



Almaden Services Research

A 'new' way to program dynamic web applications

E. M. (Max) Maximilien,

<http://maximilien.org>

Agenda

- **The past**
 - Overview of typical AJAX development
 - Problems and opportunities
- **The present: Google Web Toolkit (GWT)**
 - Architecture of a GWT project
 - Development cycle and tools
 - Widget UI library
 - Simple browser-to-server RPC
 - JavaScript native interface (JSNI)
 - Calling Web services
 - Deployment and other tricks
- **The good, the bad, and the ugly**
- **Terms, agreement, and licensing (IANAL)**
- **Other thoughts**

The past: typical AJAX web application development

- **AJAX – asynchronous JavaScript and XML**
 - Client-side JavaScript (JS) to asynchronously fetch data
 - To dynamically modify client page using DOM API
- **Typical development stack**
 - Java
 - J2EE compatible server, e.g., Tomcat or WebSphere application server
 - Model View Controller (MVC) web application framework, e.g., Struts
 - Java Server Pages (JSP) templates, tag libraries, and scriptlets for Views
 - Ruby
 - Similar except use Rails, RHTML, and JS libraries
 - RJS is an improvement (still in development)
 - Embed JavaScript for AJAX functionality in JSP
 - JS libraries such as *scriptalicious*, *prototype*, *effects*, *dojo*, Yahoo! UI, and others
 - Cascading Style Sheets (CSS) for View styling

The past: problems and opportunities

■ Problems

- Multiple languages and frameworks across stack
- Mixing JS, JSP tags, Java, XML, and other config files leads to
 - Maintenance nightmare
 - Less modularization
 - Less reusable code (Views)
 - Breaks MVC principles
- Complicated debugging and testing (different browsers)

■ Opportunities

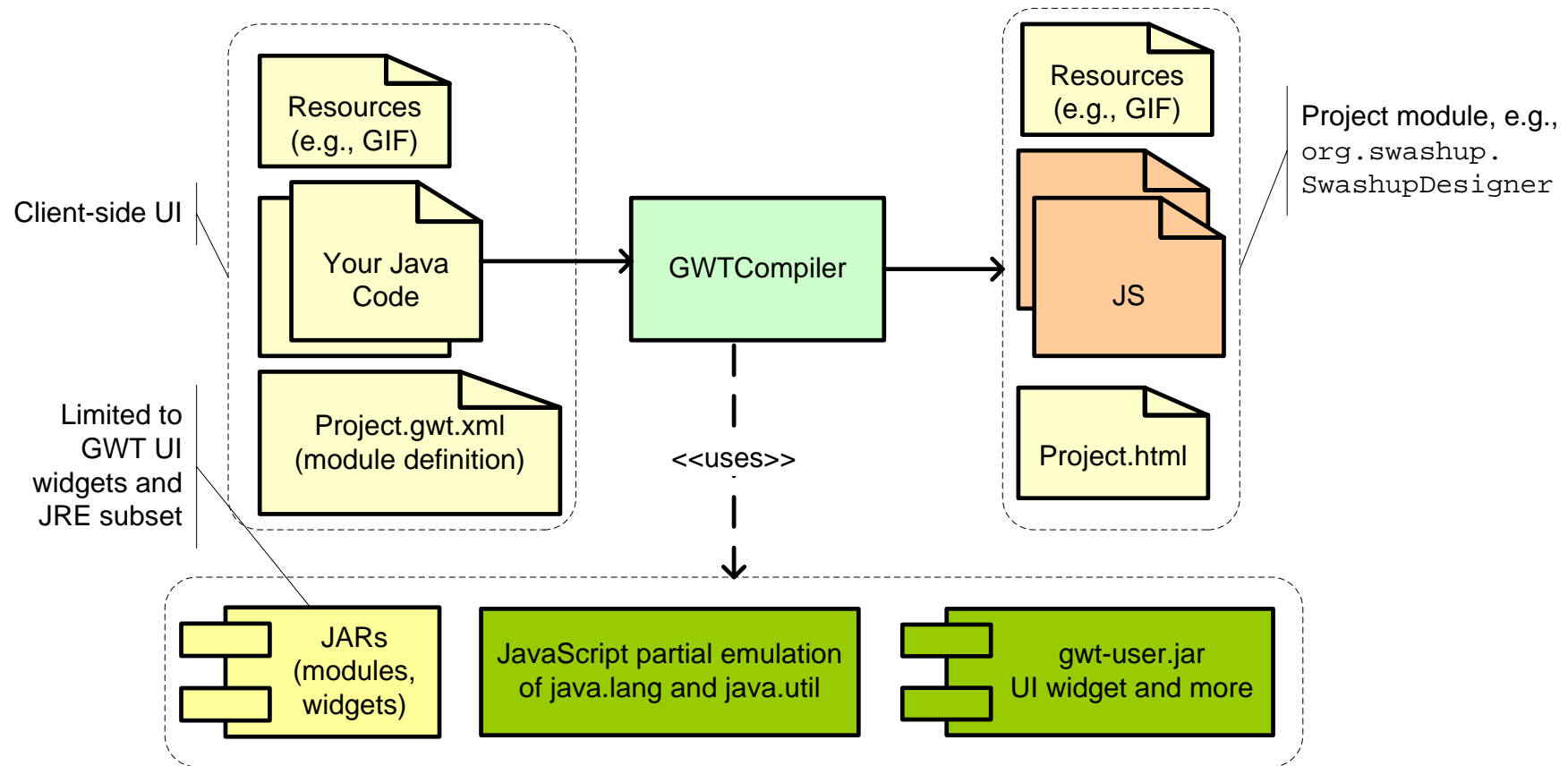
- Simplify AJAX web development with one language
- Address maintenance, debugging, and modularization issues
- Abstract client-side JS as *browser assembly code*!
- The browser as another targeted “device”



