

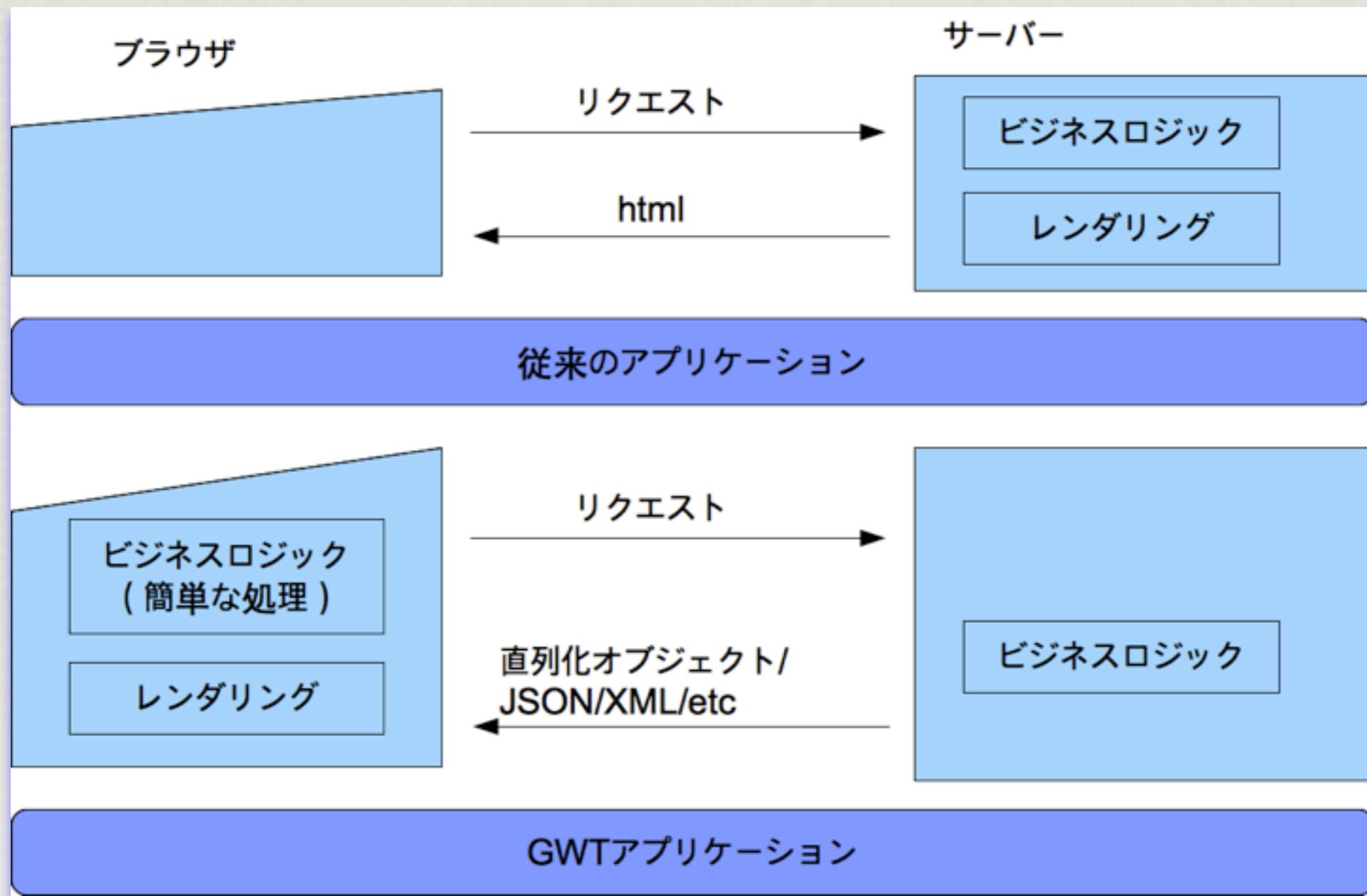
GWT



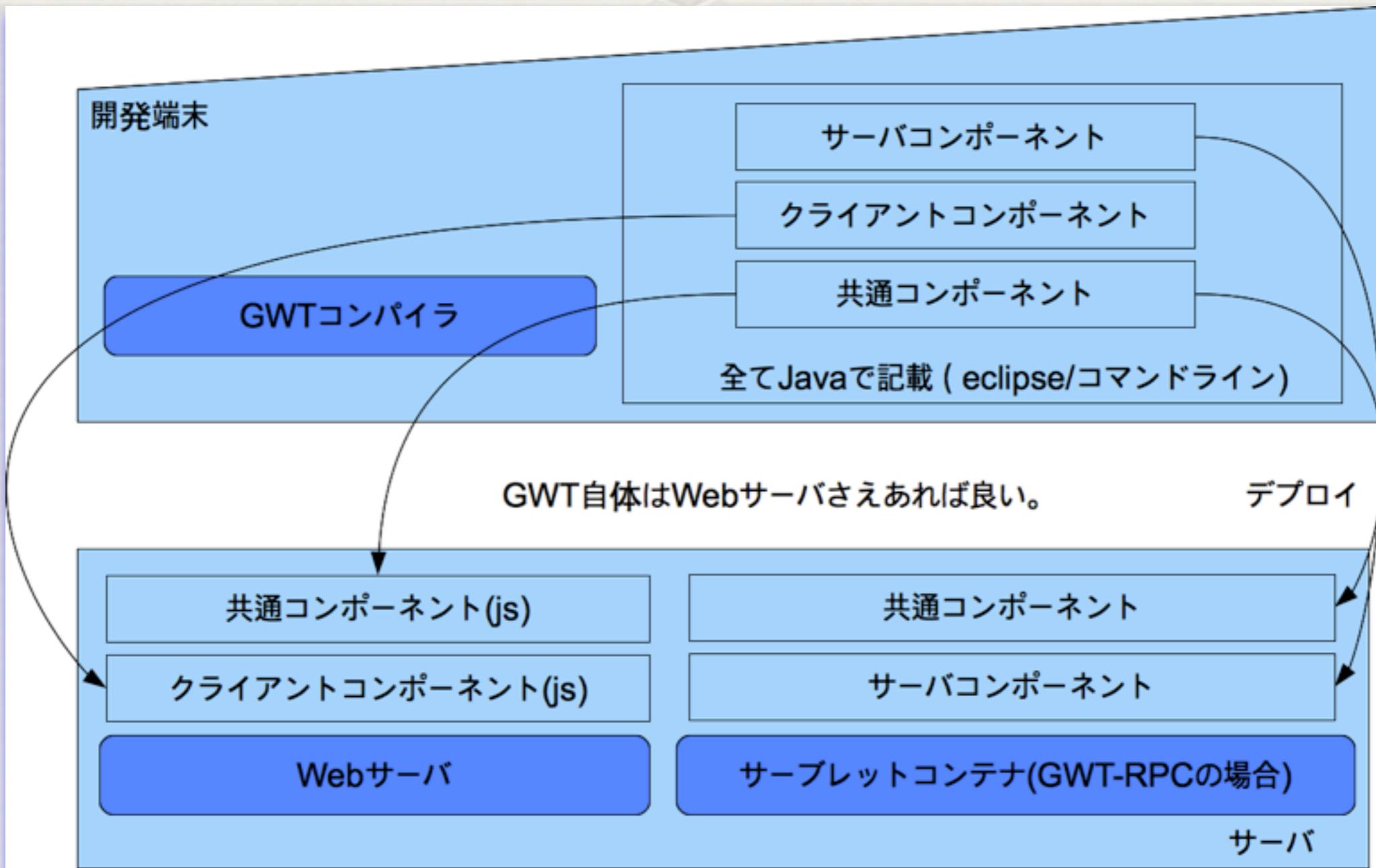
概要

- ・♪・Google Web Toolkit。
- ・♪・その名の通りGoogleが開発したツールキット。
- ・♪・AjaxアプリケーションをJavaで書くことが可能。
- ・♪・Apache License 2.0で配布。
- ・♪・AppEngine(Googleのクラウド)でデフォルトでサポート。

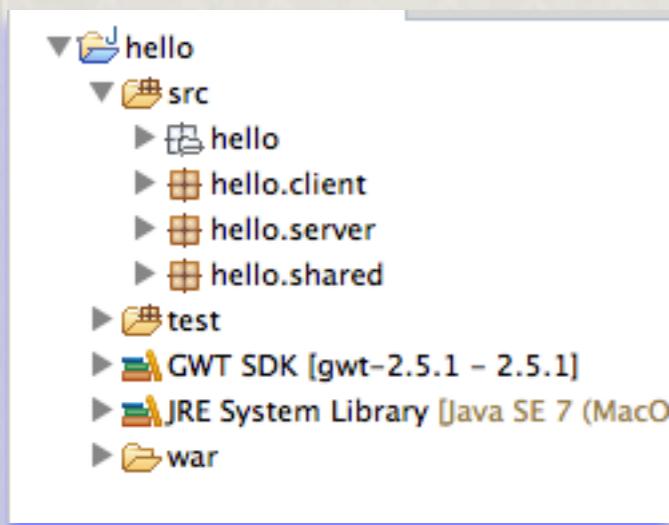
アーキテクチャ



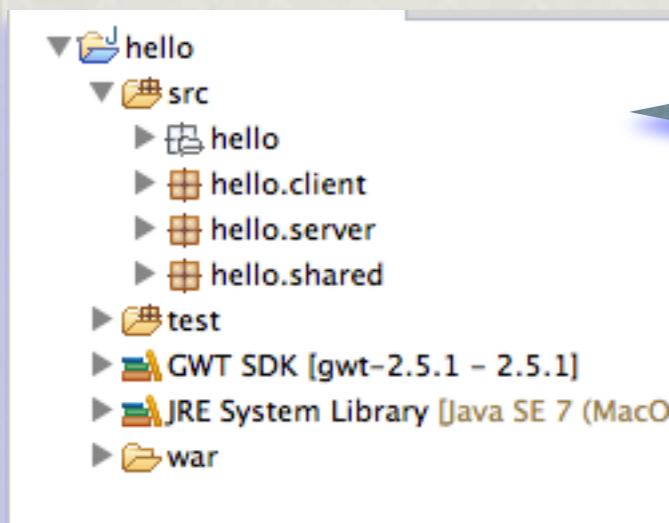
アーキテクチャ



構成

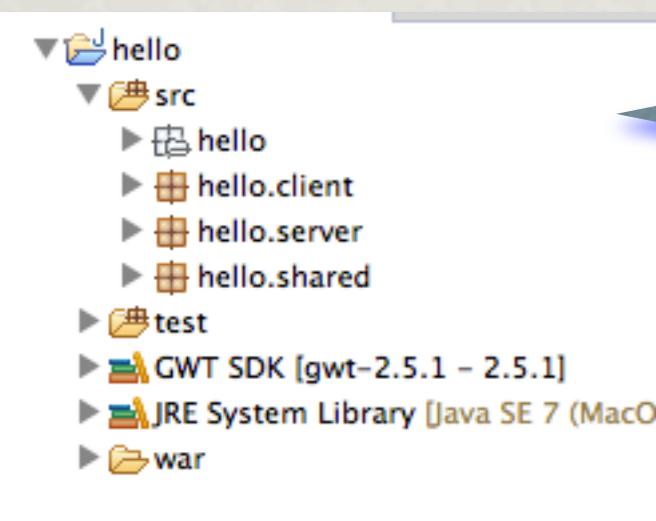


構成



