widget**
  - Nécessite de structurer l'application pour isoler la partie Widget
    - ➔ Utilisation du pattern MVP pour l'organisation du code : complexe
  - Creation de "mock" : la partie Widget, des évènements bus ...
  - Exécution en Java/JUnit classique ➔ Rapide, TNR
- **Test de scénarios**
  - Enregistrement de scénario avec Selenium IDE (+codage id JS)
  - Exécution du scénario avec WebDriver (headless)

Les tests unitaires avec Junit / GWTTestCase

- **Comment ?**
  - Ajouter les libs JUnit au projet
  - Hériter de la classe GwtTestCase
  - Implémenter getModuleName() pour récupérer son module
    package + nom du fichier .gwt.xml)
  - Utiliser des assertions pour tester le composant
- **Avantage : Simple à mettre en œuvre**
- **Inconvénient :**
  - **S'exécute dans le simulateur ➔ lent**
  - **Dépend de l'organisation graphique**
    ➔ Utilisable que pour des petits widget

```
public class MyTest
    extends GWTTestCase {

  public String getModuleName() {
       return 'sopra.gwt.tptest.TpTest';
    }

  public void testA() {

       MyWidget w = new MyWidget();
        final String s = 'Toto';
        w.setName(s);

        assertEqual(s, w.getName());

   }

}
```

GWT et les applications mobiles

- **Génération de JS compatible navigateurs mobiles**
- **3 Frameworks conseillés :**
  - **MGWT : Widgets spécifiques aux OS mobiles**
    - OS mobile :
      - Iphone, retina (for iPhone >4)
      - Ipad, ipad_retina
      - desktop
      - Android, android_tablet
      - Blackberry
    - https://code.google.com/p/mgwt/wiki/GettingStarted
    - Basé sur MVP
  - **GWTPhoneGap : Encaspulation de la librairie JS PhoneGap**
  - **Singular**
- **Si application pour mobile + desktop Alors**
  - **Utiliser Pattern MVP**
  - **Implémentation de la vue différente pour Mobile / desktop**
  - **Presenter, Model, Serveur … restent identiques**

    ➔ Option de compilation seulement !

Avantages de GWT

- Destiné au développeur Java / J2EE
  - Plus besoin de compétence HTML forte (Mais CSS oui)
  - Homogénéité des compétences : Java
  - Approche Composant : réutilisabilité
  - Facilité de développement d'une interface dynamique
  - Exécution mode debug dans l'IDE (Eclipse, Idea)
- Ergonomie web 2.0
- Multi navigateur : plus de développement spécifique IE | FF | Chrome |Opéra
- Bibliothèques de composants importantes car très extensible
- Performances :
  - Diminution des performances serveurs requises
  - Exécution sur le navigateur client
  - Code JS "compacté", génération de ce qui est utile uniquement
- Outils graphique d'édition des IHMs
- Communauté Open source très active : forum
- Mise en œuvre dans les applications Google : Inbox

Inconvénients et limitations

- Risque de consommation mémoire dans le navigateur
- Pas de support de nombre 64 bits en JS:
  - ➔ Un Long est mappé en 2 float
- L'interpréteur JS est mono thread (mais bientôt Web worker de HTML5)
- Pas de chargement dynamique de classes / introspection
- Pérennité du canevas :
  - volonté de Google
  - Complexité du compilateur Java➔JS
- Compilateur lent pour les gros projets
- Les moyens de vraiment tester sont complexes

GWT en ce moment et après

- Google n'est plus seul à tenir les reines ➔ consortium
  - + Vaadim
  - + …
- Version actuelle : GWT 2.7
  - Compilation incrémental
  - Industrialisation du SuperDevMode
  - CSS 3
- A venir : GWT 3.0
  - Support du langage Java 8
  - Closure, Lambda expression
  - Objectif : concurrencer l'expréssivité de JS

La concurrence …

- **Framework JS + HTML 5**
  - **Anuglar JS (Google aussi)**
  - **Jquery, bootstrap …**
  - **Dart ?**
  - **…**
- **Rubrique nécrologie :**
  - **Flex (Adobe)**
    - Lecteur flash nécessaire (☹ mobile)
    - Description xml de l'interface + langage action script
    - Seul le SDK en open source
    - Outils & Bibliothèques de widget/chart sont payants
  - **Microsoft Silverlight(C# , VB.NET)**
    - Environnement de dev complet (Visual Studio)
    - Composant .NET à installer sur les postes clients
    - Fonctionne sur Windows uniquement (linux/mono ☹)

Références

- **Gwt**
  - Le site officiel : http://gwtprojet.org
  - LA DOC : http://code.google.com/intl/fr/webtoolkit/doc/latest/DevGuide.html
- **Librairies**
  - GXT : http://extjs.com/products/gxt/
  - Gwt-phonegap : http://code.google.com/p/gwt-phonegap
  - MGWT : http://www.m-gwt.com