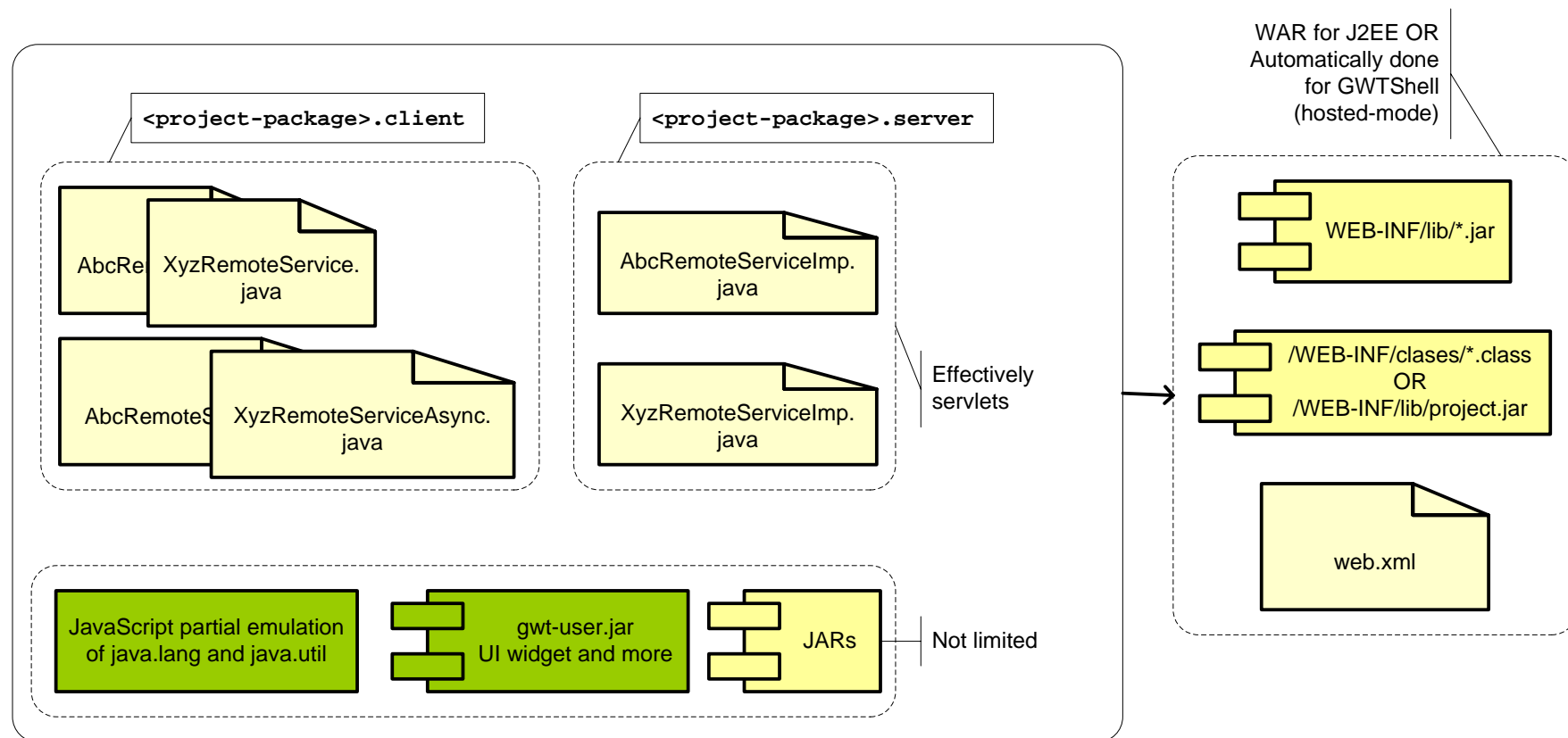
The present: Google Web Toolkit (GWT)

