WEB DEPLOYMENT OF UNIT CONTENT

By
Michael Kovacs
mkovacs@mcs.une.edu.au

A thesis submitted for the degree of
Bachelor of Computing Science (Honours)
of The University of New England
August 2004
DECLARATION

I certify that the substance of this thesis has not already been submitted for any degree and is not currently being submitted for any other degree.

I certify that to the best of my knowledge, any help received in preparing this thesis, and all sources used, have been acknowledged in this thesis.
Acknowledgements

I would like to thank Dr Ian A. Mason for his support and patience as my honours supervisor.

I would also like to thank Mr Norman Gaywood, Mr Brett Carson and Dr Junbin Gao for their help and advice on many occasions.

Last but not least, I thank my family and friends Nami Matsuo, John Owen, Luke Audsley, Ben Abiad and David Northcott for their support and friendship during my studies.
# Contents

<table>
<thead>
<tr>
<th>Acknowledgements</th>
<th>iii</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2 Current Changes</td>
<td>2</td>
</tr>
<tr>
<td>2.1 General</td>
<td>2</td>
</tr>
<tr>
<td>2.2 Installation</td>
<td>2</td>
</tr>
<tr>
<td>2.3 Index</td>
<td>3</td>
</tr>
<tr>
<td>2.4 Search</td>
<td>4</td>
</tr>
<tr>
<td>2.5 Listings</td>
<td>5</td>
</tr>
<tr>
<td>2.6 On-line Documentation</td>
<td>8</td>
</tr>
<tr>
<td>2.7 Board</td>
<td>15</td>
</tr>
<tr>
<td>2.8 Marks</td>
<td>16</td>
</tr>
<tr>
<td>2.9 Access Control</td>
<td>17</td>
</tr>
<tr>
<td>3 Foreseen Changes</td>
<td>20</td>
</tr>
<tr>
<td>3.1 XHTML and CSS</td>
<td>20</td>
</tr>
<tr>
<td>3.2 Installation</td>
<td>21</td>
</tr>
<tr>
<td>3.3 Search</td>
<td>22</td>
</tr>
<tr>
<td>3.4 Board</td>
<td>22</td>
</tr>
<tr>
<td>3.5 Marks</td>
<td>22</td>
</tr>
<tr>
<td>3.6 Access Control</td>
<td>23</td>
</tr>
</tbody>
</table>
3.7 Technology ...................................................... 23
3.8 Analysis .......................................................... 24
3.9 Conclusion ......................................................... 24

4 Documentation .................................................. 26
  4.1 Installation .................................................... 26
  4.2 Index ............................................................ 26
  4.3 Listings .......................................................... 27
  4.4 Board ............................................................. 29
  4.5 Marks ............................................................. 29

