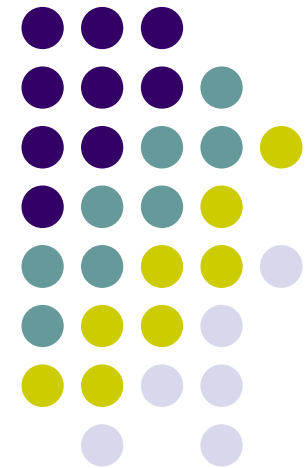


Google Web Toolkit (w praktyce)



Opowiem Wam o...

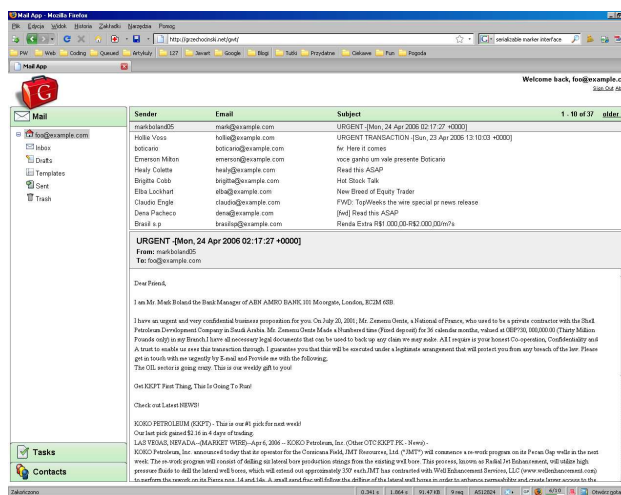


- Komunikacji
 - Automatyzacji
 - Testowaniu
 - Kompilowaniu
 - Operacjach bazodanowych
 - Integracji
 - Internacjonalizacji
 - Deploymencie
- GWT-RPC
 - integracja z Maven2
 - integracja M2+GWT+JUnit
 - kompilator GWT i „deferred binding”
 - JPA, DTO
 - Java Sript Native Interface (JSNI)
 - i18n w GWT
 - Tomcat Lite i inni



GWT, a Java

- GWT (jako klient) może działać na każdym serwerze WWW
- JRE? Dzięki, nie trzeba :)
- Jak się komunikować z usługami?





GWT-RPC

- Rdzenny mechanizm komunikacji w GWT
- „Gie-wu-teizm” :)
- Serwer udostępnia zasoby, klient wywołuje je w sposób asynchroniczny



Asynchroniczność

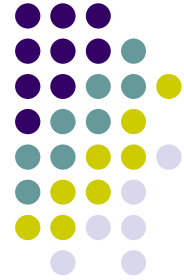
- Jeden wątek JavaScriptu
- Wywołania zwrotne (ang. *callbacks*)
- GWT nie obsługuje komunikacji synchronicznej
- Błogosławieństwo
 - problemy współbieżności znikają
- Przekleństwo
 - przywyczajenie do cyklu proceduralnego (żądanie -> odpowiedź).



Serializacja wg GWT

- Obiekt transportowy serwer<->klient musi:
 - Implementować [IsSerializable](#) lub [Serializable](#), wprost lub dziedzicząc z nadklasy
 - jego nie finalne, nie przejściowe (non-transient) pola są serializowalne
 - Mieć bezargumentowy konstruktor

- GWT uwzględnia słowo kluczowe transient. Przejściowe pola nie są serializowane, a przez to nie są przesyłane poprzez GWT-RPC



GWT-RPC Demo

- Klasa danych – `StockPrice.java`
 - marker `Serializable` lub `IsSerializable`
- U klienta
 - dwa interfejsy
 - synchroniczny – `StockPriceService.java`
 - rozszerza `RemoteService`
 - na potrzeby wewnętrzne GWT
 - asynchroniczny
 - `StockPriceServiceAsync.java`
 - wszystkie metody są 'void'
 - żadna z metod nie rzuca wyjątku
 - każda metoda ma dodatkowy parametr `AsyncCallback`



