14.–17. 09. 2009
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Werkzeugkasten

Das Google Web Toolkit (GWT)

Rüdiger Schobbert

emesit GmbH & Co. KG

Agenda

- Einführung
- Los geht's
- Bestandteile eines Projekts
- GWT in der näheren Betrachtung
- Ausblick auf GWT 2.0

Einführung...

Was verspricht GWT?

Google sagt:

„GWT's mission is to radically improve the web experience for users by enabling developers to use existing Java tools to build no-compromise AJAX for any modern browser.“

„With Google Web Toolkit (GWT), you write your AJAX front-end in the Java programming language which GWT then cross-compiles into *optimized* JavaScript that automatically works across all major browsers.“

Stärken von GWT?

- Code wird in Java geschrieben
- Jede Lieblings-IDE kann verwendet werden
- Als Bytecode in einem speziellen Webbrowser debuggen (Hosted Mode)
- Cross-compile in optimiertes JavaScript (Web Mode)
- Benötigt kein Browser Plugin
- Beinhaltet umfangreiche Cross-Browser Bibliotheken
  - User Interface (DOM, Widgets)
  - Client/Server Kommunikation (XHR, RPC, JSON)
  - Applikations-Infrastruktur (History, Timers, Unittesting, i18n)
  - Externe Services (Gadgets, Gears, Google Maps)

Was ist GWT?

- (Noch) ein Toolkit für die Entwicklung von Webanwendungen
- Besteht aus
  - Java Klassenbibliothek
  - Verschiedene Werkzeuge
- Java-to-JavaScript Compiler
  - Wandelt den clientseitigen Java Code in JavaScript um
  - 'Echter' Compiler
    - Dead code elimination (classes/methods/-parameters/fields)
    - Inlinen von Methoden
  - Beachtet Browser Inkompatibilitäten
  - Erzeugt hochoptimierten JavaScript Code

Warum GWT?

- Aus Entwicklersicht:
  - dieselbe Programmiersprache (Java) mit allen Vorteilen
    - gewohnte Werkzeuge (Eclipse, Netbeans, JProfiler, JUnit, Debugger)
    - Statische Typprüfung
    - Refactoring
  - minimale Einarbeitung
  - AJAX Programmierung wird kinderleicht
- Aus Entscheidersicht:
  - Einsatz bestehender Entwickler und Tools
  - Opensource mit freier Lizenz (Apache 2.0 License)
  - Google steht dahinter und steht zu dem Projekt

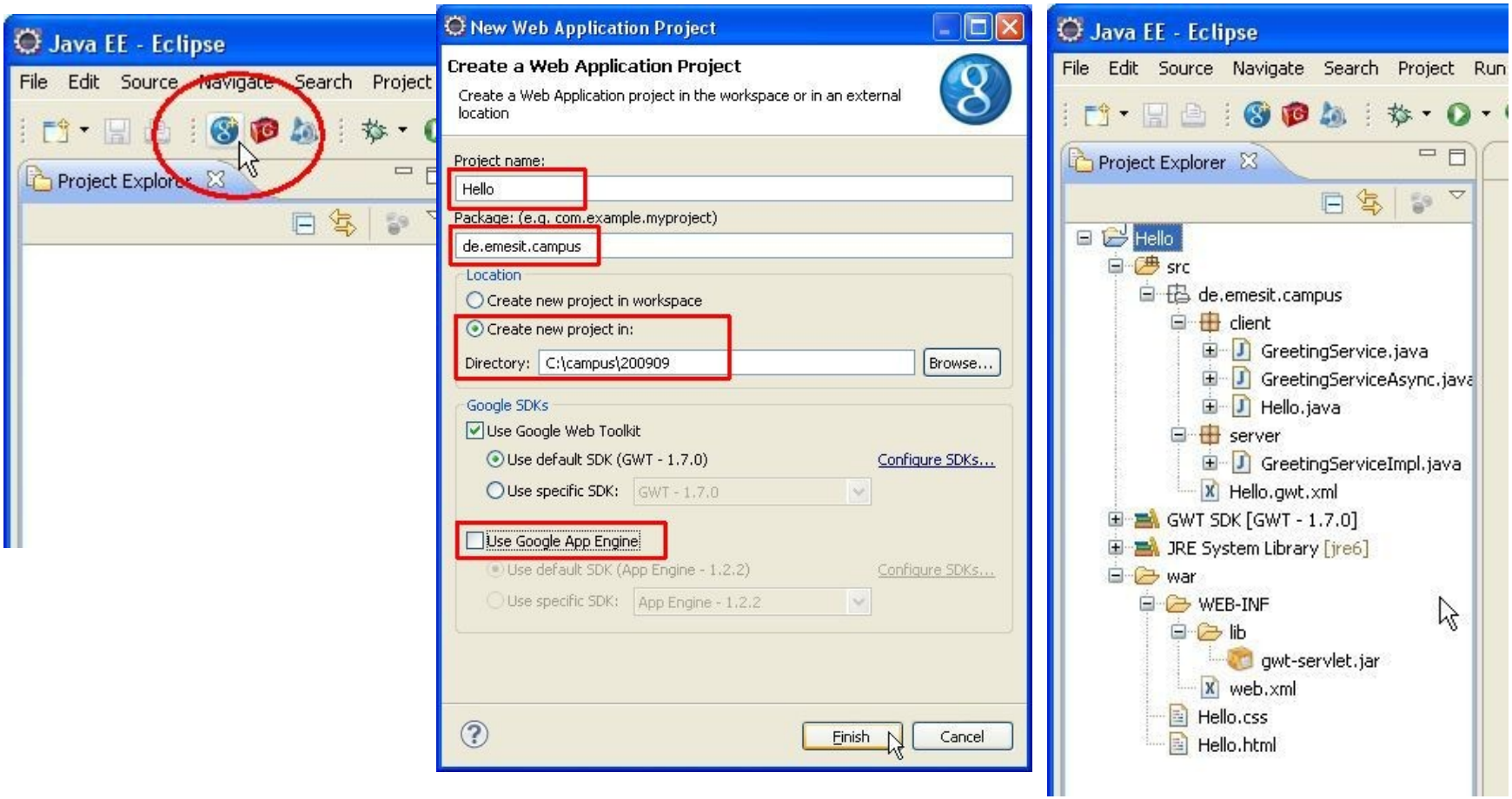
Was wird benötigt?