clientはクライアント側のコード(JavaScriptに変換される)。serverはサーバサイドで動作するコード。sharedは両方で使用されるコード。

構成

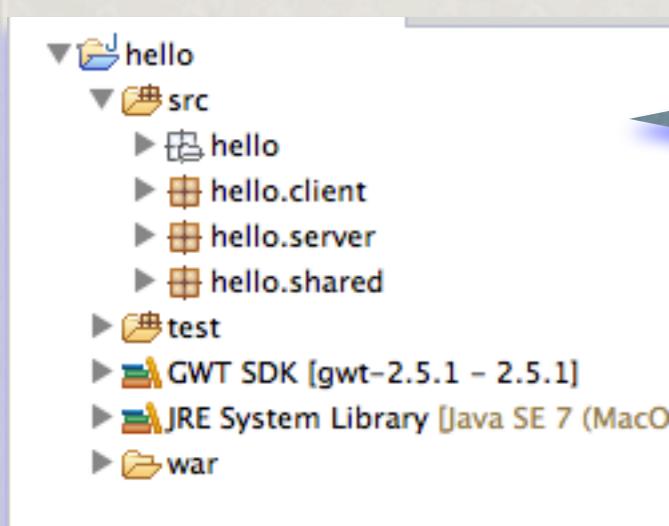


clientはクライアント側のコード(JavaScriptに変換される)。serverはサーバサイドで動作するコード。sharedは両方で使用されるコード。

sharedのコード

```
public class FieldVerifier {  
    public static boolean isValidName(String name) {  
        if (name == null) {  
            return false;  
        }  
        return name.length() > 3;  
    }  
}
```