- **Java-to-JavaScript compiler**
- **Shell to test and debug modules without compilation**
- **Partial JS emulated versions for JRE packages**
 - `java.lang.*`
 - `java.util.*`
- **UI widget library (not AWT and not Swing)**
- **Simple asynchronous browser-to-server RPC**
- **Development tools, e.g., `applicationCreator`, `projectCreator`, and `junitCreator`**
- **Not new! Microsoft ASP.NET is similar and released years ago**
 - However, limited to .NET and IE
 - Not sure of current status (likely more open)
- **Seaside (based on Squeak Smalltalk dialect) is a Smalltalk-based OO framework for coding web applications**
- **Any other similar compilers and frameworks for other languages?**

GWT: architecture of a GWT project (client side)



GWT: architecture of a GWT project (server side)



GWT: development cycle and tools

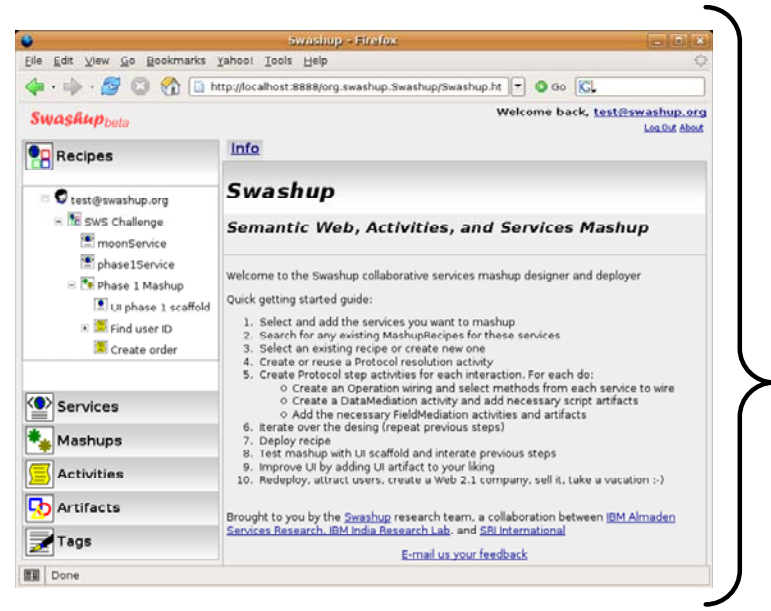
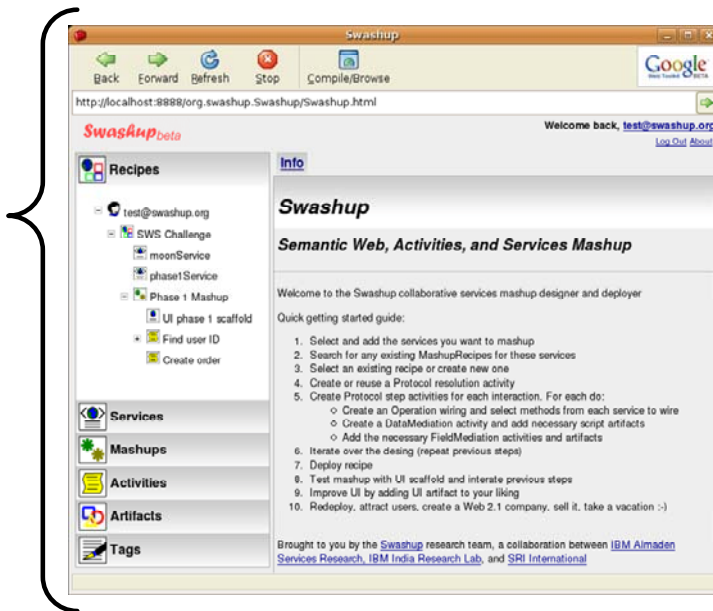
- **Hosted-mode**

- GWTShell
- Quick code-test cycle
- Style with CSS

- **Web-mode**

- GWTCompiler
 - Module (Java) compiled as JS and HTML
- Style with CSS
- Deploy into J2EE compliant server
- Test using IE, FireFox, Safari, and Opera as client browser

GWT: development and tools (cont.)



