



systinetTM

A MERCURY DIVISION

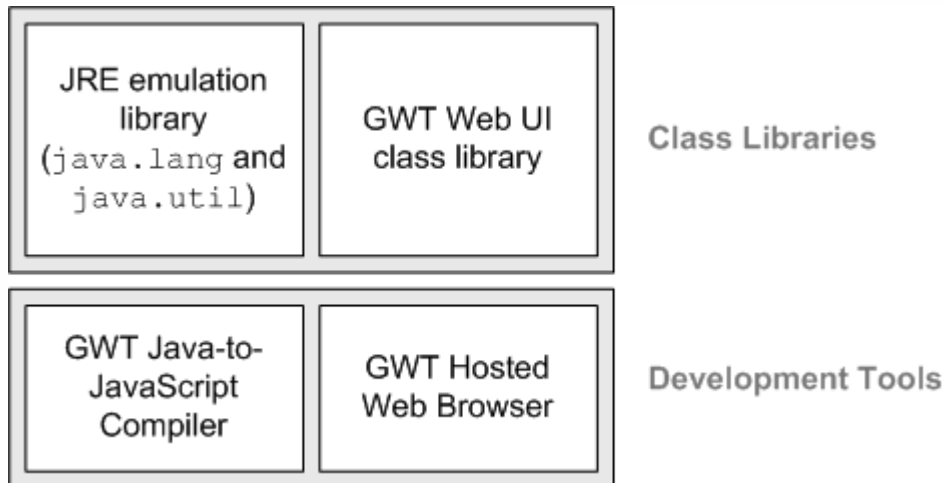


Google Web Toolkit

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

GWT architecture

- ❖ **J2SE emulation library**
- ❖ **Web UI library**
- ❖ **Java2JavaScript compiler**
- ❖ **Hosted web browser**
- ❖ **Asynchronous RPC-over-HTTP implementation**



- **The heart of GWT**
- **Java language semantics support**
- **Java core library emulation support**

GWT Compiler – language support

- **converts Java source into JavaScript**
- **compiles Java source that is compatible with J2SE 1.4.2 or earlier**
- **Java 1.4 language semantics**

Following features are unsupported or ignored

- Java 5 – syntax sugar as Generics, Autoboxing
- Assertions
- Multithreading and Synchronization
- Reflection
- Finalization
- Strict Floating-Point

GWT Compiler – core Java library

- ❖ **small subset of the classes available in the J2SE**
- ❖ **supported classes from package `java.lang` and `java.util`**
- ❖ **differences - Regular Expressions, Serialization**
- ❖ **code is checked against emulation library whenever runs in hosted mode**

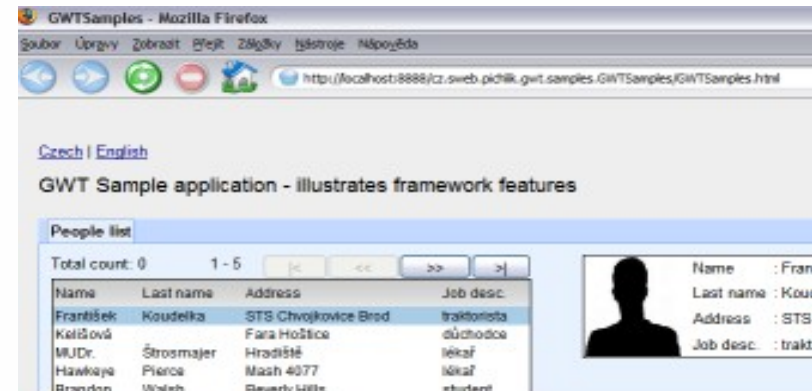
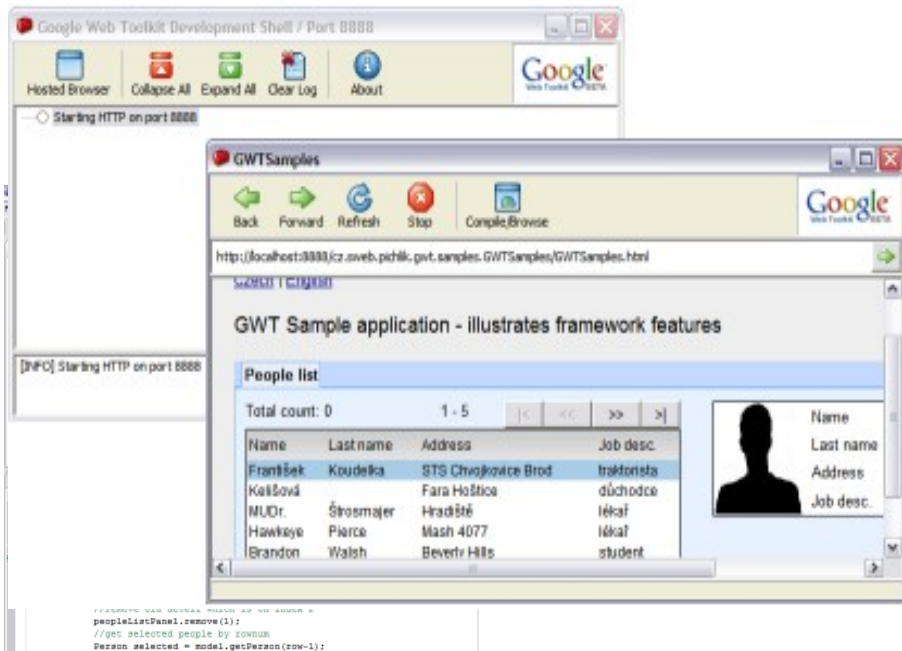
GWT Modes