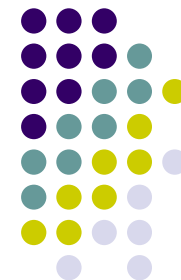
GWT-RPC Demo

- Na serwerze
 - Implementacja serwisu
 - `StockPriceServiceImpl.java` rozszerza `RemoteServiceServlet`
- Użycie
 - zdefiniowanie serwletu z usługą w `<Module>.gwt.xml`
 - uzyskanie asynchronicznego interfejsu poprzez `GWT.create(Class)`
 - zdefiniowanie `AsyncCallback`
 - `onSuccess()`
 - `onFailure()`
 - wywołanie zdalnej procedury z przekazanym parametrem `AsyncCallback`



Komunikacja - inaczej

- XML
 - POX – proste żądania HTTP
 - `HttpRequest`
 - REST - zaawansowane żądania
 - `RequestBuilder` – zmiana nagłówek, kodów statusów etc.
 - nie osiągalny w 100% :(
 - Analiza XML - moduł XML z `XMLParser` i `Document`
- GWT API dla JSON (JavaScript Object Notation)
- Flash lub aplety Javy w aplikacjach GWT jako klienci SOAP
- strumieniowe przesyłanie danych - architektura Comet



Budowanie projektu

- `<project>-compile.cmd`
- `<project>-shell.cmd`
- narzucona struktura katalogów - CoC
 - `Module.gwt.xml`
 - pakiety
 - `client`
 - `public`
 - `server`
 - `rebind` (kod czasu kompilacji)



A może Maven2 ?

"If you can't use it with maven, it's useless"

- GWT Maven Plugin

- definiujemy repozytoria
- konfigurujemy wtyczkę
- definiujemy zależności
- ...volia!

The screenshot shows the Google Code page for 'gwt-maven'. The page title is 'gwt-maven' with the subtitle 'Maven Support for Google Web Toolkit'. There are navigation tabs for 'Project Home', 'Downloads', 'Wiki', 'Issues', and 'Source'. Below the tabs, there are links for 'Summary' and 'Updates'. The main heading is 'Maven 2 and Maven 1 support for GWT'. The text below states: 'There are multiple versions of GWT-Maven, a Maven 2.x version and a Maven 1.1 version. Both versions have the same set of basic goals (which are optional):' followed by a bulleted list of goals: 'Run the GWTShell (Hosted mode, both with embedded Tomcat and -noserver)', 'Run the GWTShell and connect to it with a debugger', 'Run the GWTCompiler', 'Run GWTTestCase and GWTTestSuite based tests', and 'Generate I18N Constants and Messages interfaces'.

```
<plugin>
  <groupId>com.totsp.gwt</groupId>
  <artifactId>maven-googlewebtoolkit2-plugin</artifactId>
  <version>2.0-beta23</version>
  ⋮
</plugin>
```

Demo



Maven2

Działa!



- `$ mvn [?]`

- `gwt:gwt` - uruchamia powłokę (hosted mode)
- `gwt:compile` - uruchamia kompilator GWT
- `gwt:debug` - hosted mode z możliwością podłączenia debugera
- `gwt:testGwt` - uruchamia testy GWT
- `gwt:war` - tworzy paczkę war
- `gwt:mergewebxml` - tworzy wynikowy web.xml – nie wywoływany samodzielnie!
- `gwt:generateClientBeans` - tworzy obiekty DTO strony klienta w katalogu źródłowym

- Zapominamy o `<project>-compile.cmd` i `<project>-shell.cmd`
 - Inna struktura katalogów - `<project>-gwt.xml` problem





Unit-testy

- testuj model i kontroler, nie widok !
- extends `GWTTestCase`
 - testujemy JavaScript!
- testowanie GWT-RPC – metody opóźniające
- uwaga na nazwy klas z `TestCase`'ami!
 - problem z surefire



Kompilator GWT

- Raczej „translator” Java -> JavaScript.
 - Nie kompiluje do bytecodeu !!!
 - płacisz za to, czego używasz (optymalizacja)
- Deferred binding – opóźnione wiązanie
 - „odpowiedź GWT na refleksję Javy” (cytat z GWT Docs)
 - **generuje gotowe wersje aplikacji – optymalizacja !!!**
 - Anglik nie ściąga francuskiej lokalizacji
 - użytkownik Firefox’a nie ściąga plików dla IE
 - 4 języki i 4 przeglądarki = 16 kompilacji, 16 wersji aplikacji !



