

Google Web Toolkit

Jeudi 14 Mai 2009

Anthony Quinault

aquinault@cadesoft.com



Sommaire

- ★ Présentation
- ★ Extensions disponibles
- ★ Nouveautés de la version 1.6
- ★ Mise en place de l'environnement
- ★ Création de l'IHM
- ★ Appel distant à un service
- ★ Retour d'expérience sur la mise en oeuvre
- ★ Les bonnes pratiques
- ★ Références
- ★ Conclusion
- ★ Déployer une application dans l'AppEngine

Présentation (1/2)

Historique

Créé par Bruce Johnson

Projet acheté par Google pour ses devs internes

OpenSource fin 2006

Forte communauté

Difficulté d'écrire du JavaScript

Outils immatures

Gestion des navigateurs difficiles

Compétences rares sur le marché

Présentation (2/2)

La solution est de générer du code Javascript à partir du code Java

- ➡ Java très utilisé en entreprise
- ➡ Emulation d'une partie de la JRE (pas de réflexion)

S'appuyer sur des outils existants du monde Java pour coder/
debugger/tester

- ➡ Eclipse, Netbeans, JUnit, Maven, Ant
- ➡ Auto-completion, refactoring

Fonctionnalités

- ➡ Internationalization, Deferred Binding, CSS compliant
- ➡ Compatible Java 1.5, Gestion de l'historique

Extensions disponibles

- ★ Google Gears
- ★ Google Map
- ★ Google Visualization
- ★ Drag and Drop
- ★ PureMVC
- ★ Plugin Maven
- ★ Plugin Google pour eclipse
- ★ Ext GWT

Nouveautés de la version 1.6

Optimisation du compilateur

Nouvelle structure de projet

- Application standard de type java «WebApp»
- Utilisation du «web.xml» pour la déclaration des RPC

Mode Hosted

- Jetty à la place du tomcat
- Rafraîchissement dynamique du code serveur

EventHandler

- Une seule méthode par Event, ex: onClick(clickEvent)

Nouveaux widgets

- DatePicker, DateBox, LazyPanel

Mise en place de l'environnement (1/3)