- Java SDK (ab JDK 5.0)
- Sinnvollerweise eine IDE (am besten Eclipse samt Google GWT Plugin)
- GWT (ist im Plugin enthalten)
- Servlet Container (z.B. Tomcat)
- Gängiger Browser mit aktiviertem JavaScript
  - Firefox
  - IE6, IE7, IE8
  - Safari 2, 3, 4
  - Opera 9.0
  - Chrome

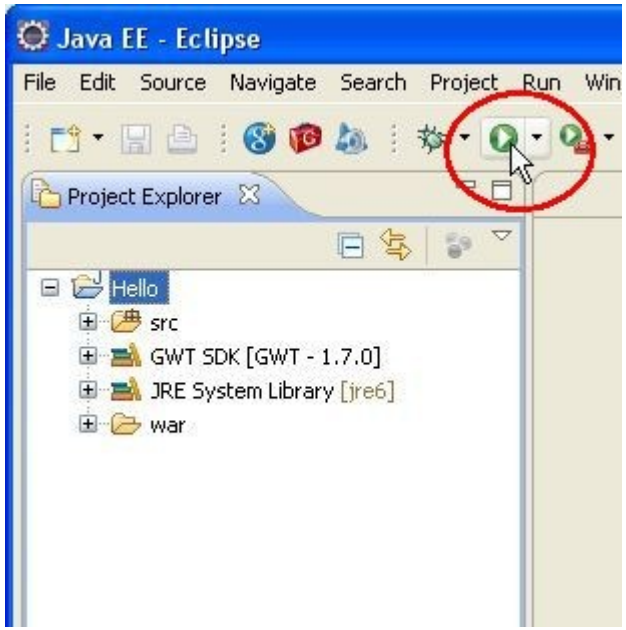
Und los geht's...

(mit eclipse 3.5 und dem google eclipse plugin)