構成



clientはクライアント側のコード(JavaScriptに変換される)。serverはサーバサイドで動作するコード。sharedは両方で使用されるコード。

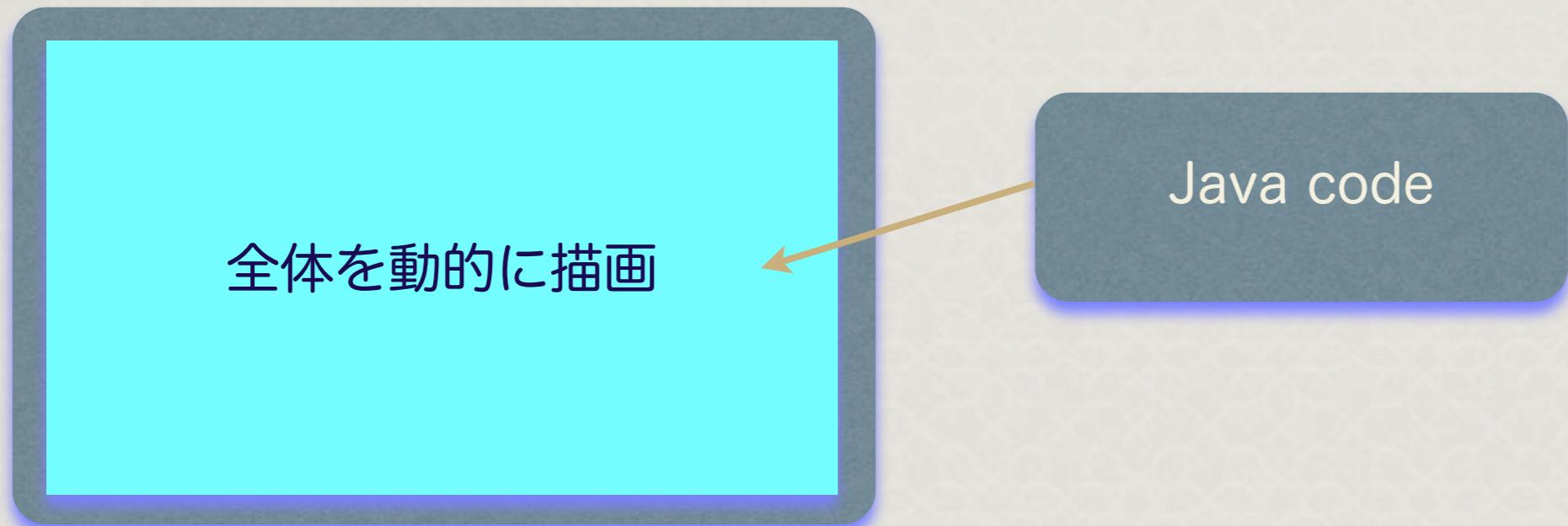
sharedのコード

```
public class FieldVerifier {  
    public static boolean isValidName(String name) {  
        if (name == null) {  
            return false;  
        }  
        return name.length() > 3;  
    }  
}
```