Télécharger GWT et/ou le plugin pour eclipse

- <http://code.google.com/intl/fr/webtoolkit/>

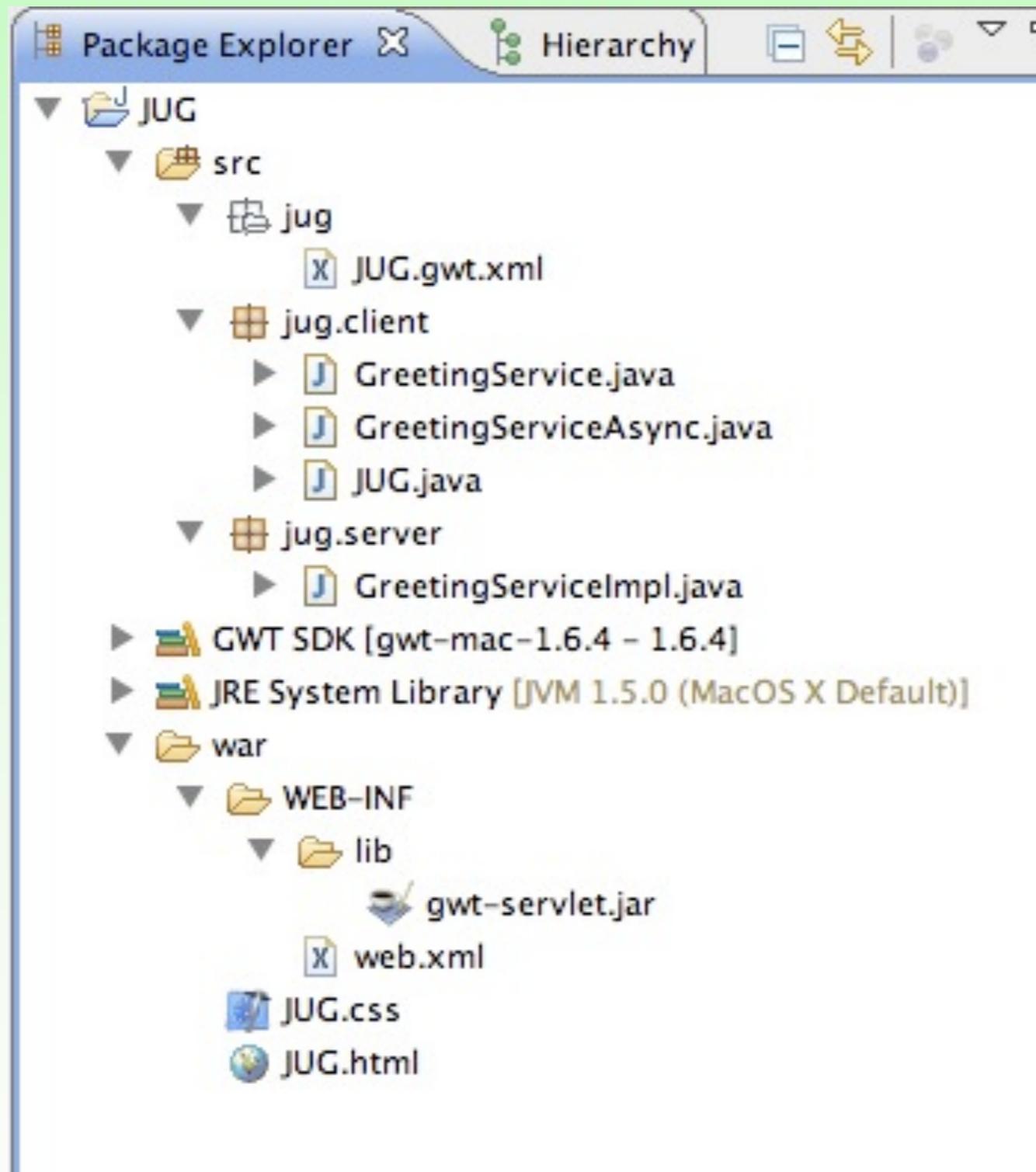
Création d'une application sans eclipse

- `webAppCreator -out MyApplication
com.mycompany.MyApplication`
- `ant hosted`

Création d'une application sous eclipse

- Import de l'application créé par webAppCreator
- Ou dans la barre d'outil, cliquer sur 

Mise en place de l'environnement (2/3)



Mise en place de l'environnement (3/3)

Module principal : `monApplication.gwt.xml`

Hosted mode (développement)

- L'application s'exécute comme du Java byte code à l'intérieur de la JVM
- Mode permettant le debuggage

Web mode (production)

- L'application s'exécute en Javascript, Html et CSS
- Le code est optimisé (exécution, taille)
- Compatible avec tous les navigateurs

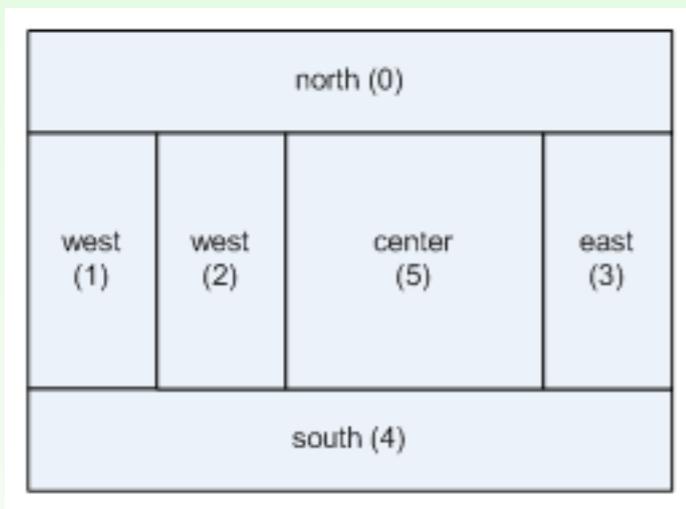
Création de l'IHM (1/4)

