FormFactory
Template based web forms

Wolfgang Friebel
Motivation

- CGI scripts are often used to collect information and send it by email or store the data in a database
- Frequently written by inexperienced users or copied from Internet archives
- Usually not thoroughly checked for security holes
- Scripts tend to remain unmaintained, bugs stay in the code as usage frequency is usually low
- Need for well maintained code
- Need to help inexperienced CGI writers
A real world example

- At DESY a malfunctioning web form was reported by users in July this year
- Bug found by K. Woller and corrected in the form
- An analysis has shown that this code was derived from “Matts Script Archive” and used in many places at DESY
- Essential parts of the code are basically unchanged since 1997 when the script was downloaded initially
- The script has still the bug and the site claims “Downloaded over 2 million times since 1997” ...
Available software

- Commercial web editors
  - can create web forms
  - but little support for postprocessing input
  - resulting script is monolithic, may rely on proprietary libraries, databases etc.
- Searching the Internet for solutions
  - overwhelming number of hits
  - many very simple scripts and commercial offers
  - no really good tool found
- Perl as a toolbox with thousands of modules
  - CGI.pm is de facto standard
Design principles

- Separation of program logic, data and layout as much as possible
- known as MVC (model, view, controller) design
- Use as much existing code as possible
- keep solution simple and modular
- Simple forms should not require code to be written
- Allow for complicated tasks
Selection of tools: Program flow

- **CGI** as the underlying base class
- **CGI::Application** to split the task (CGI processing states), many plugins available
  - **AnyTemplate** plugin to be able using templates
    - could also use a specific plugin for a given templating system
  - **ValidateRM** plugin to enable form validation
    - is calling **Data::FormValidator**
Selection of tools: Data model

- describe the data using attribute hashes (name, type, ...)

\[
payment \Rightarrow \{ \text{label} \Rightarrow \text{'Payment method'}, \\
                  \text{field type} \Rightarrow \text{'select'}, \\
                  \text{value} \Rightarrow \text{[ 'cash', 'visa' ]}, \\
                  \text{labels} \Rightarrow \{ \text{cash} \Rightarrow \text{'Cash'}, \\
                                      \text{visa} \Rightarrow \text{'Visa Card'} \}, \\
                  \text{dbtype} \Rightarrow \text{'varchar(5)'} \}, \ldots
\]

- **DBI** module for storing the data in a broad range of available databases

- **Ima::DBI** for lazy loading and SQL encapsulation

- **Net::SMTP** for data transport by email
Selection of tools: Layout

- Use of cascading style sheets (CSS)
  - provide CSS sample files, as support for a wide range of browsers is tricky (Netscape4!)
- **Template::Toolkit** to write the HTML code
  - advantage: access to all data visible within CGI, even access to database
  - used in large projects (sympa mailing list manager)
  - other templating systems possible(*HTML::*Template, *Petal*, ...)
Template toolkit sample code

The following data have been entered by you:

```
[ % USE CGI %]
[ % params = CGI.param() %]
[ % FOREACH par = params %]
  [ % NEXT IF par == 'submit' %]
  [ % NEXT IF NOT form.item.$par %]
  [ % NEXT IF form.item.$par.fieldtype == 'hidden' %]
  [ % IF form.item.$par.label %]
    [ % label = form.item.$par.label %]
  [ % ELSE %]
    [ % label = par %]
  [ % END %]
[ % END %]
[ % label %] = [ % CGI.param(par) %]
[ % END %]
```
Putting it all together: FormFactory

- work in progress (perl module)
- **FormFactory** integrates above mentioned pieces
- reads and parses a config file
- implements the workflow
  - form presentation
  - form validation
  - results presentation (in Web browser)
  - results postprocessing (store in DB, email)
- about 500 lines of code (without doc)
- directly calling perl modules containing about 30 000 LOC
The config file

[general]
  title = Notebooks at DESY
  style = /computing/style2.css
  vardefs = notebook.def

[mail]
  mailto = wolfgang.friebel@desy.de, $email
  mailfrom = uco-zeuthen@desy.de
  mailsubject = notebook mac address registration for $name

[db]
  dbname = mysql:test
  dbuser = wf
  dbpass = yyy
  dbtable = macs

[form]
  ...

Oct 10, 2005
Form definition

[form]

next_runmode = process_template(results)

title = Registration Form

# mandatory fields do start with an asterix

*name
*firstname
*email
phone
hostname

# additional text that should appear in the form starts with a colon

: Enter the MAC addresses in the form xx:xx:xx:xx:xx:xx for one or two interfaces

ethernet
wlan

<submit>
Template processing

- not yet available (for the moment a fixed schema is used)

```python
[results]
    next_runmode = process_template(mailresults)
    (results)
[mailresults]
    # the parameters from the [mail] section should be here
    next_runmode = update_db(storeresults)
    (letter)
[storeresults]
    # the parameters from the [db] section should be here
    dbtable = macs
```
The form in action

- http://www-zeuthen.desy.de/computing-bin/formfactory.pl?conf=notebook
Summary

- perl offers a huge amount of high quality modules
  - designing powerful applications gets easier
  - can be difficult to find the right modules from the almost 9000 ones offered
  - some of the modules used here are fairly new or got popular only recently
- FormFactory used up to now for
  - workshop registration
  - MAC address registration of notebooks
- Different people responsible for parts of the task
  - DB, CSS, web content, processing of data
Outlook

- to do: better definition of workflow in the config file
- Ease access to database using `Class::DBI`
- Do support more form elements
- Provide more templates for common tasks
- Try to use **Catalyst**, a very powerful MVC framework in perl (catalyst.perl.org)
  - even less code to write
  - much more integrated
  - rapidly evolving