Hosted mode

- development time mode
- application attached on JVM
- debug from IDE
- GWT development shell

Web mode

- pure JavaScript on client side
- server side part deployed as web application



- **GWT UI classes are similar to Swing/SWT**
- **widgets and panels are client-side Java classes used to build user interfaces**
- **widgets are rendered using dynamically-created HTML**
- **widgets are most easily styled using cascading style sheets (CSS)**
- **widgets publish events using the well-known listener pattern**

Widgets and Panels gallery

Default widgets

Button

Normal Button Disabled Button

CheckBox

Normal Check Disabled Check

PasswordTextBox

.....

RadioButton

Choice 1 Choice 2 (Dis)

TextBox

text box .]

TextArea

This is a big text area...

Hyperlink

Info
Buttons
Menus
Images
Layouts

MenuBar

Style Fruit Term
Bold
Italicized
More » Code
Strikethrough
Underlined

PopupPanel


over [Ludwig von Beethoven](#)
Richard Feynman
Alan John
Richard Feynman
richard@example.com

Table

sender	email
markboland05	mark@example.com
Hollie Voss	hollie@example.com
boticario	boticario@example.com
Emerson Milton	emerson@example.com
Healy Coletta	healy@example.com
Brigitte Cobb	brigitte@example.com
Elba Lockhart	elba@example.com


TabBar

1634 1640 1642 1662



DialogBox

About the Mail Sample



This sample app construction of a GWT's built-in w see how easy it

Close

ListBox

List 0 foo0
List 0 bar0
List 1 baz0
List 2 toto0
List 3 tintin0
List 4

Tree

- foo@example.com
 - Inbox
 - Drafts
 - Templates

Default panels

StackPanel

Mail
Tasks
Contacts

HorizontalPanel

0 1 2

VerticalPanel

0
1
2

FlowPanel

0 1 2
3 4

DockPanel

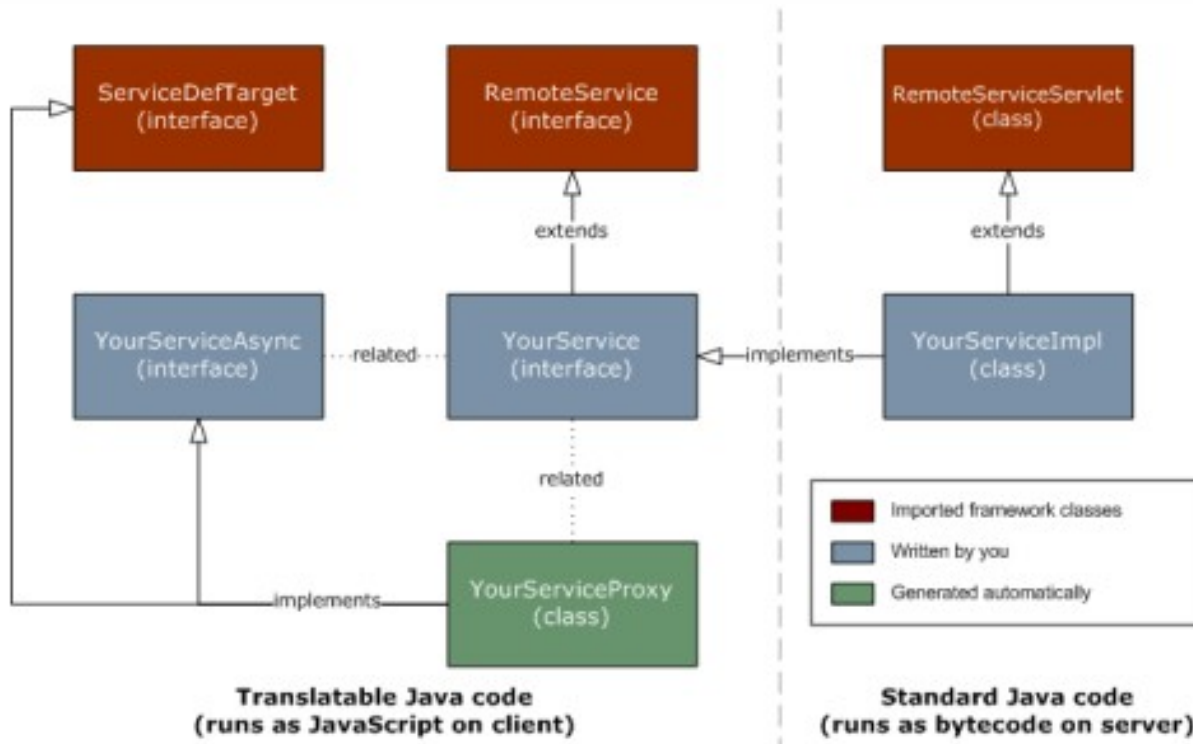
north (0)
west (1) west (2) center (5) east (3)
south (4)