Programmation événementiel similaire à Swing

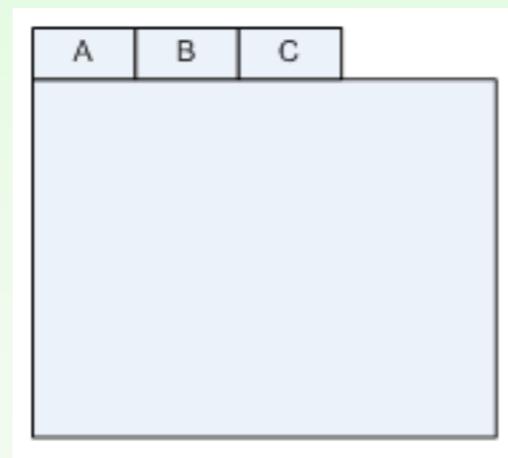
- Listeners Java : FocusListener, MouseListener,...

Panels :

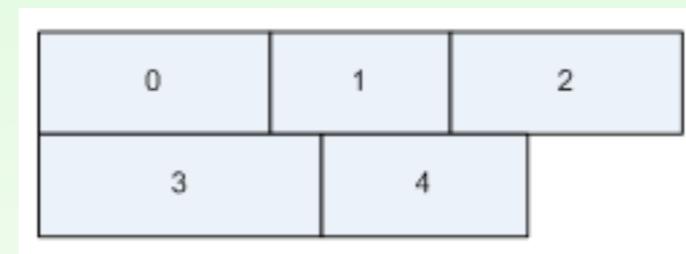
DockPanel



TabPanel



FlowPanel



et pleins d'autres : PopupPanel, StackPanel, HorizontalPanel,
VerticalPanel, VerticalSplitPanel,....

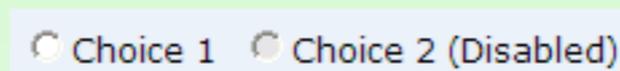
Création de l'IHM (2/4)

Widgets :

