Corso di “Sviluppo di applicazioni Web”

Docente: Lorenzo Gallucci

GWT (Introduzione)

Perchè Google Webtoolkit?

- Framework per applicazioni basato sul Web
- AJAX enabled
- Web 2.0
- <*inserire qui la propria buzzword preferita*>

Perchè Google Webtoolkit?

- Implementa un'interfaccia utente basata su HTML in Java
- La compilazione produce HTML e Javascript
- Abilita le applicazioni all'uso di AJAX utilizzando un protocollo binario (proprietario) per il remoting
  - Può usare altri protocolli (es. JSON, XML)

Cos’è AJAX?

- Asynchronous Javascript And Xml
- Costruito intorno all’oggetto JavaScript *XMLHttpRequest*
  - Introdotto da Microsoft con IE
  - Presente ormai su tutti i browser, ma con piccole differenze tra un’implementazione e l’altra

Cos’è AJAX?

- Le richieste in AJAX corrispondono a chiamate HTTP
  - Sono *asincrone* ; ad ogni richiesta è necessario fornire un oggetto callback che verrà notificato
- AJAX induce naturalmente un modello basato sulla gestione degli eventi

Tratti salienti di Javascript

- Originariamente creato da Netscape
  - Basato su una specifica ECMA (ECMAScript)
- Non è un linguaggio strongly-typed
  - … ma nemmeno del tutto dynamically-typed
- Le varie implementazioni presenti nei browser differiscono in maniera signifcativa
- Fragile e difficile da debuggare

Che vantaggi ha AJAX?

- Suona bene
- Fa un figurone nel curriculum!
- …

AJAX … seriamente

- Consente di ottenere interazioni richiesta/risposta di grana più fine
  - Non è necessario rigenerare l'intera pagina HTML per aggiornare i dati
- I dati possono venire recuperati solo al momento del bisogno
- Interfaccia utente più ricca e interattiva
  - Molta logica può essere spostata o duplicata vicino all'utente, nel browser

Svantaggi di AJAX

- Vari browser implementano la specifica con piccole differenze
- Javascript non è un linguaggio ad oggetti tradizionale
  - Non ha tool di analisi al livello di altri linguaggi
- È necessario conoscere il modello DOM e le sue API per manipolare gli elementi grafici

Come risolvere i problemi di JavaScript?

- Tramite l'uso di frameworks!
  - Scriptaculous, DWR, altri fatti-in-casa, ecc.
- Vi sono molti contendenti
- Non è emerso ancora uno standard de facto
  - Alcuni framework, come *Prototype,* sono però molto usati

Come risolvere i problemi di JavaScript?

- Tra i vantaggi principali nell'uso di un framework annoveriamo:
  - Uso di codice ben testato come base per la nostra applicazione
  - Disponibilità di codice adattato alle idiosincrasie dei vari browser

Come può rispondere un'azienda come Google?

- Anzitutto, riconoscendo questi problemi

- Poi, ponendosi delle semplici domande:
  - Come può uno sviluppatore costruire applicazioni Web efficaci, potenziate dall'approccio AJAX?
  - Come si può disporre di una varietà di strumenti di analisi e debugging per la propria applicazione Web?

Semplice, sviluppando **in Java!**

Google Web Toolkit in breve:

- Un’architettura per “rich client” orientata allo sviluppo di applicazioni Internet interattive
- Google lo descrive così:

“Google Web Toolkit (GWT) is a Java software development framework that makes writing AJAX applications like Google Maps and Gmail easy for developers ***who don't speak browser quirks as a second language****.”* http://code.google.com/webtoolkit

Google Web Toolkit in breve:

- Concettualmente simile a Swing, ma adattata ad HTML e dotata di capacità di remoting via Web
- Include nel pacchetto di base molti widgets e componenti elementari, meccanismi di RPC e supporto nativo per l'interoperatività con Javascript

Google Web Toolkit in breve:

- Supporta la gestione dell'history del browser
- Abilita il debugging
- Compatibile con molti browser
- Dispone di un modulo d'integrazione con JUnit
- Supporta l'internazionalizzazione (i18n)
- Genera versioni ad hoc dell'applicazione per le diverse tipologie di utenti

Swing

- È una UI basata su Panels e Layout Managers
- Dispone di Widgets per visualizzare alberi, liste, testo, etichette, ecc.
- Fortemente basata sulla gestione di eventi (Action Listeners, eventi tastiera, mouse e focus, ecc.)
- L'implementazione si basa sul concetto di UI Delegate

GWT

- I widgets forniti includono i “soliti noti” - testo, password, tabelle, astrazioni dell’HTML
  - Le versioni più recenti si sono arricchite di widget più complessi come i *suggest boxes*
  - Un gran numero di librerie forniscono altre tipologie di componenti
  - Alcune di queste sono basate sul wrapping di librerie Javascript esistenti (es. ExtJS)
- Il layout dei componenti è basato sui pannelli (verticale, orizzontale, deck, dock, scroll, tab ecc.)
- Fortemente basata sulla gestione di eventi (Cambio di contenuto, click, tastiera, mouse, focus, scrolling, ecc.)
- Utilizza anch’esso il pattern Delegate