Kompilator GWT

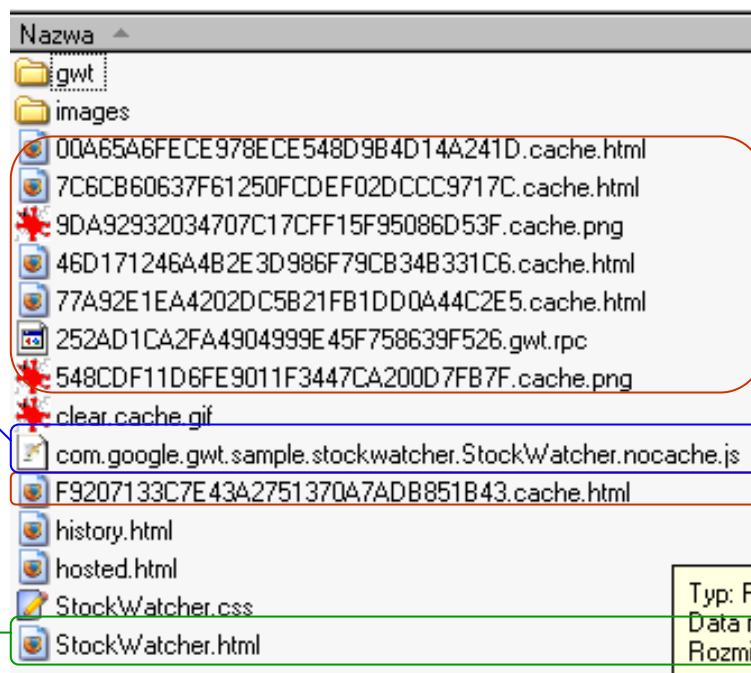
- Wynik pracy:

2

Główny JS GWT, wykrywa właściwą wersję aplikacji w oparciu o klienta (przeglądarka, język) oraz wczytuje monolityczną aplikację

1

Strona startowa (host page), wczytuje <Module>.nocache.js



3

Hash z kodu źródłowego, umożliwia buforowanie stron po stronie klienta

Kompilator GWT - ciekawostki



- `new StringBuffer().append(String toAppend)`
- `-style`
 - OBF - `function A0(){this.B0();}`
 - PRETTY - `function _append2(toAppend){ ... }`
 - DETAILED - `function java_lang_StringBuffer_append(toAppend){ ... }`



Baza danych

- Model domenowy (JPA)
 - tylko dla prostych aplikacji
- Problem sesji i serializacji z lazy-loadingiem
- Data Transfer Object(s)
 - dwa modele (domenowy i DTO)
 - automatyzacja

JavaScript Native Interface



- Definiowanie implementacji metod klasy Javy za pomocą JavaScriptu
- Integracja, użycie zewnętrznych bibliotek JS

```
public class Alert{
    public Alert(){
        super();
    }

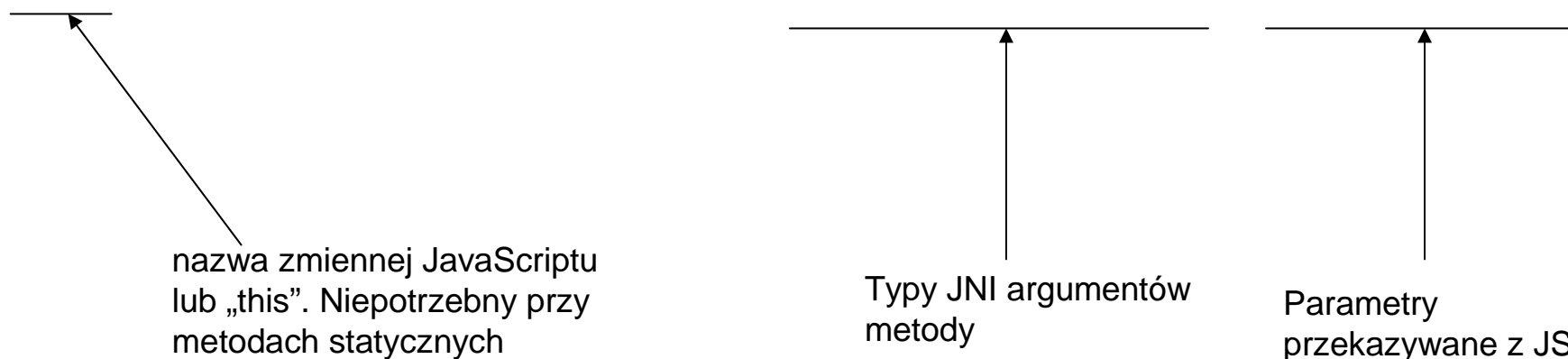
    public native void alert(String message)
    /*- { #2
        alert(message)1
    }-*/;
}
```