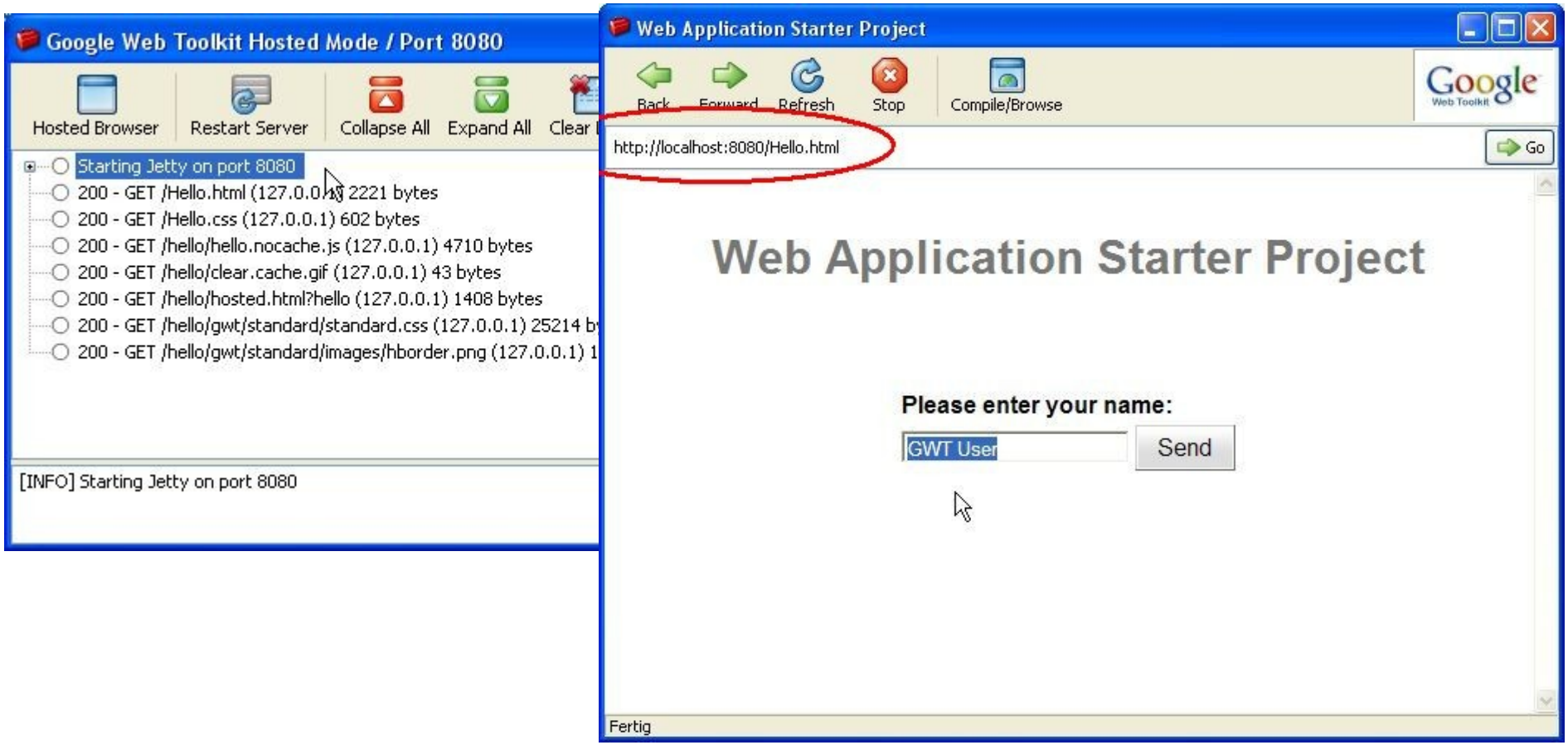
Projekt generieren



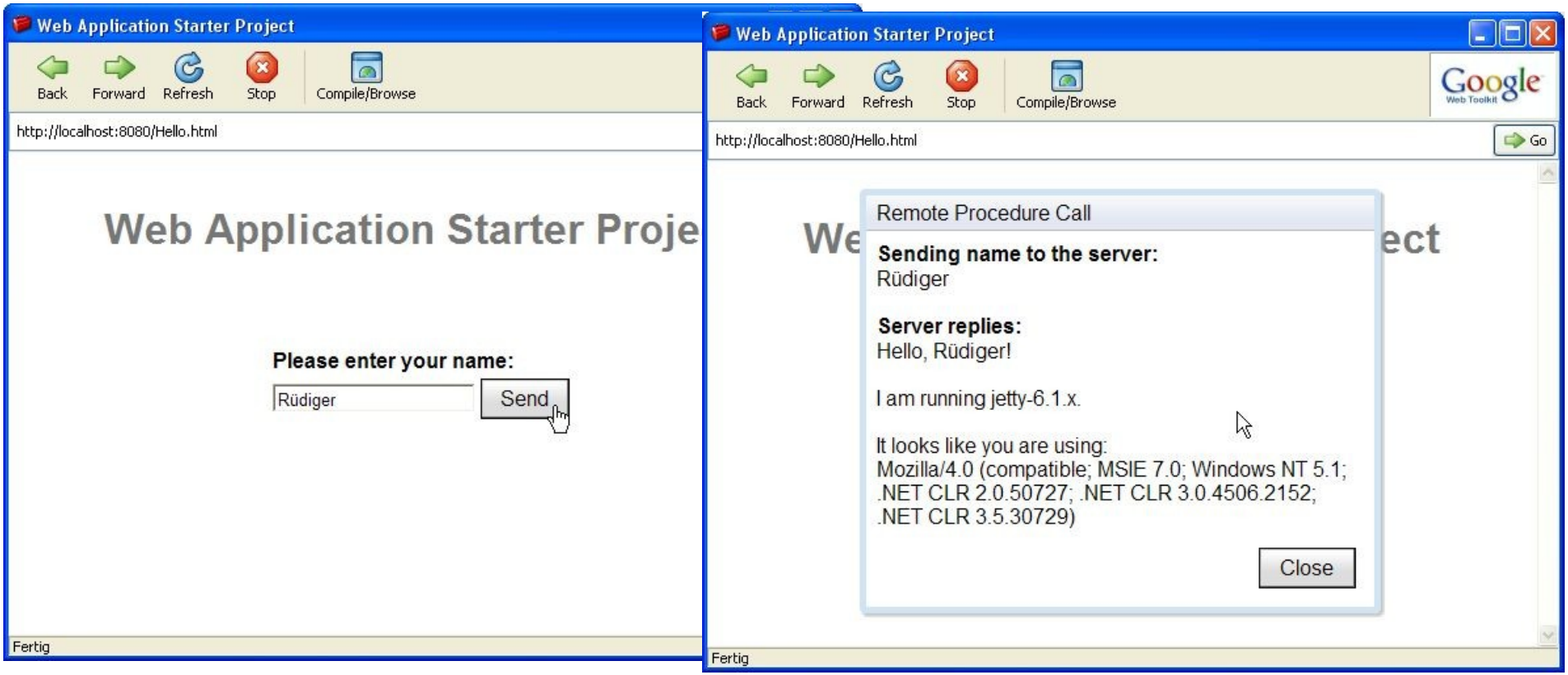
Projekt starten



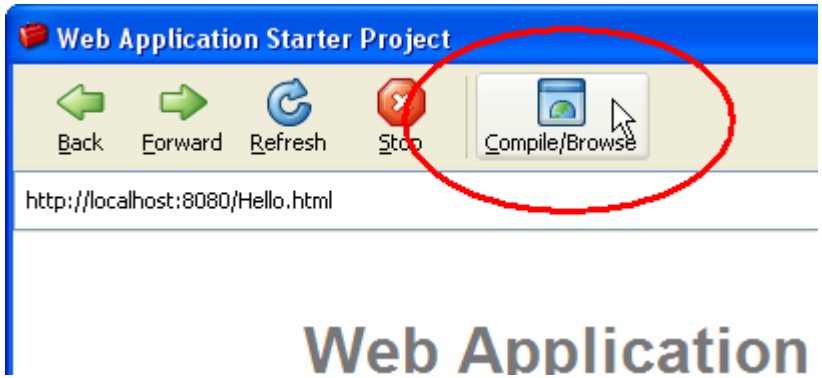
Projekt läuft

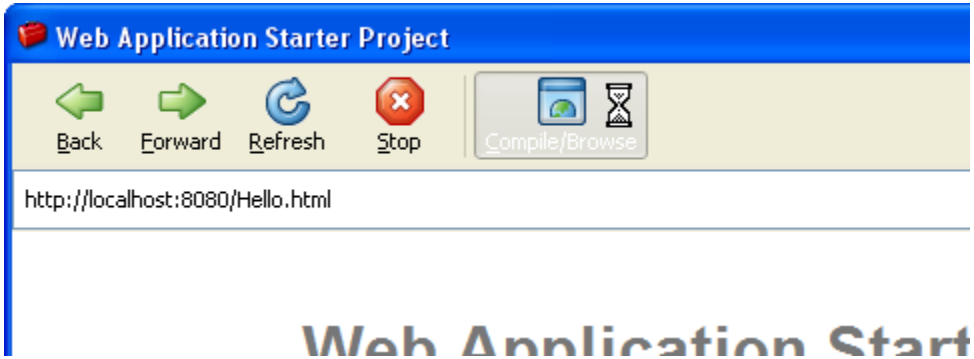


Projekt läuft

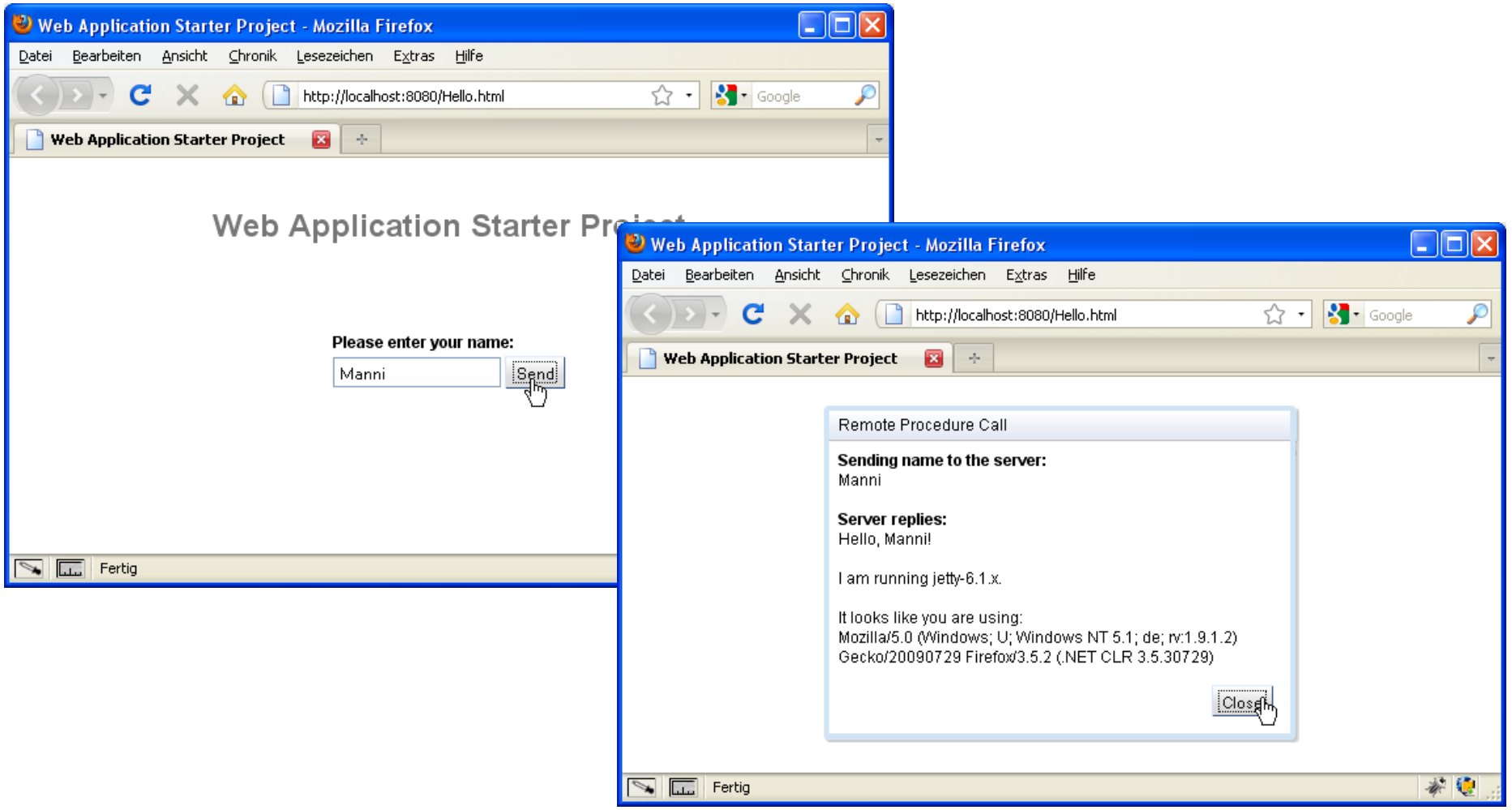


Button "Compile/Browse" drücken...



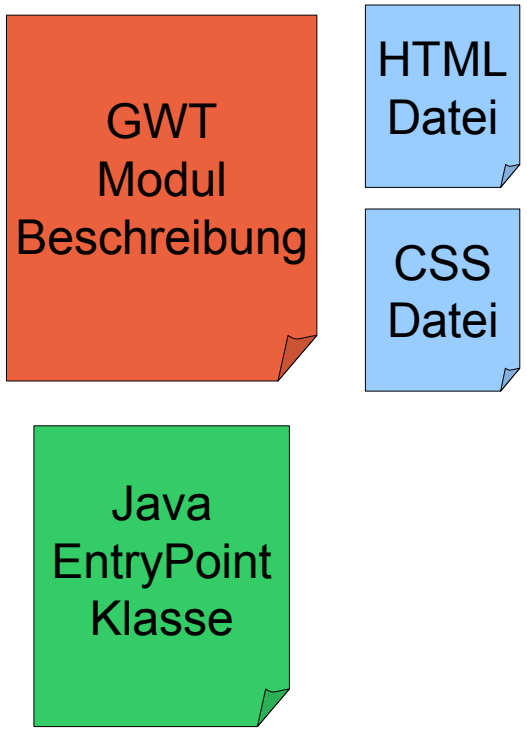


...und die Applikation im Standardbrowser betrachten



Die Bestandteile des generierten Projekts...

„Hello“ Projektbestandteile



HTML Hostpage

(Hello.html)

```
<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"/>
</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>
  </table>
</body>
</html>
```

Java EntryPoint Klasse

(Hello.java)

```
public class Hello implements EntryPoint {
    public void onModuleLoad() {
        final Button sendButton = new Button("Send");
        final TextBox nameField = new TextBox();
        nameField.setText("GWT User");

        RootPanel.get("nameFieldContainer").add(nameField);
        RootPanel.get("sendButtonContainer").add(sendButton);
        ...
    }
}
```

GWT Modul Beschreibung

(Hello.gwt.xml)

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

  <!-- Inherit the default GWT style sheet.  You can change       -->
  <!-- the theme of your GWT application by uncommenting          -->
  <!-- any one of the following lines.                            -->
  <inherits name='com.google.gwt.user.theme.standard.Standard'/>
  <!-- <inherits name='com.google.gwt.user.theme.chrome.Chrome'/> -->
  <!-- <inherits name='com.google.gwt.user.theme.dark.Dark'/>     -->