Button



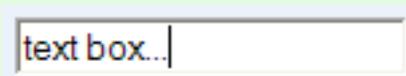
RadioButton



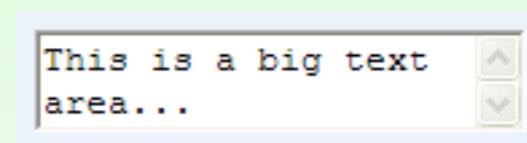
DatePicker



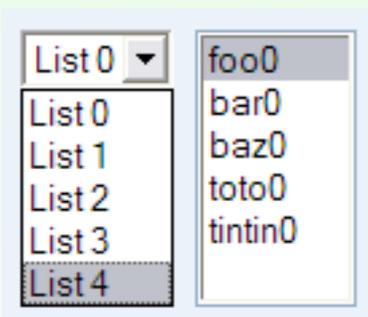
TextBox



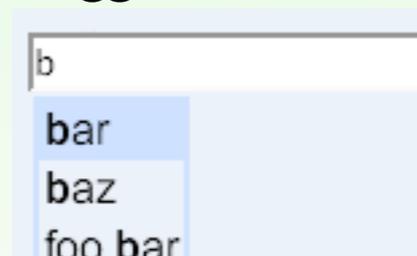
TextArea



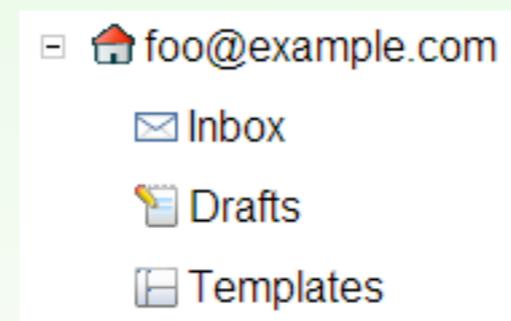
ListBox



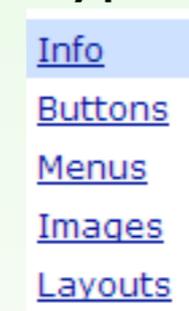
SuggestBox



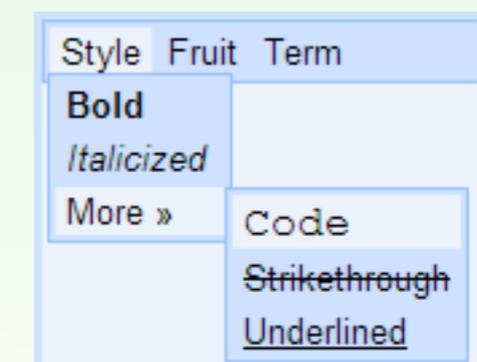
Tree



Hyperlink



Menubar



et pleins d'autres : PushButton, CheckBox, ToggleButton, PasswordTextBox
RichTextArea, Table, TabBar, DialogBox....

Création de l'IHM (3/4)

Création d'un Widget personnalisé

➡ Composition de widgets GWT

```
class DisplayBox extends Composite
{
    public DisplayBox(String header, String data)
    {
        VerticalPanel widget = new VerticalPanel();
        initWidget(widget);
        widget.addStyleName("demo-Composite");

        Label headerText = new Label(header);
        widget.add(headerText);
        headerText.addStyleName("demo-Composite-header");

        Label dataText = new Label(data);
        widget.add(dataText);
        dataText.addStyleName("demo-Composite-data");
    }
}
```



Création de l'IHM (4/4)

Wrapper de code Javascript en utilisant JSNI

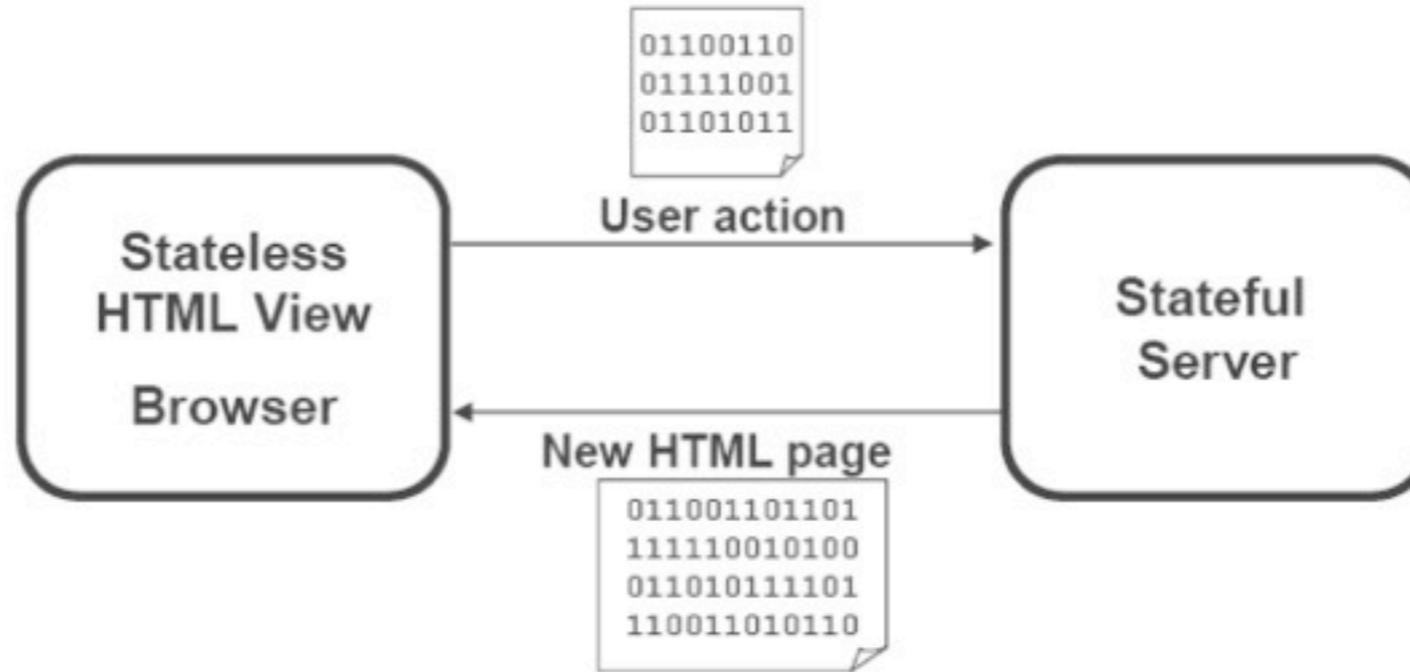
- JSNI : JavaScript Native Interface

Intégration avec des bibliothèques JavaScript externes
Interagir directement avec du JavaScript à partir de
Java et vice-versa