JavaScript Native Interface

- Wywołanie Javy z JavaScript'u

```
this.@com.my.ClassName::myMethod(Ljava.lang.String;)( „some string” );
```





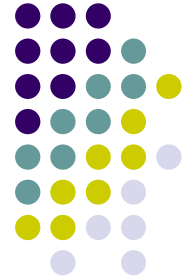
Internacjonalizacja

- Dodajemy moduł
 - `<inherits name="com.google.gwt.i18n.I18N" />`
- Static String Internationalization
- Dynamic String Internationalization
- implementacja Localizable



Static String Internationalization

- plik .properties (nareszcie) w UTF-8
- interfejsy Constants i Messages
- `<extend-property name="locale" values="pl" />`
- Kompilator generuje 2x więcej plików
- Użycie:
 - `<meta name="gwt:property" content="locale=pl">`
 - `http://www.example.org/myapp.html?locale=pl`



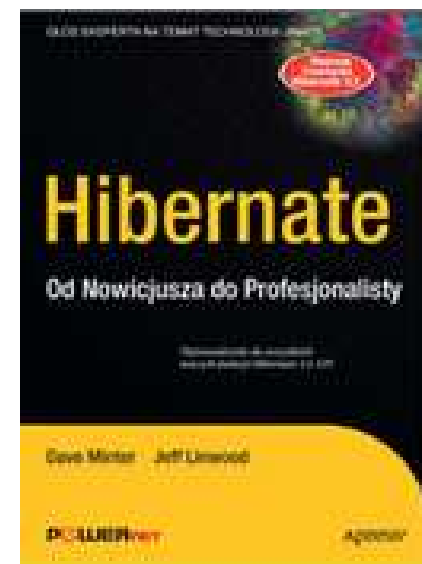
Tomcat Lite i inni

- GWT używa (w hosted mode) okrojonej i nieco niestandardowej wersji Apache Tomcat (5.5?)
- GWTSchell i GWT Maven plugin pozwalają na użycie innego serwera (-noserver i <noServer>)
 - Sami musimy skonfigurować nowy serwer.



PowerNET

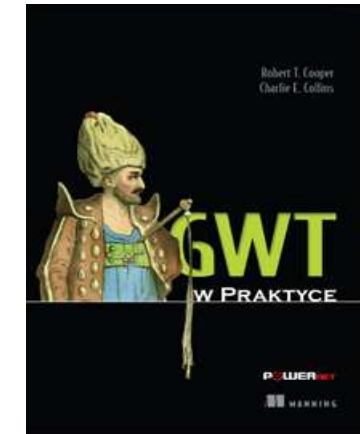
- Hosting, tworzenie stron WWW i ... wydawnictwo ;)
- 4 książki
- Apress, Manning
- konkurencja dla Helionu



Jak zacząć?



- tutorial GWT na GoogleCode
- książka „GWT w praktyce”
 - autorzy GWT Maven Plugin
 - (niestety) GWT 1.4
- Prezentacja
 - <http://grzechocinski.net>
- Gotowy projekt - StockWatcher
 - <http://grzechocinski.net/projekty/java-ee/StockWatcher>
 - 4 kroki:
 - \$ svn co <http://svn.grzechocinski.net/public/StockWatcher>
 - \$ cd StockWatcher
 - \$ mvn clean jetty:run-war ... i idziemy na kawę ;)
 - Pod <http://localhost:8080/> jest nasza aplikacja!



Dzięki za uwagę!