  <!-- Specify the app entry point class.                         -->
  <entry-point class='de.emesit.campus.client.Hello'/>
</module>
```

GWT in der näheren Betrachtung:

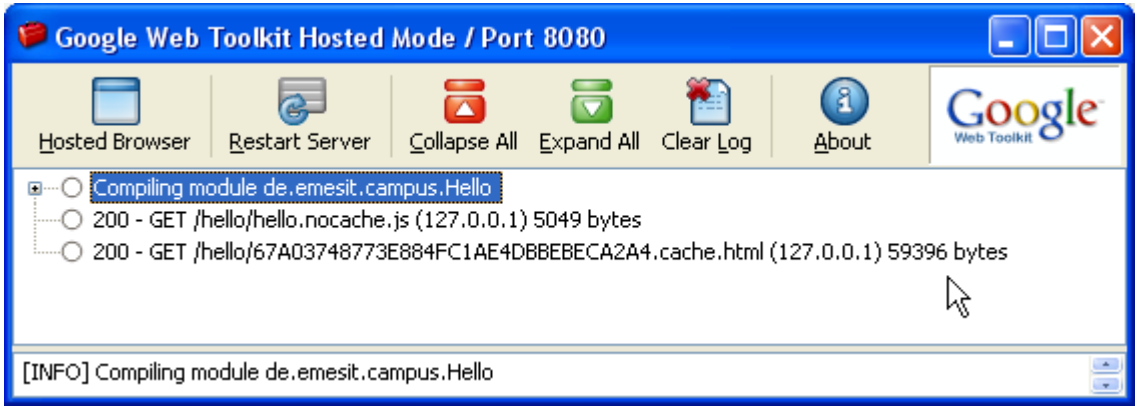
Betriebsmodi - Hosted und Web Mode...

Hosted Mode (Development Mode)

- eine Art Browser Emulator
- Klassen werden nicht nach JavaScript umgewandelt sondern direkt als Bytecode ausgeführt
- kein deployment nötig
- Änderungen am Code können sehr schnell betrachtet werden, kein Neustart nötig einfach 'Refresh' drücken
- Debuggen von client- und serverseitigem Code direkt in der IDE möglich
- Ab GWT 1.6 Refresh auch bei Serverseitige Codeänderungen möglich

Web Mode (Deployment Mode)

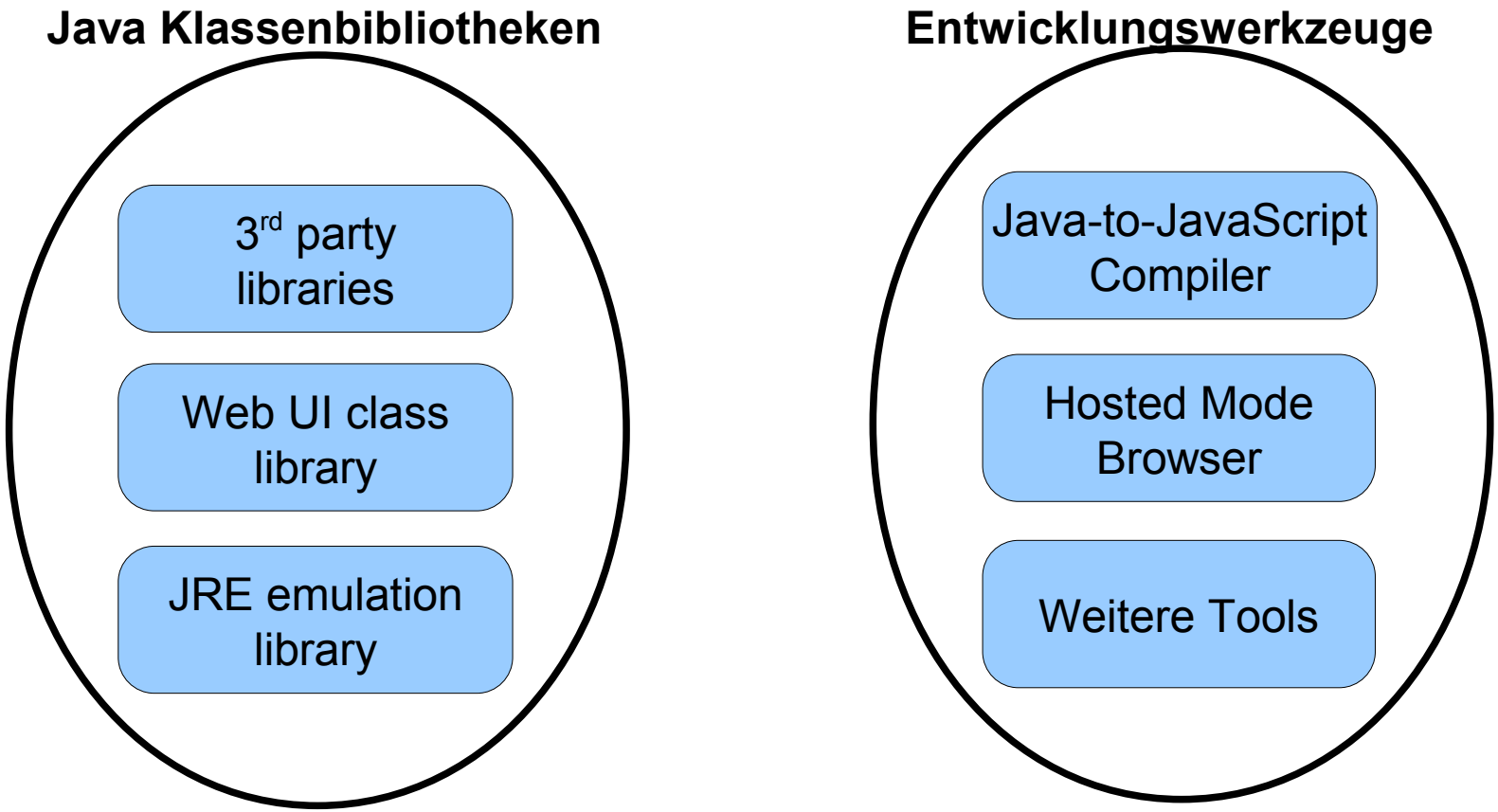
- Anwendung wird nativ als JavaScript und HTML ausgeführt
- Dazu Kompilieren nach JavaScript nötig
- Webmode kann aus dem Hosted Browser aufgerufen werden



GWT in der näheren Betrachtung:

Sprachelemente und Bibliotheken...

Die Bestandteile von GWT



Java Sprachunterstützung (1)

- Alle primitive Datentypen
  - Vorsicht: JavaScript hat nur 64 Bit Fließkommazahlen, daher Unterschied bei Ganzzahlüberlauf
- Exceptions: try, catch, finally wird unterstützt
- Assertions: assert Schlüsselwort wird unterstützt
- Java 5 Sprachfeatures:
  - Generics
  - for-each Schleifen
  - Autoboxing
  - Static imports
  - Enums

Java Sprachunterstützung (2)

- Kein Multithreading: synchronized Schlüsselwort wird ignoriert, entsprechende Methoden (wait, notify, etc.) ergeben Compiler Fehler
- Kein Reflection:
  - GWT.getTypeName(obj) statt obj.getClass().getName()
  - GWT.create(MyClass) statt Class.forName("MyClass")
- Keine Java Serialisierung

JRE Emulation Library

- Ermöglicht das Übersetzen nach JavaScript
- Enthält Klassen, die die Java Funktionalität (clientseitig) emulieren
- Es wird nur ein Subset der Klassen in java.lang, java.util (sowie java.io, java.sql) unterstützt
- Achtung: teilweise fehlen Methoden
- Im Zweifel im Hosted Mode starten

JRE Emulation Library

Package java.lang:
Interfaces:
CharSequence, Cloneable, Comparable, Iterable, Runnable

Exceptions:
