Interaction Design and Architecture(s) Journal - IxD&A, N. 7-8, 2009, pp. 99-100

ARE: Ada Rendering Engine

Stefano Penge
Lynx
Via Ostiense, 60/d
00154 - Roma
steve@lynxlab.com

Maurizio Mazzoneschi
Lynx
Via Ostiense, 60/d
00154 - Roma
graffio@lynxlab.com

Vito Modena
Lynx
Via Ostiense, 60/d
00154 - Roma
vito@lynxlab.com

ABSTRACT

E' ormai pratica diffusa, nello sviluppo di applicazioni web, l'utilizzo di *template* e di potenti *template engine* per automatizzare la generazione dei contenuti da presentare all'utente. Tuttavia a volte la potenza di tali engine è ottenuta mescolando logica e interfaccia, introducendo linguaggi diversi da quelli di descrizione della pagina, o addirittura inventando nuovi linguaggi dedicati.

ARE (ADA Rendering Engine) è pensato per gestire l'intero flusso di creazione del contenuto HTML/XHTML dinamico, la selezione del corretto template, CSS, JavaScript e la produzione dell'output separando completamente logica e interfaccia. I templates utilizzati sono puro HTML senza parti in altri linguaggi, e possono quindi essere gestiti e visualizzati autonomamente. Il codice HTML generato è uniforme e parametrizzato.

E' composto da due moduli, CORE (Common Output Rendering Engine) e ALE (ADA Layout Engine).

Il primo (CORE) viene utilizzato per la generazione OO degli elementi del DOM ed è pensato per aiutare lo sviluppatore nella produzione di codice valido rispetto al DTD utilizzato. CORE genera automaticamente gli elementi del DOM in base al DTD impostato nella configurazione

Il secondo (ALE) viene utilizzato come *template engine* per selezionare automaticamente in base ad alcuni parametri (modulo, profilo utente, tipologia del nodo, del corso, preferenze di installazione) il template HTML, i CSS e i file JavaScript appropriati. ALE permette di usare templates di default e microtemplates ricorsivi per semplificare il lavoro del grafico.

I due moduli possono in ogni caso essere utilizzati indipendentemente l'uno dall'altro. E' possibile generare e renderizzare una pagina HTML utilizzando solo CORE oppure inviare gli oggetti CORE al template engine ALE che provvede a renderizzare la pagina HTML. Viceversa è possibile generare HTML senza utilizzare CORE ed inviarlo al template engine ALE

CORE è alla prima release ed è già utilizzato all'interno dei progetti ADA e MAKO.

Tra gli sviluppi previsti: il completamento della libreria per diverse DTD; la creazione di classi di livello superiori che automatizzino compiti ripetitivi (creazione di form, tabelle, etc).

ARE è software libero, rilasciato sotto licenza GPL 2, ed è scaricabile all'indirizzo: http://ada.lynxlab.com

Categories and Subject Descriptors

*D.3.4 [**Programming Languages**]: Processors—Code generation;*

*D.2.11 [**Software Engineering**]: Software Architectures - Languages;*

*D.1.1 [**Programming Techniques**]: Applicative (Functional) Programming*

General Terms

Documentation, Design, Standardization, Languages.

Keywords

Template engine, Model View Controller, Personal Learning Environment, Personal Interface.

1.INTRODUZIONE

Si parla oggi molto di Personal Learning Environment facendo riferimento alla possibilità di personalizzare i contenuti (e le sorgenti) delle informazioni come alternativa ai sistemi rigidi e mono-contenuto. Anche a prescindere dalla validità didattica di questo modello, sarebbe necessario affrontare anche la spinosa questione della personalizzazione delle interfacce degli ambienti stessi, che attualmente va poco oltre la modifica di colori e immagini di fondo.

In questo articolo ci occupiamo di una soluzione che pur essendo sviluppata nell'ambito del processo di sviluppo e riprogettazione di una piattaforma di elearning (ADA) è stata utilizzata in altri contesti, sempre con l'obiettivo di rendere possibile una personalizzazione dell'interfaccia a molti livelli e per esigenze diverse: dalla personalizzazione gestita autonomamente dall'utente a quella legata a contenuti specifici o ai diversi device di interazione.

E' ormai pratica diffusa, nello sviluppo di applicazioni web, l'utilizzo di *template* e di potenti *template engine* per automatizzare la generazione dei contenuti da presentare all'utente. L'idea di base è quella di dividere l'interfaccia in due parti: una statica, fissa, e una dinamica, generate in tempo reale in base al processo. Questo sistema consente un equilibrio tra le esigenze di performance e quelle di dinamicità.[1]

Tuttavia a volte la potenza di tali engine è ottenuta mescolando business logic e presentation logic, in aperta violazione del principio di separazione tra dati, logica e interfaccia,[2] introducendo linguaggi diversi da quelli di descrizione della pagina, o addirittura inventando nuovi linguaggi dedicati. Ad esempio in quasi tutte le applicazioni web i template

1

2

non sono puri documenti HTML ma contengono parti in PHP o in altri linguaggi. Questo rende il template di difficile modifica da parte del grafico al di fuori dell'applicazione stessa.