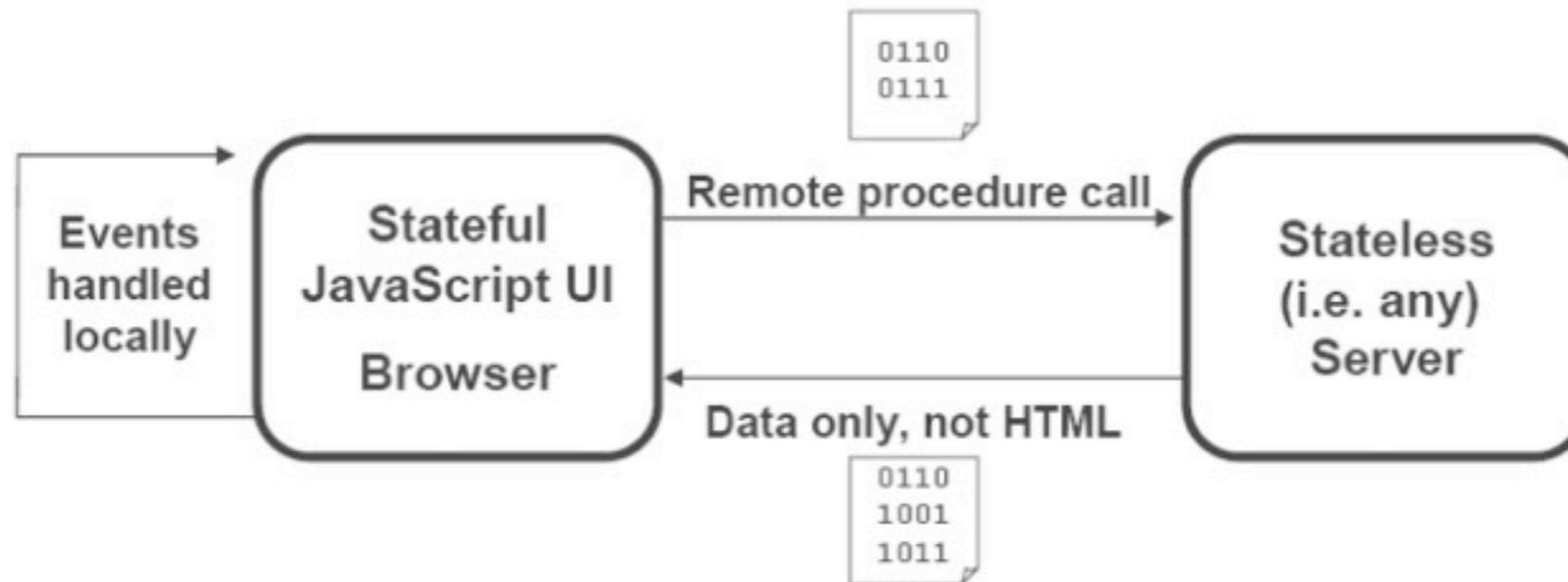
```
public static native void alert(String msg) /*-{  
    $wnd.alert(msg);  
}-*/;
```

Appel distant à un service (1/2)

Traditional HTML



Ajax



Appel distant à un service (2/2)

Plusieurs implémentations disponibles

GWT RPC (Remote Procedure Call), appel de procédure distance

- Serialisation d'objet Java entre le code client et le code serveur
- Uniquement dans un conteneur de Servlet

JSON, XML via HTTP

- Echange de données au format Json ou XML
- Mashup

Retour d'expérience sur la mise en oeuvre (1/5)