Un po’ di coordinate

- Può essere scaricato da http://code.google.com/webtoolkit
- Tra le piattaforme supportate: Windows, Linux (GTK+) e Mac OS/X
- La licenza non impedisce lo sviluppo di applicazioni opensource

Un po’ di coordinate

- Esistono vari plugin per Eclipse
  - Alcuni (quelli commerciali) consentono di sviluppare graficamente l’interfaccia utente
  - Altri (quelli gratuiti) si limitano all’integrazione delle fasi di un’applicazione GWT nell’ ambiente Eclipse

Cosa contiene il pacchetto GWT?

- Utility a linea di comando, nella versione specifica per il proprio OS: projectCreator, applicationCreator, i18nCreator, junitCreator
- Jar da usare per lo sviluppo sulla piattaforma scelte: gwt-dev-xxx.jar – dove xxx sta per win32, linux, mac
- Un jar da usare in fase di deploy: gwt-user.jar
- Applicazioni d'esempio
- Documentazione dell'API

Modalità operative

- GWT supporta due modalità operative:
  - **Hosted mode:**
    - utilizza un'istanza di Tomcat built-in come ambiente di esecuzione
    - L'applicazione gira come bytecode Java all'interno della JVM
    - È il più usato durante lo sviluppo, poiché, l'ambiente di esecuzione determina la disponibilità di facilities per lo sviluppo, senza abbandonare l'IDE (es. Eclipse)
  - **Web mode :**
    - L'applicazione è, in questo caso, composta solo da HTML e da codice Javascript compilato dal nostro codice Java originario, tramite il compilatore Java-to-JavaScript di GWT
    - Al momento del deploy su un ambiente di produzione, solo l'accoppiata JavaScript/HTML dovrà essere sottoposta ai web server

Come partire?

- Si possono lanciare due degli script forniti:
  - projectCreator crea i file di progetto
    - Directory Src/bin
    - Files .project e .classpath
  - applicationCreator crea i file di supporto per l'applicazione
- Il plugin Eclipse offre funzionalità simili

Struttura di progetto

- <nome modulo> (ad es. it/saw/prova) - Il package radice del progetto, contiene un file XML di descrizione del modulo
- <nome modulo>/client – File sorgente client-side
- <nome modulo>/server – Codice server-side code
- <nome modulo>/public – Risorse statiche che possono essere richieste all'applicazione Web (HTML, immagini, ecc.)

Applicazione GWT

- È necessario un punto di accesso (*main* in altri ambienti), definito “module entry point”

```
public interface EntryPoint
{
    void onModuleLoad();
}
```

- Cos’è un modulo?
- Definito da un file di configurazione XML
- Specifica un punto di accesso
  - Può trattarsi di una classe che genera l’HTML iniziale
- Indica un mapping delle servlet, da usare in Hosted Mode
- Può ereditare da altri moduli

Alcuni dei package più significativi:

- user.client.rpc – Implementazione client side delle classi per RPC (IsSerializable, AsyncCallback)
- user.client.ui –Widgets, Panels e altri classi di supporto alla GUI

Alcuni dei package più significativi:

- core.client:
  - GWT (uncaught exception handler)
  - JavascriptException
  - Interfaccia EntryPoint
- user.client – Browser history, manipolazione DOM, event handling ecc

Package user.client.ui

- Contiene alcuni elementi di base delle UI: TextBox, PasswordTextBox, Grid, Label, Listbox, MenuBar, MenuItem, Tree, HTMLTable
- Tutti gli elementi discendono da Widget
- Le astrazioni di Panel : Panel, VerticalPanel, HorizontalPanel, DeckPanel, DockPanel, RootPanel
- I panels sono oggetti compositi e supportano gerarchie guidate dal part-of