Arithmetic-, ArrayIndexOutOfBounds-, ArrayStore-, ClassCast-, Exception, IllegalArgument-, IllegalState-, IndexOutOfBounds-, NegativeArraySize-, NullPointer-, NumberFormat-, Runtime-, StringIndexOutOfBounds-, UnsupportedOperation-

JRE Emulation Library

Package java.lang:
Classes:
Boolean, Byte, Character, Double, Float, Integer,
Long, Short
Object, Number, String, StringBuffer, StringBuilder
Class, Enum, Throwable, Void
Math, StackTraceElement, System

JRE Emulation Library

Package java.util:
Interfaces:
Collection, List, Set, SortedSet, Map, Map.Entry, SortedMap
Enumeration, Iterator, ListIterator, Comparator
Queue, RandomAccess, EventListener

Exceptions:
ConcurrentModification-, EmptyStack-, MissingResource-, NoSuchElement-, TooManyListeners-

JRE Emulation Library

Package java.util:
Classes:
AbstractCollection, AbstractList, AbstractMap,
AbstractQueue, AbstractSequentialList, AbstractSet,
ArrayList, EnumMap, EnumSet, HashMap, HashSet,
IdentityHashMap, LinkedHashMap, LinkedHashSet,
LinkedList, TreeMap, TreeSet, Vector
Arrays, Collections, PriorityQueue, Stack
Date, EventObject

JRE Emulation Library

Package java.io:
FilterOutputStream
OutputStream
PrintStream
Serializable

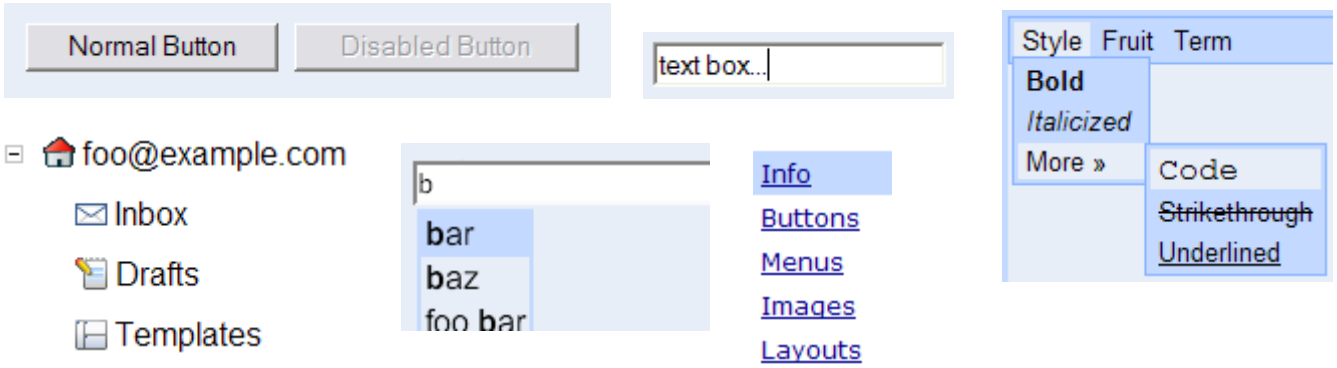
Package java.sql:
Date
Time
Timestamp

Web UI Class Library

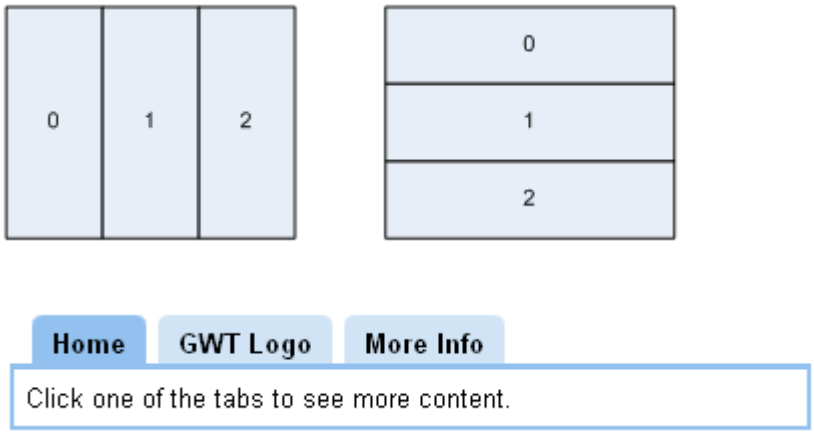
- Ähnlich zu anderen UI Bibliotheken wie Swing oder SWT
- Enthält fertige Widgets und Panels
  - Button, TextBox, MenuBar, Tree, SuggestBox, Hyperlink, ...
  - HorizontalPanel, VerticalPanel, TabPanel, …
- Events und Handlers
- DOM Support
- Image Bundles
- Internationalisierung
- Browser History Management
- Cookies
- Eigene Komponenten

Web UI Class Library

- Widgets:



- Panels:



Auf Ereignisse reagieren

- Soll auf Ereignisse reagiert werden muss das entsprechende Interface implementiert und beim Widget angemeldet werden
- Handler Interfaces (Auszug):
  - ChangeHandler - onChange(ChangeEvent)
  - ClickHandler - onClick(ClickEvent)
  - FocusHandler - onFocus(FocusEvent)
  - BlurHandler - onBlur(BlurEvent)
  - KeyDownHandler - onKeyDown(KeyDownEvent)
  - KeyUpHandler - onKeyUp(KeyUpEvent)
  - MouseDownHandler - onMouseDown(MouseDownEvent)
  - ...
- (siehe auch API Dokumentation)

EventHandler für einen Button

- Button:

  - ClickHandler anmelden um auf drücken des Knopfes zu reagieren