Hosted-mode
(GWSHELL)

GWTShell

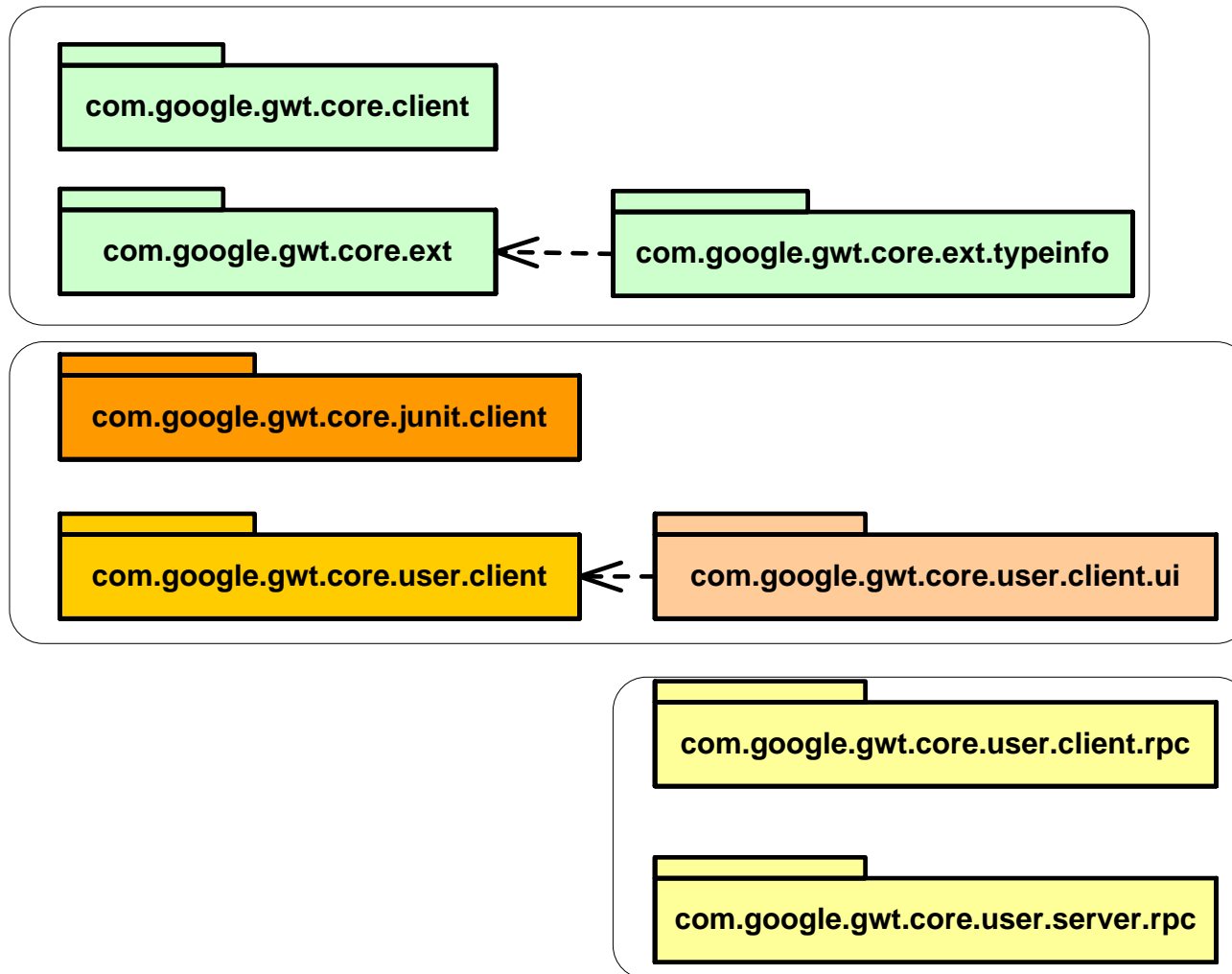
GWTCompiler

Hosted-mode
(GWSHELL)

gwt-user.jar
(JRE emulation and UI widget)

JAR

GWT: package diagram



GWT: UI widget library, listeners, and other

- **Widgets**

Button, CheckBox, TextBox, PasswordTextBox, TextArea, DialogBox, FlexTable, Label, ListBox, MenuBar, MenuItem, Tree, TreeItem, Widget

HTML, HTMLTable, Hyperlink

- **Composite to create custom composed widget**

- **Panels**

Panel, SimplePanel, VerticalPanel, HorizontalPanel, DockPanel, DeckPanel, PopupPanel, ScrollPanel, RootPanel

- **Listeners**

ChangeListener, ClickListener, KeyboardListener, MouseListener, FocusListener, LoadListener, TreeListener, TabListener

- **Other interfaces**

EntryPoint, IsSerializable, RemoteService, AsyncCallback, Has<XYZ>, Sources<XYZ>Events