Réalisation d'une étude pour un client Niortais sur le choix d'un outil de développement d'IHM

- ✓ Client léger avec ergonomie client/serveur
- ✓ Avoir de la productivité (dev/test/debug)
- ✓ Utilisation d'un IDE pour le design et enchaînement des pages
- ✓ Réutilisation des composants métiers Java existants
- ✓ Pouvoir créer des templates de composants
- ✓ Permettre la gestion du clavier
- ✓ Multi-navigateurs
- ✓ Interfaçable avec l'atelier de développement existant
 - ▶ eclipse, maven, hudson
- ✓ Pouvoir trouver des ressources sur le marché Niortais

Retour d'expérience sur la mise en oeuvre (2/5)

Les critères mis en avant :

Développement

- Intégration avec eclipse
- Richesse de l'atelier de développement
- Extensibilité, disponibilité de widgets

Architecture

- Séparation métier / présentation
- Respect de l'architecture MVC
- Compatibilité WAN
- Robustesse, performance

Technique

- Richesse des composants

Industrialisation

- Productivité
- Pérennité
- Effort de formation

Support produit

- Documentation
- Formation disponible

Produits fini

- Potentiel ergonomique

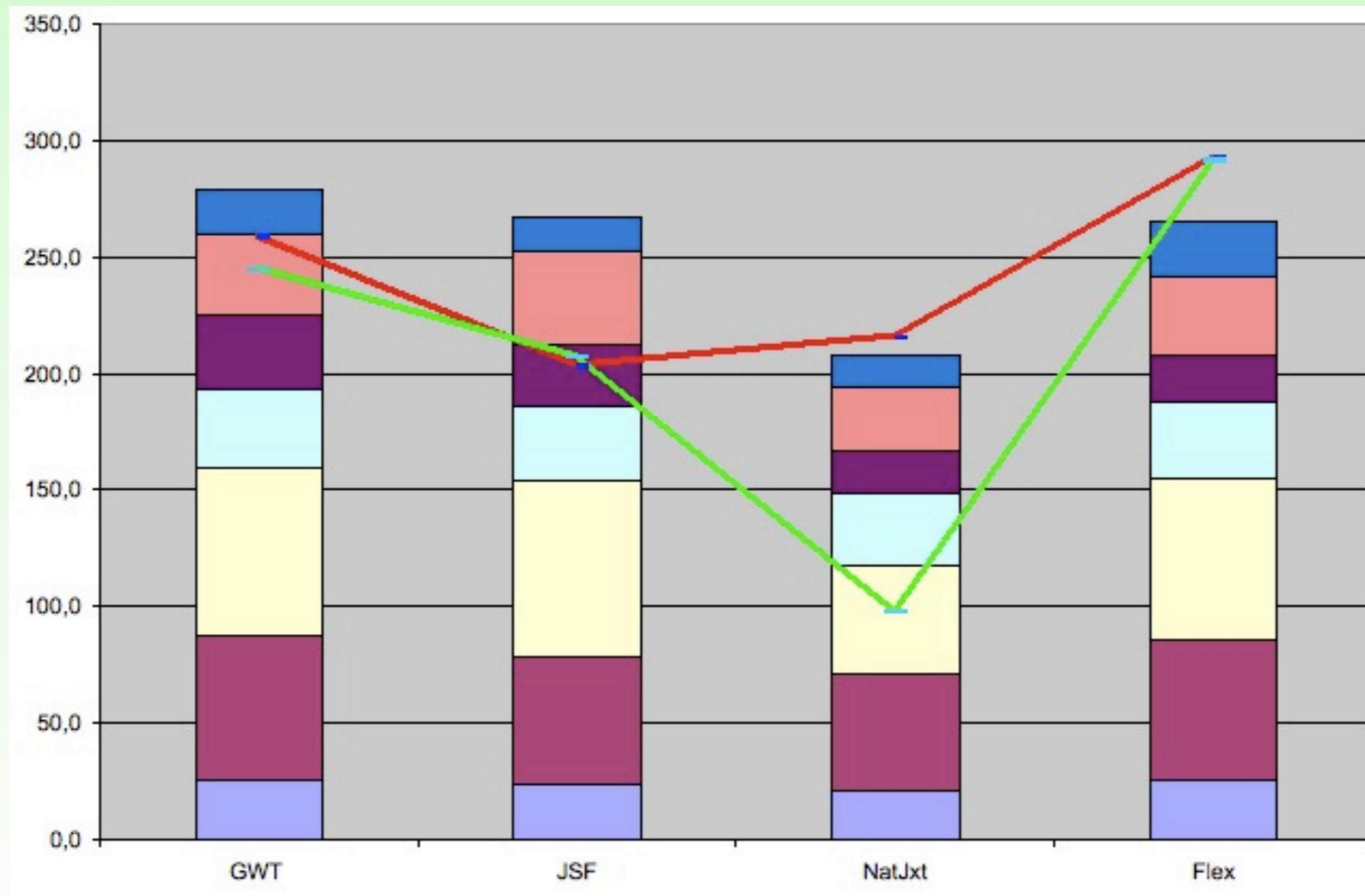
Retour d'expérience sur la mise en oeuvre (3/5)