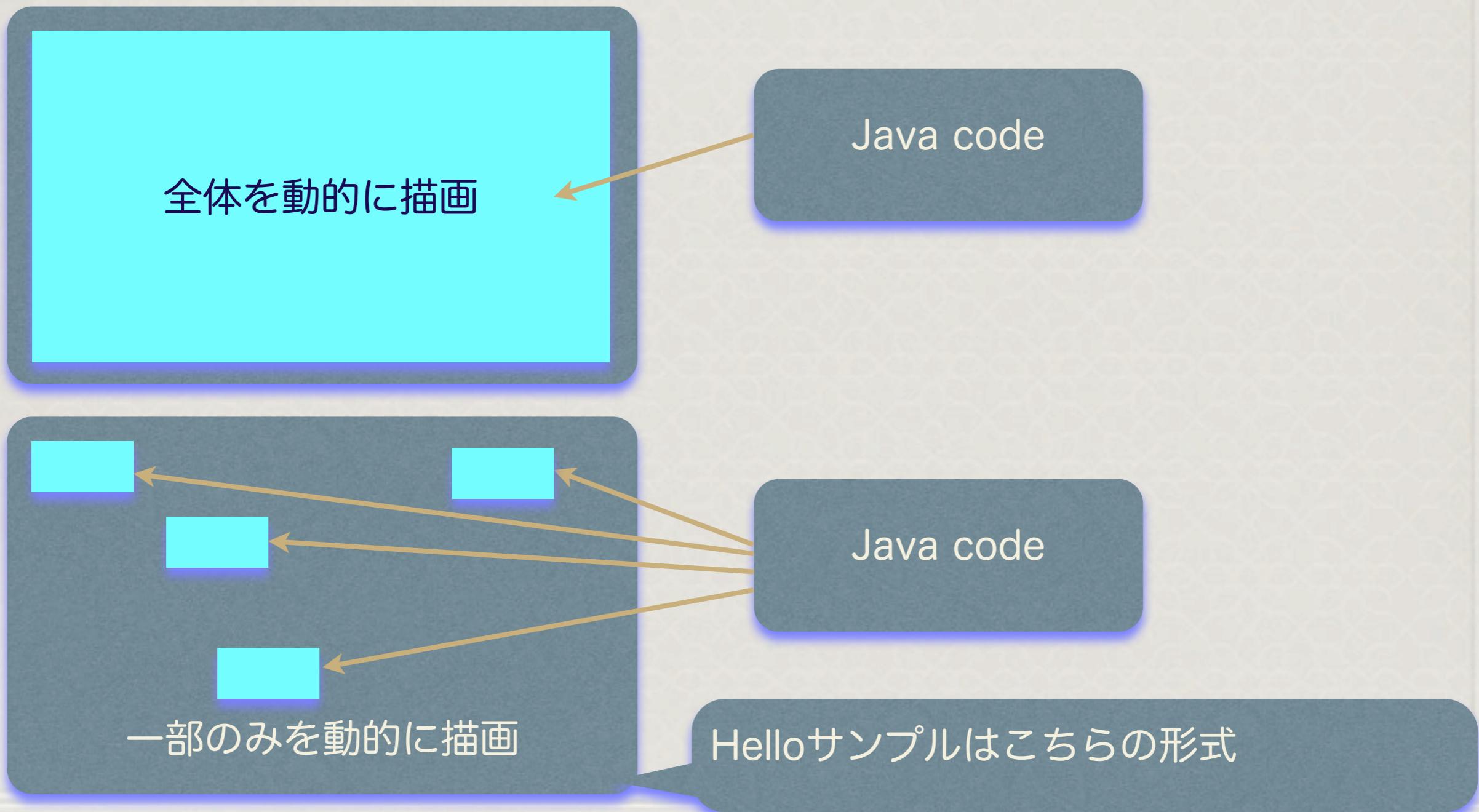
バリデーションはクライアント、
サーバで共用できる。

htmlとの関係

htmlとの関係



htmlとの関係



htmlとの関係

```
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <link type="text/css" rel="stylesheet" href="Hello.css">
    <title>Web Application Starter Project</title>
    <script type="text/javascript" language="javascript" src="hello/hello.nocache.js"></script>
  </head>
  <body>
    <h1>Web Application Starter Project</h1>

    <table align="center">
      <tr>
        <td colspan="2" style="font-weight:bold;">Please enter your name:</td>
      </tr>
      <tr>
        <td id="nameFieldContainer"></td>
        <td id="sendButtonContainer"></td>
      </tr>
      <tr>
        <td colspan="2" style="color:red;" id="errorLabelContainer"></td>
      </tr>
    </table>
  </body>
</html>
```

htmlとの関係

```
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <link type="text/css" rel="stylesheet" href="Hello.css">
    <title>Web Application Starter Project</title>
    <script type="text/javascript" language="javascript" src="hello/hello.nocache.js"></script>
</head>
<body>
    <h1>Web Application Starter Project</h1>

    <table align="center">
        <tr>
            <td colspan="2" style="font-weight:bold;">Please enter your name:</td>
        </tr>
        <tr>
            <td id="nameFieldContainer"></td>
            <td id="sendButtonContainer"></td>
        </tr>
        <tr>
            <td colspan="2" style="color:red;" id="errorLabelContainer"></td>
        </tr>
    </table>
</body>
</html>
```

JavaからJavaScriptにコンパイルされたコード。

htmlとの関係

```
<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <link type="text/css" rel="stylesheet" href="Hello.css">
    <title>Web Application Starter Project</title>
    <script type="text/javascript" language="javascript" src="hello/hello.nocache.js"></script>
</head>
<body>
    <h1>Web Application Starter Project</h1>

    <table align="center">
        <tr>
            <td colspan="2" style="font-weight:bold;">Please enter your name:</td>
        </tr>
        <tr>
            <td id="nameFieldContainer"></td>
            <td id="sendButtonContainer"></td>
        </tr>
        <tr>
            <td colspan="2" style="color:red;" id="errorLabelContainer"></td>
        </tr>
    </table>
</body>
</html>
```

JavaからJavaScriptにコンパイルされたコード。

通常のHTML内で要素にid属性を付けておく。

htmlとの関係

```
public class Hello implements EntryPoint {  
    public void onModuleLoad() {  
        final Button sendButton = new Button("Send");  
        final TextBox nameField = new TextBox();  
        nameField.setText("GWT User");  
        final Label errorLabel = new Label();  
  
        sendButton.addStyleName("sendButton");  
  
        RootPanel.get("nameFieldContainer").add(nameField);  
        RootPanel.get("sendButtonContainer").add(sendButton);  
        RootPanel.get("errorLabelContainer").add(errorLabel);  
  
        nameField.setFocus(true);  
        nameField.selectAll();  
    }  
}
```

htmlとの関係

```
public class Hello implements EntryPoint {  
    public void onModuleLoad() {  
        final Button sendButton = new Button("Send");  
        final TextBox nameField = new TextBox();  
        nameField.setText("GWT User");  
        final Label errorLabel = new Label();  
  
        sendButton.addStyleName("sendButton");  
  
        RootPanel.get("nameFieldContainer").add(nameField);  
        RootPanel.get("sendButtonContainer").add(sendButton);  
        RootPanel.get("errorLabelContainer").add(errorLabel);  
  
        nameField.setFocus(true);  
        nameField.selectAll();
```

idを指定してGUIのパートをアタッチする。

イベントハンドラ

```
class MyHandler implements ClickHandler, KeyUpHandler {  
    public void onClick(ClickEvent event) {  
        sendNameToServer();  
    }  
  
    public void onKeyUp(KeyUpEvent event) {  
        if (event.getNativeKeyCode() == KeyCodes.KEY_ENTER) {  
            sendNameToServer();  
        }  
    }  
  
    private void sendNameToServer() {  
        ...  
    }  
}  
  
MyHandler handler = new MyHandler();  
sendButton.addClickHandler(handler);  
nameField.addKeyUpHandler(handler);
```

ボタンクリック用

キー入力用

ハンドラを登録する

サーバ送信

- ◆ GWTにはサーブレットと通信するためのRPCモジュールが入っているので、これを利用すると簡単にサーバ間のAjax通信ができる。
- ◆ RPCは何を使っても良いので、サーバはJavaでなくても良い。

サーバ送信

```
private void sendNameToServer() {  
    errorLabel.setText("");  
    String textToServer = nameField.getText();  
    if (!FieldVerifier.isValidName(textToServer)) {  
        errorLabel.setText("Please enter at least four characters");  
        return;  
    }  
  
    sendButton.setEnabled(false);  
    textToServerLabel.setText(textToServer);  
    serverResponseLabel.setText("");
```

クライアントサイドバリデーション

サーバ通信中、ボタンを無効化

shared/FieldVerifier.java

```
public class FieldVerifier {  
    public static boolean isValidName(String name) {  
        if (name == null) {  
            return false;  
        }  
        return name.length() > 3;  
    }  
}
```

GWT RPC

client/GreetingService.java

```
@RemoteServiceRelativePath("greet")
public interface GreetingService extends RemoteService {
    String greetServer(String name) throws IllegalArgumentException;
}
```

サービスインターフェイス

サーバ処理完了時に呼び出される

client/GreetingServiceAsync.java

```
public interface GreetingServiceAsync {
    void greetServer(String input, AsyncCallback<String> callback)
        throws IllegalArgumentException;
}
```