```
Button loginButton = new Button("Anmelden");
loginButton.addClickHandler(new ClickHandler() {
  public void onClick(ClickEvent event) {
    //...z.B. Meldung anzeigen
  }
});
loginButton.setEnabled(false);
```

GWT in der näheren Betrachtung:

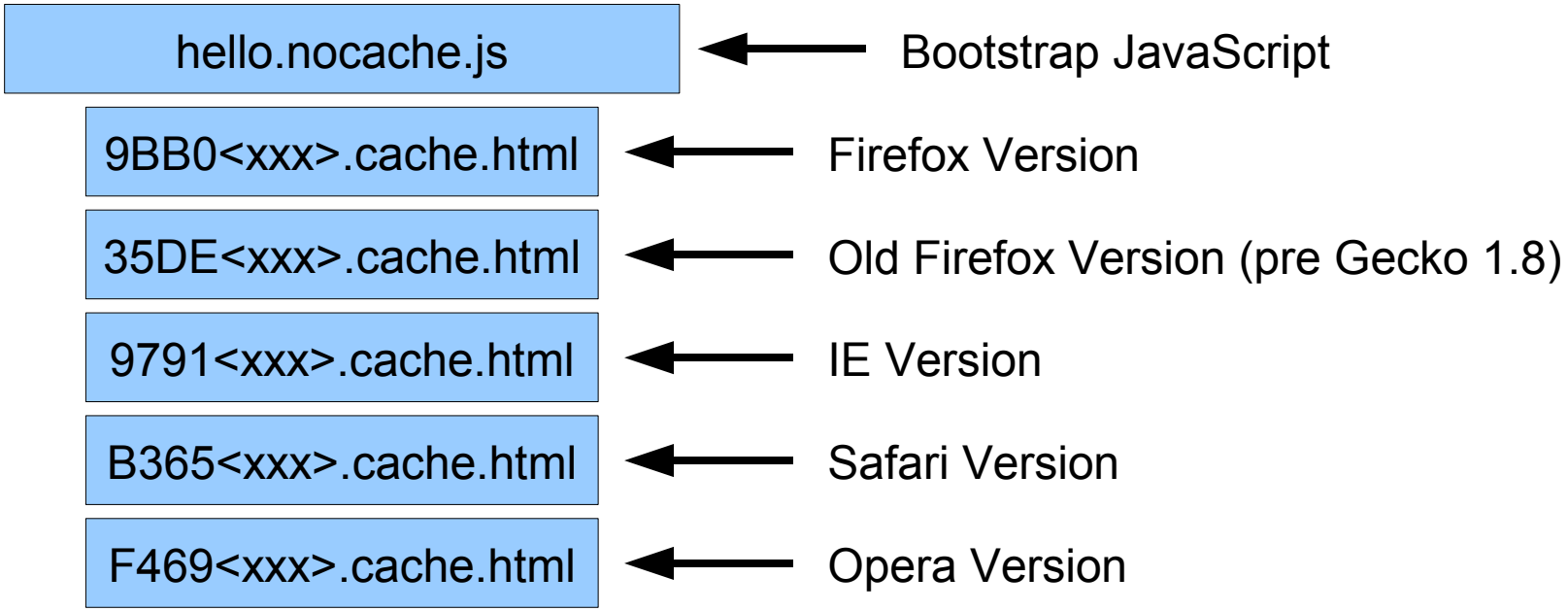
Deferred Binding...

Deferred Binding

- Das Problem:
  - JavaScript unterstützt kein reflection und kein dynamic classloading
- Die Lösung:
  - deferred binding - "dynamic class-loading that occurs at compile time instead of execution time."
- Der Vorteil:
  - optimale Größe und minimale Startzeit
- Beispiele:
  - Browser Support
  - Internationalisierung

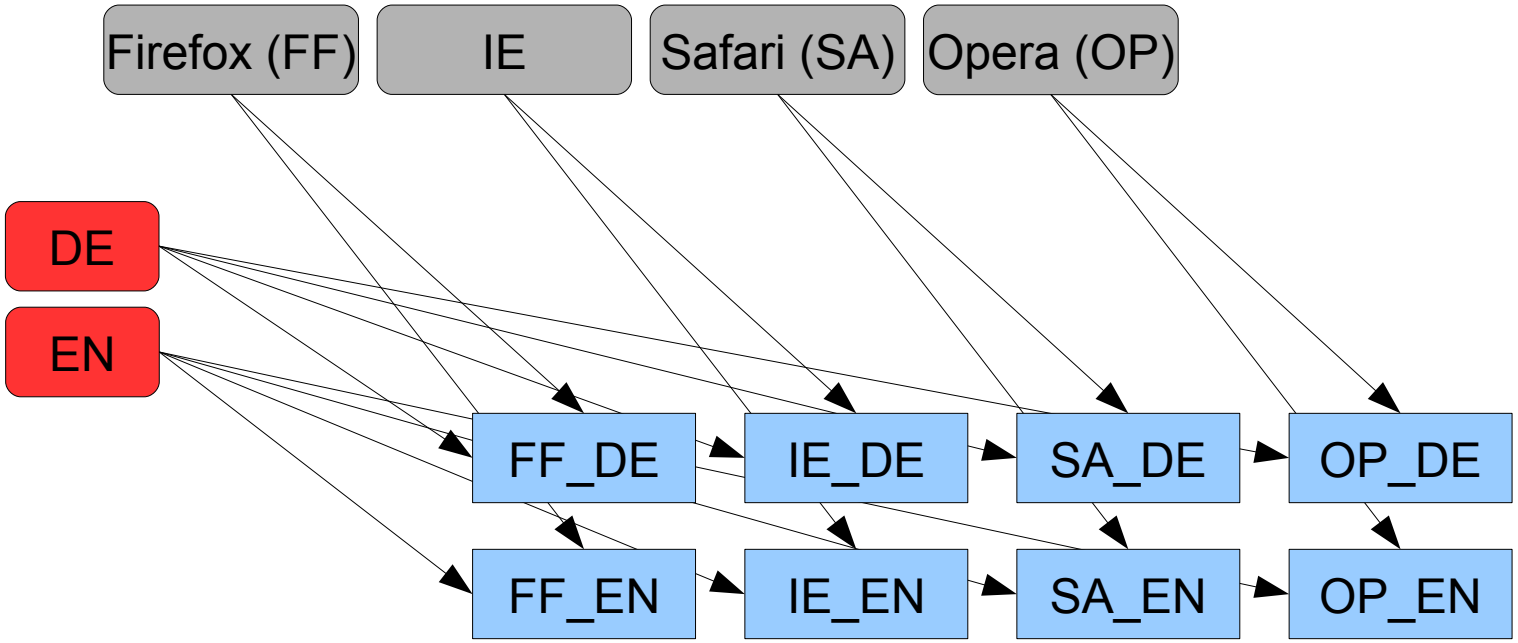
Deferred Binding

- Pro Browser wird eine eigene Version erzeugt



Deferred Binding

- Bei Verwendung von Internationalisierung



GWT in der näheren Betrachtung:

Remote Procedure Calls...

Remote Procedure Calls

- Traditionelle Webanwendungen (ohne AJAX) laden neue Html Seiten
- GWT (AJAX) Anwendungen verwenden einen RPC Mechanismus um Daten mit dem Server auszutauschen
  - Ähnlich zu sonstigen Client/Server Anwendungen
  - Servlet auf Serverseite
- GWT verbirgt die Komplexität
  - Zwei Interfaces und ein Servlet erstellen
  - GWT übernimmt das marshalling/unmarshalling der Objekte
- Aufrufe sind asynchron
  - Flüssige user-experience durch Callbacks

RPC Beispiel - serverseitig

