Relazione Tecnologie Web 2014

Andrea Giacomo Baldan, Alberto De Agostini

18 Giugno 2014

- E-mail referenti
  - *E-mail: a.g.baldan@gmail.com*
  - *E-mail: miniotta@hotmail.it*
- Specifiche
  - *URL sito: http://tecnologie-web.studenti.math.unipd.it/tecweb/~utente*
  - *Nome utente:*
  - *Password:*

1 Abstract

L' attivita sportiva

2 Perl

2.1 Organizzazione

Trattandosi di un sito con una buona quantità di contenuti dinamici, è stato studiato un approccio quanto più modularizzato possibile, in modo da garantire maggior chiarezza e manutenibilità, una sorta di *pattern MVC*, dove le *view* sono rappresentate da templates (`.tmpl`) raccolti in una directory completamente separata dal codice, modelli e controller sono contenuti in 3 file contenenti inoltre le funzioni principali necessarie al popolamento dinamico del sito, si è quindi resa necessaria la suddivisione di esse in una gerarchia formata da tre moduli:

- `UTILS` classe padre principale, raccoglie le funzioni di uso generale per il funzionamento e la popolazione delle varie pagine, caricamento ed interfaccia dei vari database XML (Model)
- `UTILS::Admin` classe figlio di UTILS, raccoglie le funzioni strettamente necessarie al backend dell'applicazione, funzionalità di login e mantenimento delle sessioni
- `UTILS::UserService` classe figlio di UTILS, raccoglie le funzioni necessarie al compimento delle operazioni strettamente legate all'utente (e.g CRUD delle proprie generalità), prenotazione risorse

Alcune funzioni all'interno di questi moduli sono state "privatizzate", in quanto funzioni di utilità non direttamente finalizzate all'utilizzo da parte dell'utente (e.g. creazione scheletro tabelle, calcolo e conversione dei giorni della settimana etc.). In particolare ognuno di questi moduli fa da appoggio a rispettivi script utilizzati per effettuare le varie operazione per mezzo di dispatch tables, che consentono di risparmiare un gran numero di operazioni ridondanti e di automatizzare il piu possibile le operazioni da eseguire, aumentando inoltre la separazione tra codice e contenuto, avvicinandosi ad un approccio MVC:

- `load.cgi` si appoggia ad `UTILS` ed è il motore di popolamento principale del sito, ogni pagina accessibile è generata e popolata da questo script, per mezzo di dispatch tables
- `admin.cgi` si appoggia ad `UTILS::Admin`, controparte backend di `load.cgi`, ogni pagina della parte amministrativa è generata da questo script
- `process.pl` script necessario alle basilari operazioni di modifica/popolamento

risorse/pagine (CRUD)

- `user_jobs.pl` controparte frontend di `process.pl`, tutte le operazioni che l'utente può effettuare sono gestite da questo codice

Vi sono infine `login.pl`, `login.cgi` e `logout.pl`, piccoli script atti solo all'autenticazione dell'utente, *frontend* e *backend* ed alla chiusura di eventuali sessioni aperte. `vbooked.pl` è infine lo script utilizzato per visualizzare le tabelle di prenotazione via AJAX senza il bisogno di effettuare *refresh* della pagina.

2.2 Sistema di popolamento templates

Ogni *route* richiama il *dispatcher* da `UTILS` e passa un *hash* contenente i parametri necessari al popolamento del template richiamato, che inoltre possiede lo stesso nome della *route* appunto.
Da `load.cgi` attraverso l'oggetto `$utils` e la dispatch table viene automaticamente richiamato e popolato il template corretto:

Dispatch table all'interno di `load.cgi`:

```
my %routes = (
  'home'          => \&index,
  'impianti'      => \&impianti,
  'contatti'      => \&contatti,
  'corsi'         => \&corsi,
  'prenotazioni'  => \&prenotazioni,
  'registrazione' => \&registrazione,
  'personale'     => \&personale,
  'prenota'       => \&prenota,
  'edit_personal' => \&edit_personal
);

if( grep { $page eq $_} keys %routes){
  $routes{$page}->();
}
```

Funzione `corsi` associata alla route corsi:

```
sub corsi {
  my @loop_prices = $utils->list_prices;
  my @loop_scheduling = $utils->list_scheduling;
  my %params = (
title => 'Centro sportivo - Corsi',
page      => 'corsi',
path      => 'Corsi',
courses_price => \@loop_prices,
courses_scheduling => \@loop_scheduling,
LOGIN     => $sess_params{is_logged},
USER      => $sess_params{profile},
attempt   => $sess_params{attempt}
```

```
);
    $utils->dispatcher('corsi', %params);
  }
```

Funzione `dispatcher` all'interno di `UTILS.pm`, avendo per convenzione `$route` il nome del template a cui la route è associata, esso viene richiamato e popolato con i parametri contenuti in `%params` settati nella funzione `corsi` in `load.cgi`:

```
  sub dispatcher {
    my $self = shift;
    my $route = shift;
    my %params = @_;
    my $template = HTML::Template->new(filename => $route.".tmpl", utf8 => 1);
    foreach(keys %params){
       $template->param($_ => $params{$_});
    }
    my @loop_news = $self->getNews;
    foreach(@loop_news){
       delete $_->{N_ID};
    }
    $template->param(NEWS => @loop_news);
    print "Content-Type: text/html\n\n", $template->output;
  }
```