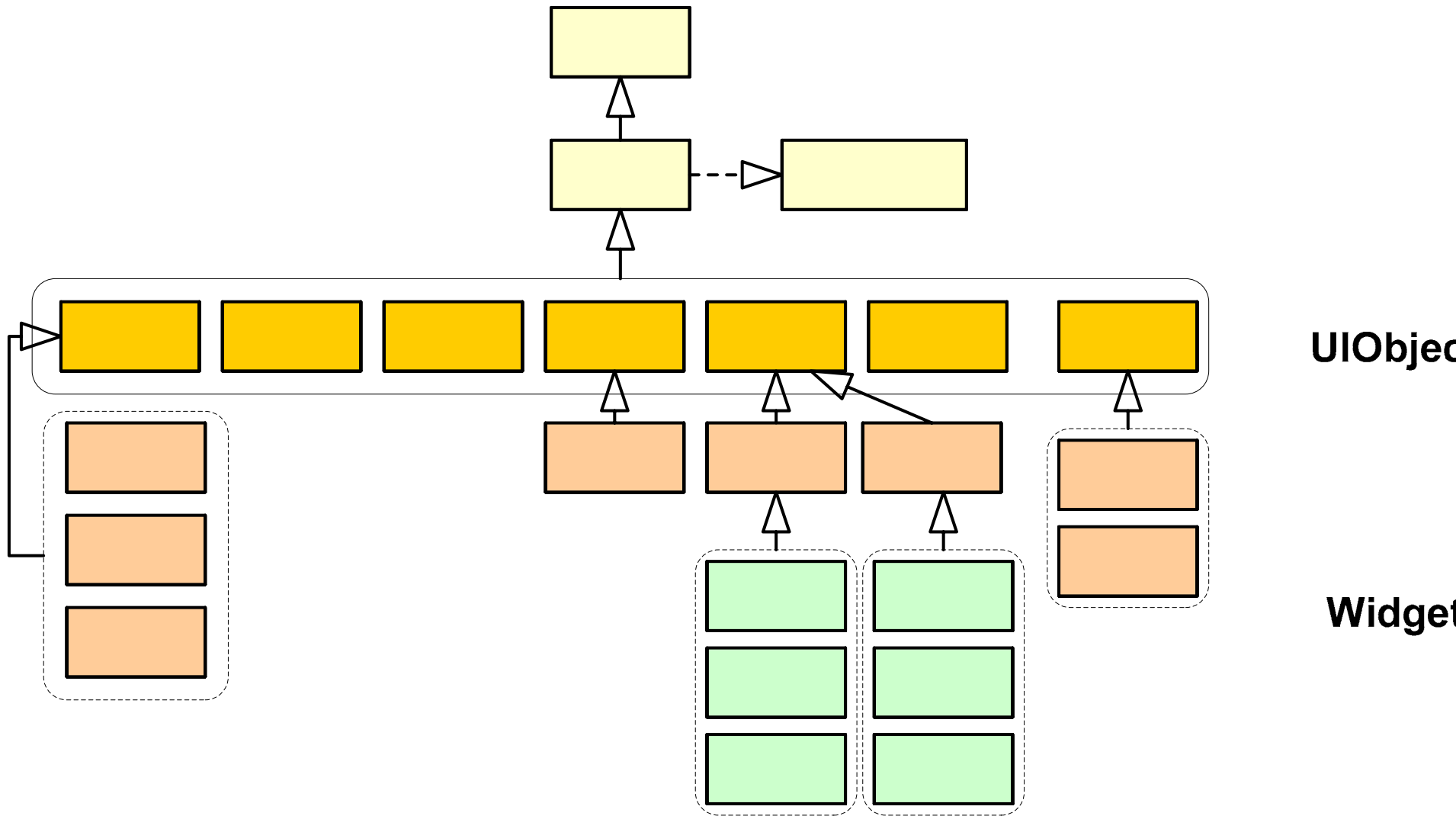
- **Other classes**

GWT, Window, History, Timer, Random, DOM, Cookies, JavaScriptObject, and others

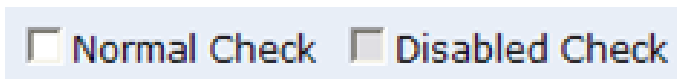
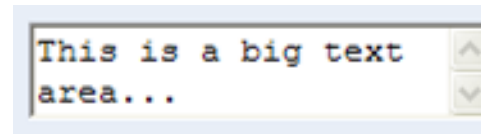
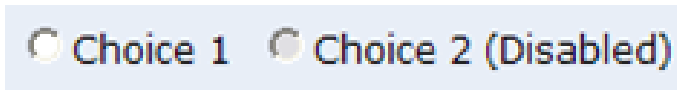
- **Exceptions**

InvocationException, SerializableException

GWT: UI widget library, listeners, and other (cont.)