対応するASYNCインターフェイスが必要。
Eclipseを使用している場合は自動生成してくれる。

GWT RPC

server/GreetingServiceImpl.java

```
public class GreetingServiceImpl  
    extends RemoteServiceServlet implements GreetingService  
{  
    public String greetServer(String input) throws IllegalArgumentException {  
        if (!FieldVerifier.isValidName(input)) {  
            throw new IllegalArgumentException("Name must be at least 4 characters long");  
        }  
        String serverInfo = getServletContext().getServerInfo();  
        String userAgent = getThreadLocalRequest().getHeader("User-Agent");  
  
        input = escapeHtml(input);  
        userAgent = escapeHtml(userAgent);  
  
        return "Hello, " + input + "<br><br>I am running " + serverInfo  
            + ".<br><br>It looks like you are using:<br>" + userAgent;  
    }  
}
```

サーバーサイドの実装

サーバーサイドバリデーション(同じクラス)

GWT RPC

client/Hello.java

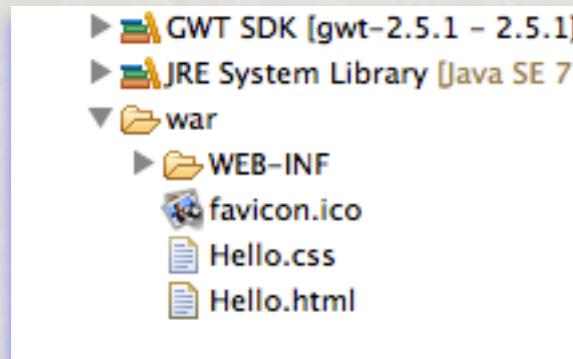
クライアントからの呼び出し

```
private final GreetingServiceAsync greetingService = GWT.create(GreetingService.class);

greetingService.greetServer
(textToServer,
new AsyncCallback<String>() {
    public void onFailure(Throwable caught) {
        // 失敗した時の処理
    }

    public void onSuccess(String result) {
        // 成功した時の処理
    }
});
```

構成



静的コンテンツが格納される。WEB-INFはサーバ側にサーブレットを使用する場合のみ必要。GWTはサーバーサイドを規定しない。Javaでなくとも良い。

モジュール

◆ GWTにおいて、機能をまとめる単位。

モジュール定義ファイル: src/hello/Hello.gwt.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<module rename-to='hello'>
  <inherits name='com.google.gwt.user.User'/>
  <inherits name='com.google.gwt.user.theme.clean.Clean'/>

  <entry-point class='hello.client.Hello'/>

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

クラスパスのルートに配置する
のが推奨(mustではない)

他のモジュールを取り
込む

開始画面(オプション)

変換対象のソースが格納されている場所

rename-to属性で短縮名を付けることができる。これが無いと、このケースでは‘hello.Hello’がモジュール名になる。

モジュールの利用

Hello.html

```
<script type="text/javascript" language="javascript"  
src="hello/hello.nocache.js"></script>
```

モジュールを指定する(短縮名を使用した場合)

```
<script type="text/javascript" language="javascript"  
src="hello.Hello/hello.Hello.nocache.js"></script>
```

短縮名を使用しないと、このようになる。

全画面

アプリケーション

全画面アプリケーション

- ◆ Ajaxの普及とともに、1つの画面から遷移せずに動作するアプリケーションが増えてきた。
- ◆ 例：GMail、Google Map。これらは1つの画面の中で、JavaScriptを用いて画面内部を書き変えることで動作する。
- ◆ helloアプリケーションを全画面アプリケーションに書き換えてみましょう。

全画面アプリケーション

client/Hello.java

```
// RootPanel.get("nameFieldContainer").add(nameField);
// RootPanel.get("sendButtonContainer").add(sendButton);
// RootPanel.get("errorLabelContainer").add(errorLabel);

RootPanel.get().add(nameField);
RootPanel.get().add(sendButton);
RootPanel.get().add(errorLabel);
```

RootPanelに直接add()する

war/Hello.html

```
<body>
<iframe src="javascript:''" id="__gwt_historyFrame" tabIndex='-1' style="position:absolute;width:0;height:0;border:0"></iframe>
</body>
```

ヒストリー制御以外を消去

内容を変更したらブラウザの再読み込み。GWTを開発モードで使用中は、ビルドは不要で、ブラウザを再読み込みすれば変更が反映される。

全画面アプリケーション

GWT User

Send

```
<input class="gwt-TextBox" type="text">
<button class="gwt-Button sendButton" type="button">Send</button>
<div class="gwt-Label"></div>
```

Firebugでdomを見るとinput/button/divタグが生成されていることが分かる。

パネル

- ◆ 複数のパートを配置するコンポーネント。
- ◆ これによりパートをまとめて1つのパートとして扱うことができる。
- ◆ リサイズに反応したい場合は、xxxPanelのかわりに、xxxLayoutPanelを使用する。

パネル

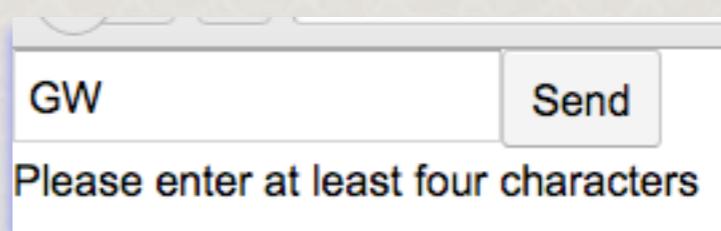
```
HorizontalPanel fp = new HorizontalPanel();
fp.add(nameField);
fp.add(sendButton);
```

子供を水平に並べる

```
VerticalPanel vp = new VerticalPanel();
vp.add(fp);
vp.add(errorLabel);
```

子供を垂直に並べる

```
RootLayoutPanel.get().add(vp);
```



CSSスタイルの付与

hello/client/Hello.java

```
HorizontalPanel fp = new HorizontalPanel();
fp.setStyleName("inputArea");
fp.add(nameField);
fp.add(sendButton);
nameField.setStyleName("inputField");

VerticalPanel vp = new VerticalPanel();
vp.setStyleName("inputBlock");
vp.add(fp);
vp.add(errorLabel);
```

hello/client/Hello.java

```
.inputArea {
    width: 100%;
}

.inputBlock {
    margin: 20px;
    width: 80%;
}

.inputField {
    width: 95%;
}
```

GWT User

Send

開発モードと製品モード

・❖ 開発モード

これまで見てきたのが開発モード。開発モードではブラウザの更新で変更を取り込みことができ、デバッガも使用できる。実際にはJavaのコードは、そのままJavaとして実行されている。