```
@RemoteServiceRelativePath("Speak")
public interface StringService extends RemoteService {
  public String hallo(String s);
}
```

```
public interface StringServiceAsync {
  public void hallo(String s, AsyncCallback<String> callback);
}
```

```
public class StringServiceServlet extends RemoteServiceServlet
implements StringService {
  public String hallo(String s) {
    return "hello "+s;
  }
}
```

RPC Beispiel - clientseitig

```
final StringServiceAsync speakService =
                          (StringServiceAsync)GWT.create(StringService.class);

final AsyncCallback<String> callback = new AsyncCallback<String>() {
  public void onSuccess(String result) {
    label.setText(result);
  }
  public void onFailure(Throwable caught) {
  }
};

button.addClickListener(new ClickListener() {
  public void onClick(Widget sender) {
    speakService.hallo("Frank", callback);
  }
});
```

RPC Beispiel - clientseitig

```
<module>
<inherits name='com.google.gwt.user.User'/>
<inherits name='com.google.gwt.user.theme.standard.Standard'/>

<entry-point class='de.emesit.campus.client.RpcUI'/>

<servlet path="/Speak" class="de.emesit.campus.server.StringServiceServlet"/>

<stylesheet src='RpcUI.css' />
</module>
```

GWT in der näheren Betrachtung:

ImageBundle...

ImageBundle

- Optimierte Behandlung von Bildern
- Die Probleme:
  - Jedes Bild erzeugt einen eigenen HTTP Request
  - Beim Starten 'Freshness Check' für Bilder im Cache
    - für unveränderte Bilder folgt eine HTTP 304 Antwort ("Not Modified")
    - unnötige freshness checks:
      - unnötige Verzögerung der Anzeige der Bilder
      - Nach HTTP 1.1 ist die Anzahl der Requests auf 2 pro Domain und Port limitiert, dadurch werden andere Requests geblockt
  - Bouncing UI Startup wegen Platzhaltern für Bilder (diese haben in der Regel eine andere Größe als die echten Bilder)

ImageBundle

- Die Lösung: ImageBundle / ImageBundleGenerator
  - Bündeln der statischen Bilder in einem einzigen Bild
    - ImageBundles werden beim Kompilieren automatisch durch den ImageBundleGenerator erzeugt
  - Verwendung von Clip Regions um den gewünschten Bereich darzustellen
- Der Vorteil
  - ein HTTP Request pro ImageBundle
  - keine freshness checks ("great caching story")
  - kein Bouncy startup
  - Größe des ImageBundle ist deutlich kleiner als die Summe der Größen der Einzelbilder
  - Compile-time Existenzprüfung der Bilder

ImageBundle - Beispiel

```
public interface MyImageBundle extends ImageBundle {
  public AbstractImagePrototype new_file();

  @Resource("open_file.png")
  public AbstractImagePrototype openFile();

  @Resource("de/emesit/campus/images/save_file.png")
  public AbstractImagePrototype saveFile();
}
```

ImageBundle - Beispiel

```
public class MySimpleUI implements EntryPoint {
  private static final MyImageBundle IMGBUNDLE =
                                        GWT.create(MyImageBundle.class);
  public void onModuleLoad() {
    //...
    Image newFileImage = IMGBUNDLE.new_file().createImage();
    //...
}
```

GWT in der näheren Betrachtung:

History...

History (Back Button) Unterstützung

- AJAX Applikation => kein History Support
  - da nur eine Seite (Hostpage)
- GWT (aber auch andere AJAX Frameworks):
  - Zustand in URL Fragment Identifier (#)
  - Dadurch:
    - Feedback für Benutzer
    - Bookmarkable
  - IFrame in Hostpage einbetten

```
<iframe src="javascript:''" id="__gwt_historyFrame"
        style="width:0;height:0;border:0"/>
```

  - History.newItem(String) aufrufen
  - ValueChangeHandler implementieren

GWT in der näheren Betrachtung:

JavaScript Native Interface...

Natives JavaScript einbetten

- Durch JavaScript Native Interface (JSNI)
- Warum?
  - Lowlevel JS Funktionen aufrufen
  - Verwenden externer JS Bibliotheken (z.B. scriptaculous)
- Beispiel:

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

- Aus JavaScript Zugriff auf Java Methoden, Instanz- und Klassenvariablen, und anders herum
- Aber mitunter trickreich
  - Cross Browser JS

Ausblick auf GWT 2.0...

Ausblick auf GWT 2.0

- In-Browser Hosted Mode
  - Hosted Mode läuft mit jedem externen Browser
  - Vorteil:
    - Verwendung von Entwicklungswerkzeugen wie Firebug
    - (unter Linux war der Hosted Browser ein uralter Mozilla)
    - Bessere Interaktion mit anderen Technologien (z.B. Flash)
- Compiler Enhancements
  - schnellere Kompilierung
    - Parallelisierung (-localWorkers=4)
    - Draft Compile
  - kleineres Kompilat

Ausblick auf GWT 2.0

- Entwickler unterstütztes Code Splitting
  - Applikation wird aufgeteilt in kleinere Häppchen
  - Dadurch schnellerer Applikationsstart,
  - GWT.runAsync() im eigenen Code definiert einen Split Point
- Predictable Layout
  - Besseres regelbasiertes Layout
  - Kein Reagieren mehr auf WindowResizeEvent
  - Neue Panels für besseres Layouting
- ClientBundle (ImageBundle war nur der Anfang)
  - beliebige Text-/Xml-/Css-Resourcen zu Bundles zusammenfassen
  - Vorteil:
    - ...wie bei ImageBundle...

ClientBundle - Beispiel

```
interface MyBundle extends ClientBundle {
  @Source("smiley.gif")
  ImageResource smileyImage();

  @Source("app_config.xml")
  TextResource appConfig();

  @Source("wordlist.txt")
  ExternalTextResource wordlist();

  @Source("manual.pdf")
  DataResource ownersManual();

  @Source("super-fancy.css")
  CssResource superFancy();
}
```

ClientBundle Killerfeature: CssResource

```
//Verwenden von Konstanten:
@def hardToMissThickness 8px;
@def scaryColor #F00;
.error-border {
    border: hardToMissThickness solid scaryColor;
}
//Verwenden von Bedingungen
@if user.agent safari {
    .error-border { -webkit-border-radius: 4px; }
} @elif user.agent gecko {
    .error-border { -moz-border-radius: 4px; }
}

// !!!...und mehr... !!!
```

Demo...

Noch Zeit?

Demo

Vielen Dank!

Rüdiger Schobbert

emesit GmbH & Co. KG