GWT: UI widget library, listeners, and other (*cont.*)



Hyperlink

[Info](#)

[Buttons](#)

[Menus](#)

[Images](#)

[Layouts](#)

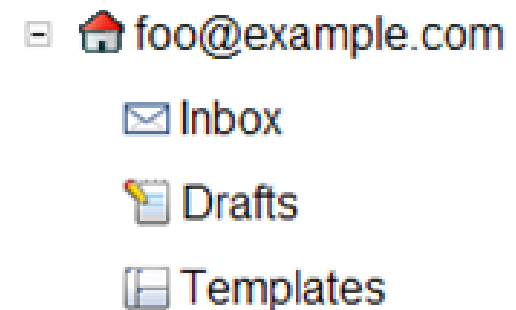
List



Menu



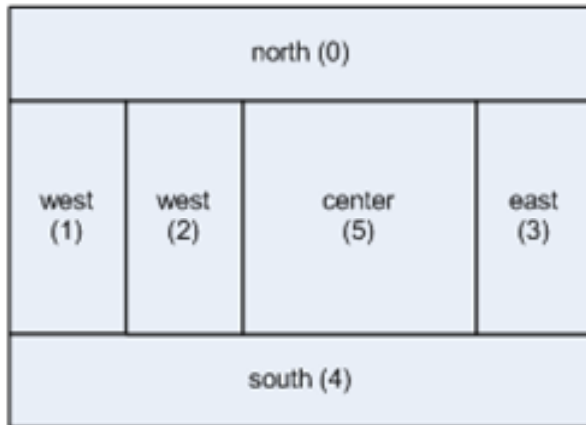
Tree



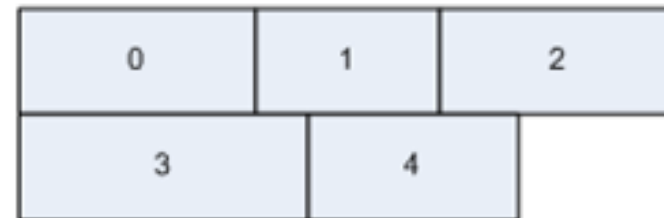
Pictures from GWT web site: <http://code.google.com/webtoolkit>

GWT: UI widget library, listeners, and other (*cont.*)