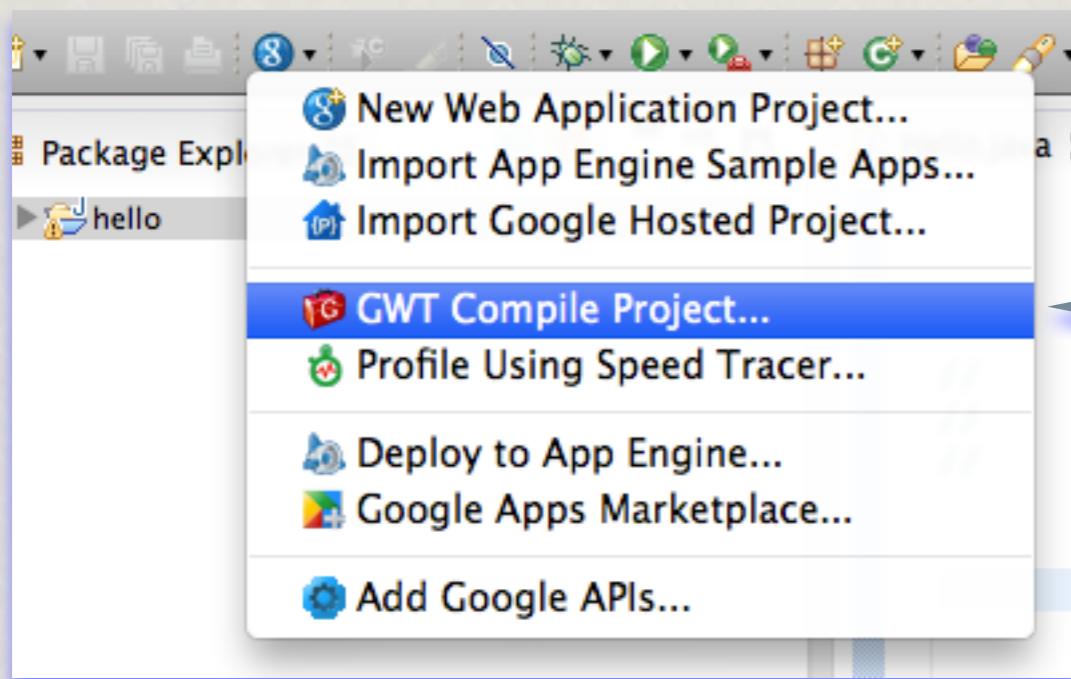
url: `http://127.0.0.1:8888/Hello.html?`

`gwt.codesvr=127.0.0.1:9997`

・❖ 製品モード

JavaをJavaScriptにコンパイルして実行する。

製品モードを試す



GWT Compile Project.. を選び、対象として今回のプロジェクトを選ぶ。

製品モードを試す

- ・ 今回はServletと通信するため、これまで同様、Run->Web Applicationでサーバを起動しておく。
- ・ URLには、<http://127.0.0.1:8888>Hello.html> を指定(？以降を削除)。

製品モードを試す

生成されるファイル

```
war\hello\18EEC2DA45CB5F0C2050E2539AE61FCE.cache.html
war\hello\813B962DC4C22396EA14405DDEF020EE.cache.html
war\hello\86DA1DCEF4F40731BE71E7978CD4776A.cache.html
war\hello\A37FC20FF4D8F11605B2C4C53AF20B6F.cache.html
war\hello\E3C1ABB32E39A126A9194DB727F7742A.cache.html
war\hello\14A43CD7E24B0A0136C2B8B20D6DF3C0.cache.png
war\hello\548CDF11D6FE9011F3447CA200D7FB7F.cache.png
war\hello\9DA92932034707C17CFF15F95086D53F.cache.png
war\hello\A7CD51F9E5A7DED5F85AD1D82BA67A8A.cache.png
war\hello\B8517E9C2E38AA39AB7C0051564224D3.cache.png
war\hello\clear.cache.gif
war\hello\hello.nocache.js
war\hello\hosted.html
war\Hello.html
```

製品モードを試す

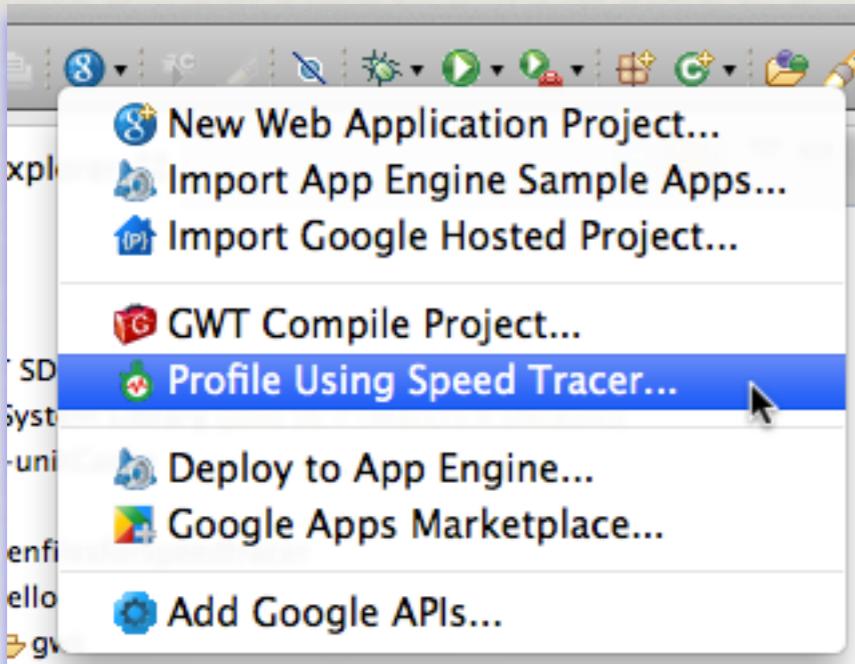
- ・ ファイル名にcacheと入っているファイルは、Webサーバで永遠にキャッシュして良いと応答を返すように設定する。
- ・ ファイル名にnocacheが入っているファイルは、Webサーバでキャッシュを禁止する応答を返すように設定する。
- ・ 複数のhtmlが存在するのは、特定のブラウザ用に最適化されたファイル。GWT側でブラウザを判定して、互換性を確保したhtmlを生成してくれる。

製品モードを試す

- ・ ファイル名にcacheと入っているファイルで内容が変化するものは、ファイル名にMD5ハッシュが付加される。このためクライアント側のキャッシュ寿命を無限大にして構わない。
これによりネットワークトラフィックを最低限に抑えることができる。