TabPanel

A B C

Asynchronous RPC

- **AJAX concept**
- **built-in serialization/deserialization**
- **exception handling**



RPC example

```
public interface HelloWorldService extends RemoteService{
    public String sayHello();
}

public interface HelloWorldServiceAsync {
    public void sayHello(AsyncCallback callback);
}

public void foo(){
    //(1) Create the client proxy
    HelloWorldServiceAsync helloService = (HelloWorldServiceAsync) GWT.create(HelloWorldServiceAsync.class);

    //(2) Specify the URL at which our service implementation is running.
    ServiceDefTarget endpoint = (ServiceDefTarget) helloService;
    endpoint.setServiceEntryPoint("/helloService");

    //(3) Create an asynchronous callback to handle the result.

    AsyncCallback callback = new AsyncCallback() {
        public void onSuccess(Object helloText) {
            Window.alert((String) helloText);
        }

        public void onFailure(Throwable caught) {
            Window.alert("Ooops something went wrong:" + caught.getMessage());
        }
    };

    //(4) Make the call. Control flow will continue immediately and later
    // 'callback' will be invoked when the RPC completes.
    helloService.sayHello(callback);
}

class HelloWorldServiceImpl extends RemoteServiceServlet implements HelloWorldService {
    public String sayHello() {
        return "Hello world!";
    }
}
```

client code

server code

Application configuration and bootstrap

bootstrap HTML

```
<html>
<head>
  <title>GWTSamples</title>

  <!--
  <!-- The module reference below is the link
  <!-- between html and your Web Toolkit module
  <!--
  <meta name='gwt:module' content='cz.sweb.pichlik.gwt.samples.GWTSamples'>

  <!-- Link to external CSS file
  <link rel="stylesheet" type="text/css" href="GWTSamples.css" />
</head>

<!--
<!-- The body can have arbitrary html, or
<!-- you can leave the body empty if you want
<!-- to create a completely dynamic ui
<!--
<body style="background-color: #EEEEEE">

  <!--
  <!-- This script is required bootstrap stuff.
  <!-- You can put it in the HEAD, but startup
  <!-- is slightly faster if you include it here.
  <!--
  <script language="javascript" src="gwt.js"></script>

  <!-- OPTIONAL: include this if you want history support -->
  <iframe id="__gwt_historyFrame" style="width:0;height:0;border:0"></iframe>
```

App configuration

```
<module>

  <!-- Inherit the core Web Toolkit stuff.
  <inherits name='com.google.gwt.user.User' />

  <!-- Inherit GWT i18n support
  <inherits name="com.google.gwt.i18n.I18N" />

  <!-- Specify the app entry point class.
  <entry-point class='cz.sweb.pichlik.gwt.samples.client.GWTSamples' />

  <!-- Add Czech locale -->
  <extend-property name="locale" values="cs" />

  <!-- Expose service in order to remote call -->
  <servlet path='/personService' class='cz.sweb.pichlik.gwt.samples.server.PersonServiceImpl' />

</module>
```

GWT features

- **history management (back button)**
- **serialization (IsSerializable marker, Type Arguments @gwt.typeArgs)**
- **I18N (constants, messages)**
- **UI composition and CSS styling**
- **command line tools (project creator, I18N generator)**
- **jUnit integration**
- **JavaScript Native Interface**

GWT approach

- **rapid development based on Java technology**
- **adopt concept of desktop application in web environment**
- **all in one solution (compiler, UI library, RCP)**
- **clean separation between UI composition and UI styling**

Benefits

- **easy RIA development**
- **well documented**
- **growing community**
- **Google support**
- **Java platform tools integration (IntelliJ Idea)**
- **existence of 3rd party subprojects (UI libraries, JSF integration)**

Questions and Answers

?