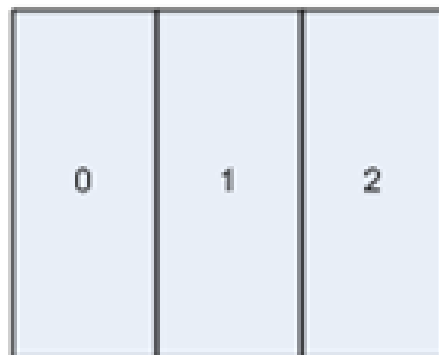
DockPanel



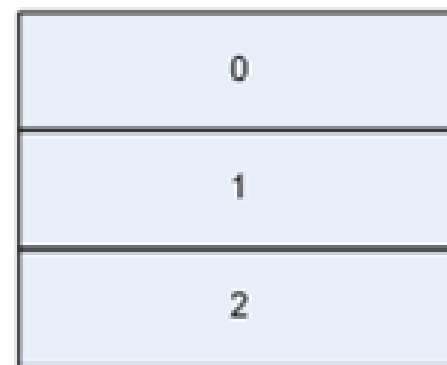
FlowPanel



VerticalPanel



HorizontalPanel



Pictures from GWT web site: <http://code.google.com/webtoolkit>

GWT: simple browser-to-server RPC mechanism

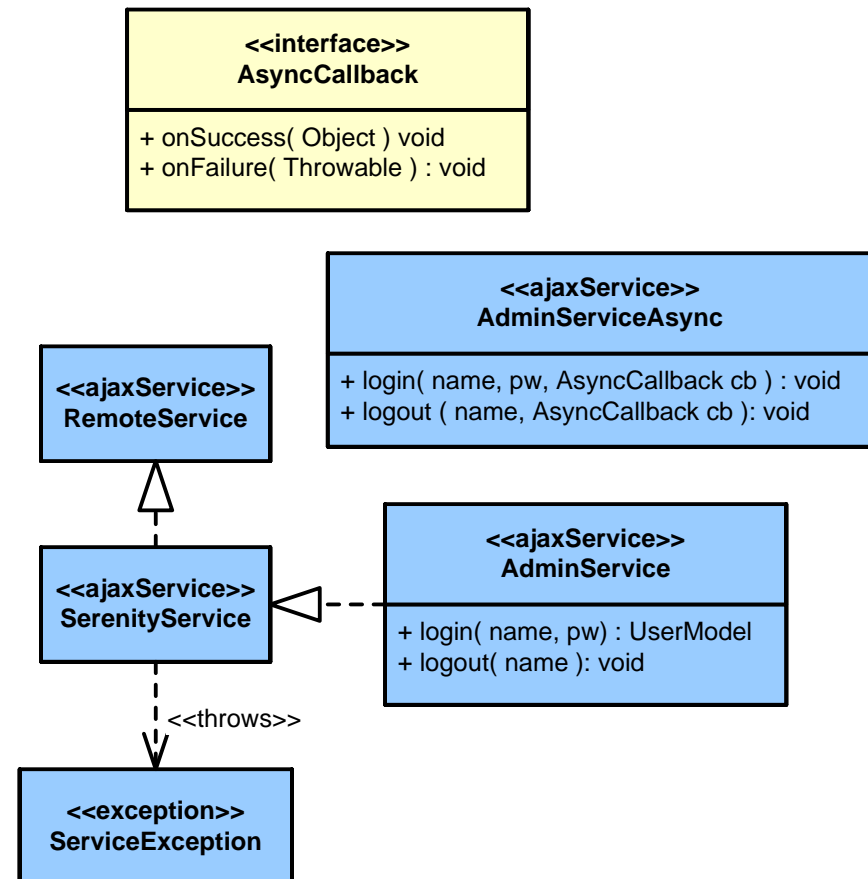
▪ `<project-package>.client`

- `XYZService` interface
 - Extends `RemoteService`
 - Can throw custom exception
 - Primitive types in signature
 - `IsSerializable` class parameters
- `XYZServiceAsync` interface
 - Matching service methods
 - Same signature but:
 - Callback future object parameter
 - `void` return type

▪ Model objects

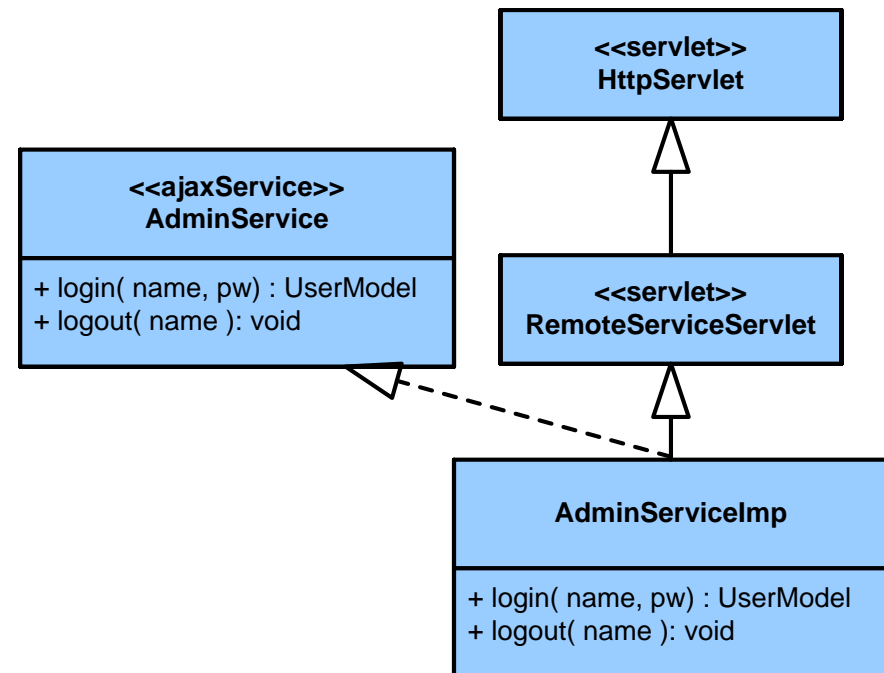
- Must implement `IsSerializable`
- Use `transient` for non-serializable members

▪ Restricted to JRE emulation subset



GWT: simple browser-to-server RPC mechanism (*cont.*)

- `<project-package>.server`
 - `XyzServiceImp` class
 - Extends `RemoteServiceServlet`
- **No restrictions on imports**
- **Make call to back-end tiers**
 - Web services
 - Databases
- **JAR dependencies**
 - Add to `GWTShell` and `GWTCompiler` scripts
 - Include in WAR file



GWT: simple browser-to-server RPC mechanism (*cont.*)

```
//Example client of AdminService.login
AdminServiceAsync adminAsync = getAdminServiceAsync();
//...
String name = nameTextBox.getText();
String password = passwordTextBox.getText();
adminAsync.login( name, password, new AsyncCallback() {
    public void onSuccess( Object result ) {
        UserModel user = (UserModel)result;
        //Do post-login processing
    }
    public void onFailure( Throwable throwable ) {
        //Process login failure
        Window.alert( "Error login, invalid user name or password" );
    }
});
```