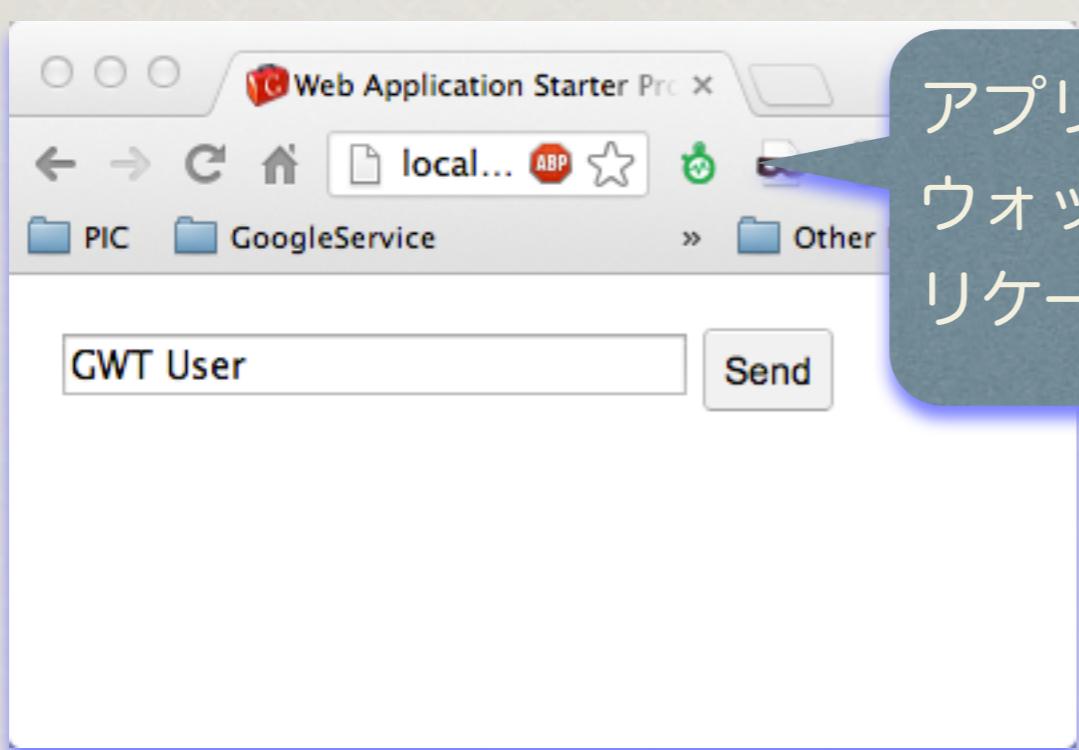
プロファイル

- ・ JavaScriptプロファイルが標準で用意されている。
- ・ Google Chromeが必要。



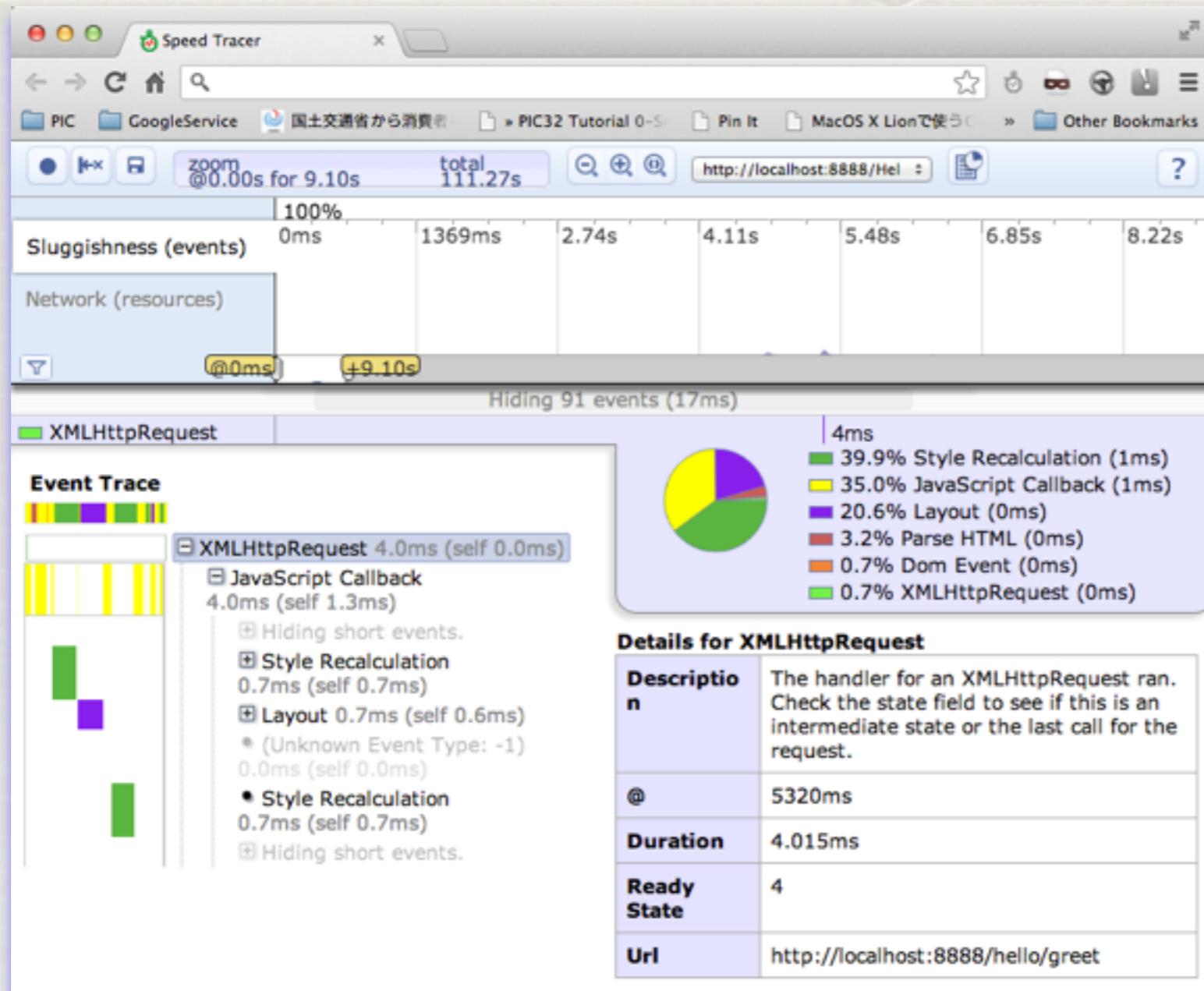
メニューから選んで、対象アプリケーションを指定し、Chromeの場所を指定する。
最初はプラグインのインストールを要求される。