In altri casi si fa ricorso a linguaggi proprietari per introdurre salti condizionali, inclusioni, computazioni di valori di variabili in base al contesto.[3]

In questo articolo presentiamo una soluzione diversa, ARE, che mira a facilitare la scrittura di codice che consenta la personalizzazione dell'interfaccia utente, pur mantenendo la netta separazione tra *business logic* e *presentation logic*. La soluzione presentata aiuta la progettazione e l'implementazione di applicazioni web che tengano in debito conto le necessità del singolo utente sia in termini di accessibilità che di usabilità.

La soluzione è composta da due parti separate, integrabili fra loro, scritte nel linguaggio PHP 5. E' stata inizialmente sviluppata da Lynx all'interno della piattaforma e-learning ADA (http://ada.lynxlab.com) ma viene usata anche al di fuori di essa (es.: Mako, Avipa).

ARE è software libero, rilasciato con licenza GPL 2.

2.ADA Rendering Engine

ARE (ADA Rendering Engine) è pensato per gestire l'intero flusso di creazione del contenuto HTML/XHTML dinamico, la selezione del corretto template, CSS, JavaScript e la produzione dell'output separando completamente logica e interfaccia. I templates utilizzati sono puro HTML senza parti in altri linguaggi, e possono quindi essere gestiti e visualizzati autonomamente facilitando il lavoro di progettazione grafica dell'applicazione (il grafico può, ad esempio, utilizzare DreamWeaver per disegnare il template delle pagine HTML dell'applicazione). Il codice HTML generato è uniforme e parametrizzato.

E' composto da due moduli, CORE (Common Output Rendering Engine) e ALE(ADA Layout Engine).

2.1Common Output Rendering Engine

CORE viene utilizzato per la generazione OO degli elementi del DOM ed è pensato per aiutare lo sviluppatore nella produzione di codice valido rispetto al DTD utilizzato. CORE genera automaticamente gli elementi del DOM in base al DTD impostato nella configurazione. Questo significa svincolare lo sviluppo dell'applicazione dal formato dell'interfaccia. Per esempio, è possibile avere diverse versioni della stessa applicazione che producono pagine in HTML 4, XHTML 1.0 etc.

D'altra parte, questo sistema consente anche di proteggere l'investimento effettuato per adeguarsi a nuovi standard in corso di definizione. Ad esempio, nel caso in cui si voglia produrre output in standard HTML 5 non sarà necessario apportare modifiche al codice dell'applicazione, in quanto sarà CORE a farsi carico delle eventuali differenze nel rendering degli elementi, oltre a supportare le novità introdotte dalla nuova versione di HTML.

In teoria, CORE permette un'uscita in linguaggi di definizione della pagina diversi dalla famiglia dell'HTML, come ad esempio PDF o SWF.

CORE è alla prima release ed è già utilizzato all'interno dei progetti ADA e MAKO.

2.2ADA Layout Engine

ALE viene utilizzato come *template engine* per selezionare automaticamente in base ad alcuni parametri (modulo, profilo utente, tipologia del contenuto, tipo di device, preferenze di installazione) il template HTML, i CSS e i file JavaScript appropriati. ALE permette di usare template di default e microtemplate ricorsivi per semplificare il lavoro del grafico.

E' possibile avere nella stessa applicazione diverse interfacce di default predefinite (per tutta l'applicazione, per un modulo, per una categoria di utenti o per un singolo utente).

I livelli di preferenza più specifici sovrascrivono i livelli più generali nella selezione della tripletta di layout (template + css + js). Al momento della renderizzazione dell'interfaccia le preferenze dell'utente sono considerate prioritarie sulle preferenze generali dell'applicazione.

L'algoritmo usato per la selezione è il seguente:

- per un dato modulo M, ALE cerca una specifica tripletta di layout (template + css + js) tra quelli forniti dallo stile S specificato nel profilo dell'utente;
- se la tripletta precedente non è disponibile, ALE utilizza lo stile di default del modulo;
- se neanche questo è disponibile, ALE caricherà lo stile definito come default nella configurazione generale.

I template possono contenere codice HTML/XHTML ma non codice PHP o in altri linguaggi.

All'interno di ogni template dovranno essere definiti dei *segnaposto* che saranno automaticamente riempiti con i dati inviati dall'applicazione. ALE si occuperà di mostrare l'output unendo i dati provenienti dall'applicazione con la tripletta di layout selezionata.

3.SVILUPPI

Tra i principali sviluppi di CORE sono previsti:

- il completamento della libreria per diverse DTD;
- il completamento di classi di livello superiore che automatizzino compiti ripetitivi (creazione di form, tabelle, etc).

Per ciò che riguarda ALE, è prevista la semplificazione della gerarchia nell'inclusione dei template, pur mantenendo la granularità nella selezione di template.

4.REFERENCES

[1] Una lettura obbligata a questo proposito è quella dell'articolo di Terence Parr dell'University of San Francisco “Enforcing Strict ModelView Separation in Template Engines” http://www.cs.usfca.edu/~parrt/papers/mvc.templates.pdf

[2] Il paradigma Model/View/Controller .... Vedi G. Krasner and S. Pope. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. Journal of Object Oriented Programming, 1(3):26–49, 1988.

[3] Alcuni esempi molto noti e usati sono Smarty (http://www.smarty.net), Dwoo (http://dwoo.org/)

3