GWT: JavaScript native interface (JSNI)

- **Java native code extension**

- Use `native` keyword on method
- Call Java methods and other JS code
- Pass JS object (`JavaScriptObject`) to Java

- **Example:**

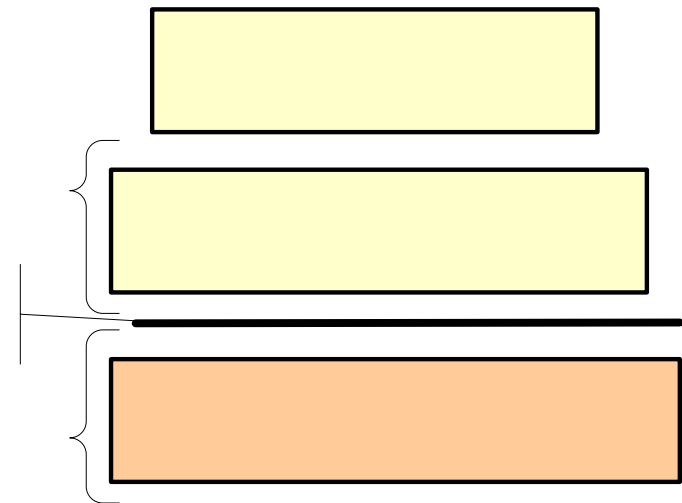
```
public class Native
{
    public static native void callNative()
    /*- { $wnd.alert("Hello from JavaScript!"); }-*/;

    public native void callJavaNative()
    /*- { this@com.google.gwt.client.
        Window::alert(Ljava/lang/String;
        ("Hello called from JavaScript!")); }-*/
}
```

- **Customize generated widget code**
- **Fine-grained control over client JS**

GWT: calling web services (back-end)

- **Include web services (WS) proxies JARs**
- **Define client model objects**
- **Cache (optional)**
 - AJAX server cache
 - Browser client cache
- **Use browser-to-server RPC**
 - Move data from server cache to client cache
 - Browser-to-server RPC call equivalent to:
 - One WS call
 - Aggregate multiple WS calls



GWT: deployment and other tricks

■ GWTShell

- Uses embedded version of Tomcat
- Add dependency JARs in `<Project>-compile` and `<Project>-shell` scripts (generated by `applicationCreator`)

■ Deploying as WAR

- Include `gwt-user.jar`
- Include other JAR dependencies in `WEB-INF/lib/`
- Include `www/<project-module-directory>/*` in WAR
- Add RPC service entries in `web.xml`

■ Mix environment

- Add JAR dependencies into WAR (as above)
- Add `<module> ... </module>` in JSP (or other) template header
- Add RPC service entries in `web.xml`

The good, the bad, and the ugly

- **The good**

- All things mentioned earlier
- Java development tools, e.g., Eclipse, IDEAJ, and others
- Excellent browser-to-server RPC
- Performance seems on par with custom JS code
- Ability to hand-craft JS code with JSNI

- **The bad**

- Layouts and panels and some widgets have bugs
- Some limitations in JRE emulation
- Issues running on Windows XP (not sure of details)

- **The ugly**

- No i18n or accessibility support
- Browser compatibility pretty good but issues with some IE versions
- Licensing terms and agreement – half OSS ☺ and half proprietary ☹

Terms, agreement, and licensing (IANAL)

- **Apache 2.0 license**
 - UI widgets
 - Other libraries
- **No restrictions in generated code (no IP restrictions)**
- **Compiler, tools, and shell under different terms and agreement**
- **Cannot repackage tools**
- **Shell “pings” home; although Google claims only for usage statistics and update purposes...**
- **What’s in it for Google?**

Other thoughts

- **GWT and the likes**
 - Paradigm shift in web development
 - Unified and streamlined development
 - Needed if AJAX is to be widely successful
 - Maintenance, debugging, development scalability, and so on
- **The browser as development client platform**
- **GWT is still at version 1.0.21**
 - Expect more stable future versions
 - Better cross-browser support (better IE support)
 - Mac support
 - I18n and accessibility
- **Predict support for advance UI, i.e., drag-and-drop, animation, and so on**
- **Predict widget libraries on top of GWT by Google and third parties**

धन्यवाद

Hindi

多謝

Traditional Chinese

ขอบคุณ

Thai

Спасибо

Russian

Gracias

Spanish

شكراً

Arabic

Thank You

English

Obrigado

Brazilian Portuguese

Grazie

Italian

多谢

Simplified Chinese

Danke

German

Merci

French

நன்றி

Tamil

ありがとうございました

Japanese

감사합니다

Korean