Etude portant sur 4 outils :

		GWT	JSF	NatJxt	Flex
Solution	Framework MVC	GWT	RI v1.2 (Implémentation sun) seam v1.2 (conversation)	NatJxt	PureMVC
	Bibliothèque supplémentaire	GWT Ext 0.9.2	Icefaces		
	Serveur frontal	WAS 6.1	WAS 6.1	WAS 6.1 ou tomcat	WAS 6.1
	JDK	1.5	1.5	1.5	1.5
Développement	IDE	Eclipse 3.2	Eclipse 3.2 WTP 1.5	Eclipse 3.2	Eclipse 3.2
	Plugin	GWT Designer 3.0.0.0	plug-in JSF	NatJxt	Plugin Flex 2

Retour d'expérience sur la mise en oeuvre (4/5)

Résultat de l'étude GWT et Flex au coude à coude



Retour d'expérience sur la mise en oeuvre (5/5)

Réalisation d'un POC (Proof of Concept) :

- Lecture et écriture d'une fiche client
 - Sollicitation d'un service existant
- Utilisation d'une applet java CTI
- Création d'un widget métier
 - Recherche d'une ville / Département / Pays

GWT finalement est choisi pour sa :

- Simplicité d'utilisation
- Bonne intégration dans les outils existants
- Forte communauté OpenSource
- Gratuité

Les bonnes pratiques

Développer toute l'application en GWT

- Mélanger Struts ou JSF avec GWT n'est vraiment pas idéal

Développer ses widgets en Java

- Éviter au maximum l'utilisation de JSNI (wrapper Javascript)
- Favoriser de la composition de widgets natifs

Utiliser Maven pour industrialiser ses développements

Structurer son application en différents projets eclipse

- Gérer les dépendances de projets avec Maven

Penser à regarder les frameworks existants et l'incubator

Références

★ Homepage GWT

<http://code.google.com/intl/fr/webtoolkit/>

★ Forum GWT

<http://groups.google.com/group/Google-Web-Toolkit>

★ OnGWT

<http://www.ongwt.com>

★ PureMVC pour Java/GWT

http://trac.puremvc.org/PureMVC_Java_MultiCore/

★ La démo de GWT

<http://gwt.google.com/samples/KitchenSink/KitchenSink.html>

★ Le showcase de GWT

<http://gwt.google.com/samples/Showcase/Showcase.html>

Conclusion

GWT c'est facile, efficace et beau!

Merci à :

- ★ L'équipe GWT
- ★ Poitou-Charentes JUG
- ★ Serli
- ★ Didier Girard
- ★ Cliff Hall
- ★ La communauté
- ★ Google

Déployer une application dans l'AppEngine

L'Appengine est :

- ★ Une infrastructure hébergeant des applications Web
- ★ Un environnement d'exécution Java et python
- ★ Un espace de persistance de données de 500Mo

➔ Démonstration



+



QUESTIONS / REPOONSES