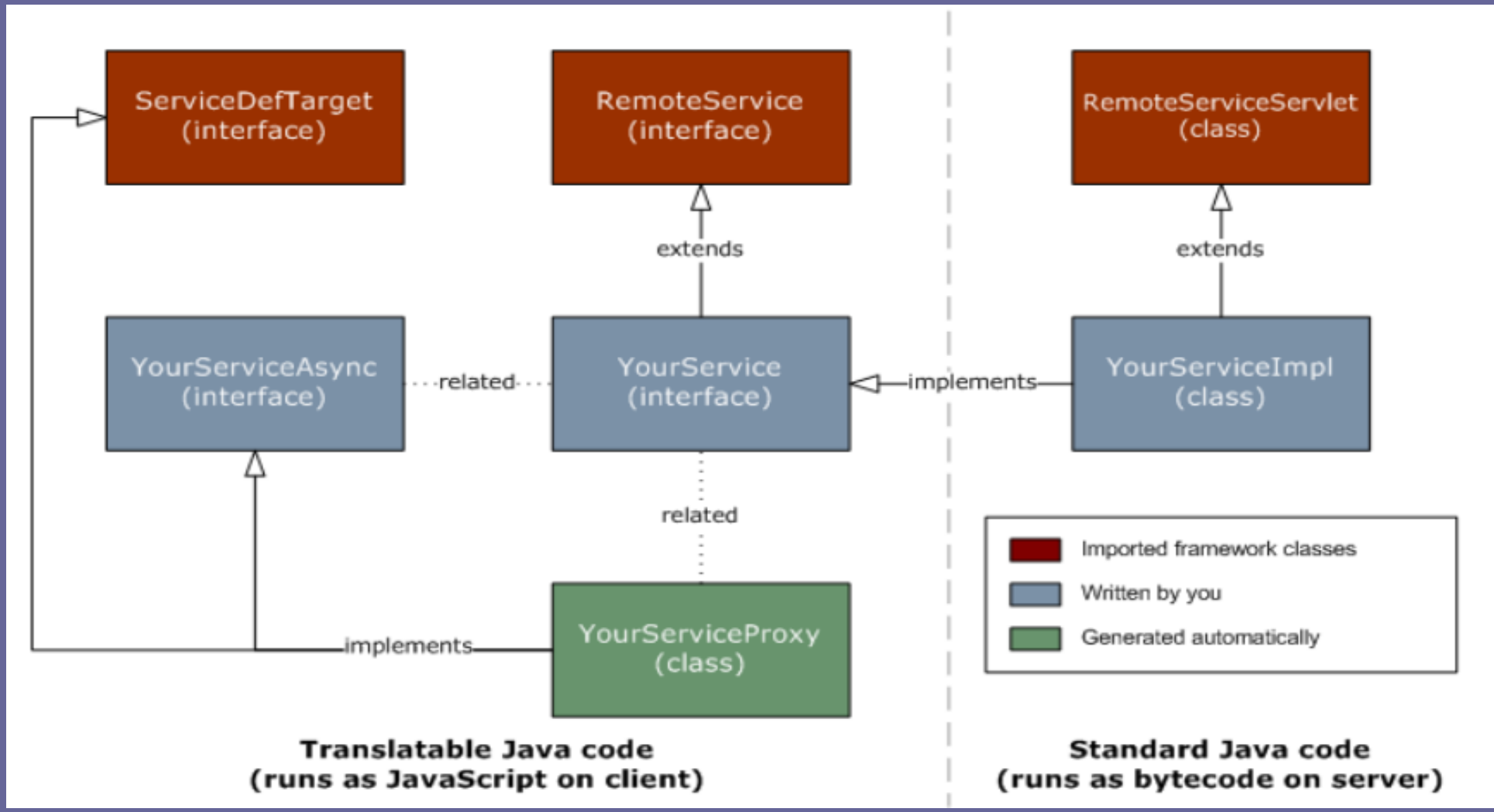
Gestione di eventi

- GWT supporta un'ampia scelta di interfacce per la gestione di eventi, tra cui: KeyboardListener, MouseListener, ClickListener, ChangeListener, FocusListener, ecc.
- Gli elementi dell'interfaccia utente hanno metodi per aggiungere e togliere listener per ognuno dei tipi supportati
  - Ad esempio, SourcesClickEvents contiene i metodi per la manipolazione dell'elenco di ClickListener

RPC in GWT

- Basato su un protocollo proprietario
- Utilizza AJAX
- Lato server, una Servlet è usata per ricevere le richieste
- Il procedimento di creazione è basato su un numero di passi ben definiti

Service “Plumbing” Diagram



Come si scrive un servizio in RPC?

- Creiamo un'interfaccia client-side che rappresenta il servizio
  - Deve estendere RemoteService di GWT, va annotata con @RemoteServiceRelativePath("<path della servlet>")
  - È necessario che tutti i parametri ed il tipo di ritorno estendano Serializable
- Lato server, scriviamo una Servlet, che estende RemoteServiceServlet di GWT ed implementa la prima interfaccia
  - In Hosted mode, <servlet path="*<path della servlet>*" class="*<classe della servlet>*" /> nel file .gwt.xml richiede l'istanziazione
- Creiamo un'interfaccia equivalente, ma asincrona; per ogni metodo dell'interfaccia originaria, ne creiamo uno equivalente, che:
  - Non restituisce nulla (è *void*)

Effettuare la chiamata

- Creaimo un’istanza del proxy creato da GWT per il servizio, tramite GWT.create (<interfaccia del servizio>.class)
- L’oggetto che ci viene restituito è però del tipo della seconda interfaccia (quella asincrona)

Effettuare la chiamata

- Ogni metodo chiamato sull'oggetto restituito agisce sulla servlet, tramite HTTP
- L'oggetto AsyncCallback che bisogna fornire ha due metodi:
  - onSuccess(T), chiamato se la richiesta ha esito positivo
  - onFailure(Throwable), in caso di fallimento