プロファイル



アプリケーションを表示してから、ストップウォッチのアイコンをクリックした後、アプリケーションを操作する。

プロファイル例



4ms

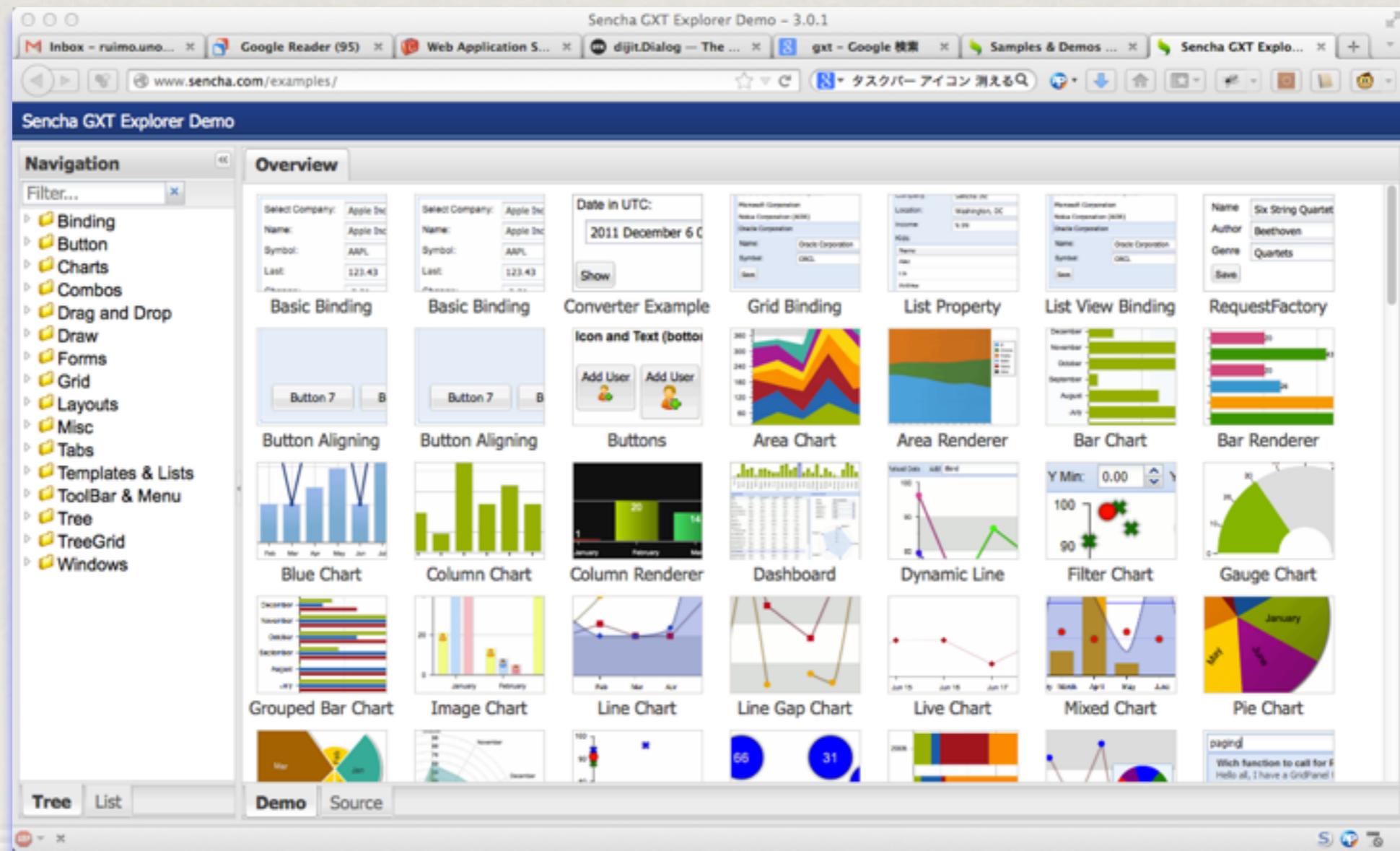
- 39.9% Style Recalculation (1ms)
- 35.0% JavaScript Callback (1ms)
- 20.6% Layout (0ms)
- 3.2% Parse HTML (0ms)
- 0.7% Dom Event (0ms)
- 0.7% XMLHttpRequest (0ms)

Details for XMLHttpRequest

Description	The handler for an XMLHttpRequest ran. Check the state field to see if this is an intermediate state or the last call for the request.
@	5320ms
Duration	4.015ms
Ready State	4
Url	http://localhost:8888/hello/greet

サードパーティ・ウィジェット

Sencha GXT: <http://www.sencha.com/products/gxt/examples/>



Vaadin

- ・ GWTを利用したサーバサイドも含めたWebフレームワーク
- ・ サーバサイドで動作するため、Javaのフル機能を利用できる(GWTの場合、JavaScriptにコンパイルする関係で使用できるAPIに制限がある)

まとめ

- ・ GWT = Google Web Toolkit
- ・ JavaでAjaxアプリケーションを構築できる
- ・ 開発モードはソースを直してブラウザ更新で反映
- ・ コンパイル時にブラウザの差異を吸収
- ・ コンテンツのキャッシュを効率的に実現
- ・ サーバサイドは何でも良い(Servletを用いたRPCが標準で提供される)
- ・ プロファイラが標準添付

まとめ

- ・既存のhtml内に簡単に組み込むことが可能
- ・動的コンポーネントをJavaで書けるため、JSPのスクリプト要素のような複雑な記述が不要
- ・画面全体をJavaで記述することも可能
- ・全てJavaで書けるため、バリデーション処理などはクライアントとサーバで共用できる

まとめ

- ・BigDecimalが使えるので勘定系計算処理で悩まなくて良い(JavaScriptは浮動小数点数しかないので、桁落ちのやっかいなバグが発生しがち)
- ・JUnitなどJava用の開発ツールが活用できる
- ・GWT RPCを使えば、サーバ通信の引数、戻り値は自動的にマーシャリングされる
- ・エラー時は例外をサーバから送出して、クライアントで、そのままJavaの例外として受け取れば良い