5 How To ............................................................. 32
  5.1 Listings .......................................................... 32
    5.1.1 Unit based customization ................................. 32
    5.1.2 Local customization .................................... 34
  5.2 Search .......................................................... 36
    5.2.1 Indexer .................................................... 36
    5.2.2 Front End ................................................ 38
  5.3 Board ............................................................ 39
  5.4 Marks ............................................................. 40
    5.4.1 Marking .................................................... 40
    5.4.2 The ut-install script ................................. 40
    5.4.3 Access Control ....................................... 48
  5.5 Sysadmin Notes ............................................... 51
List of Tables
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>The home page of typical unit, using the unit template.</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>The unit template search feature.</td>
<td>5</td>
</tr>
<tr>
<td>2.3</td>
<td>The unit template search feature with keyword highlighting.</td>
<td>6</td>
</tr>
<tr>
<td>2.4</td>
<td>The <code>seriesDirListing</code> for a sample unit.</td>
<td>9</td>
</tr>
<tr>
<td>2.5</td>
<td>The <code>seriesUnitListing</code> for a sample unit.</td>
<td>10</td>
</tr>
<tr>
<td>2.6</td>
<td>The <code>dirListing</code> for a sample unit.</td>
<td>11</td>
</tr>
<tr>
<td>2.7</td>
<td>The <code>documentListing</code> for a sample unit.</td>
<td>12</td>
</tr>
<tr>
<td>2.8</td>
<td>The <code>fileListing</code> for a sample unit.</td>
<td>13</td>
</tr>
<tr>
<td>2.9</td>
<td>The <code>seriesListing</code> for a sample unit.</td>
<td>14</td>
</tr>
<tr>
<td>2.10</td>
<td>The <code>simpleListing</code> for a sample unit.</td>
<td>16</td>
</tr>
<tr>
<td>2.11</td>
<td>The <em>Psunami</em> bulletin board.</td>
<td>17</td>
</tr>
<tr>
<td>2.12</td>
<td>The <code>marks</code> page, showing a students marks.</td>
<td>18</td>
</tr>
<tr>
<td>2.13</td>
<td>The <code>stats</code> page, showing the statistics for a particular assessment.</td>
<td>18</td>
</tr>
<tr>
<td>2.14</td>
<td>The <code>class</code> page, showing unit coordinators class results.</td>
<td>19</td>
</tr>
<tr>
<td>4.1</td>
<td>The <code>index.xml</code> file, used to configure the home page of the unit.</td>
<td>28</td>
</tr>
<tr>
<td>5.1</td>
<td>The <code>ut.cfg</code> file used to configure the global look of the unit template.</td>
<td>33</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The Unit Template is a toolkit designed to ease the development and everyday maintenance of on-line units. Several changes have been made to the original Unit Template [11, 12] created by Brett Carson. Not only has it changed in terms of content, but also stylistically, many of its features have been modified and redesigned to add functionality, robustness and the ability to be customizable. This paper will highlight the changes made to each section of the template, as well as provide a brief prospectus of future development.
Chapter 2

Current Changes

2.1 General

The overall design of the template has changed significantly, centralization being the main theme. All standalone CGI [1] scripts have been rewritten into Perl modules, found in the `~wwwdev/UT_0.2/` directory. All command line scripts are now located in a centralised directory, `~wwwdev/UT_Version_0.2/Bin/`. This centralization allows the template to be amended transparently, and simultaneously provide a friendlier object orientated interface to the end user. Centralization extends to the unit account. CGI scripts and configuration files within a unit account are situated within the `~/UT` or the `~/public_html/UT` directory.

2.2 Installation

For the Unit Template to obtain a wide user acceptance, the creation of new units as well as the installation of components (e.g. Listings) must be effortless. To address these needs two scripts have been developed:

- `ut-new`
The `ut-new` script is used to build a new unit from scratch. The new unit comprises of a home page containing links to generic sections (e.g. Lectures, Assignments, UnitDescription), generic sections comprise of empty directories containing a Web page (i.e. a particular Listing type) indicating their presence. On the other-hand, the `ut-install` script is used to install individual components, for example, a particular Listing into a directory, the unit home page, the marks component or the bulletin board.

The installation script plays a major role in the theme of centralization, its use facilitates the enforcement of:

- Ensuring the latest version of a particular component or module is installed.
- All component and module dependencies are automatically met.
- Reasonable defaults are set upon installation.
- File and directory permissions are set appropriately.
- Security (e.g. `.htaccess`) is configured suitably.

## 2.3 Index

The `Index` module is used to dynamically generate the default home page of the unit account. Customizations include the ability to:

- Adjust the unit name and unit code.
- Set new announcements for the unit.
- Enable or disable the unit account search feature.
Adjust the links and images on the unit account home page.

The ability exists for the unit account search feature to log all supplied keywords. This ability may be used to identify topics requiring further attention in the near future.

The Index listing of a typical unit is shown in figure 2.3

![Typical Unit Index Listing](image)

Figure 2.1: The home page of typical unit, using the unit template.

2.4 Search

The Web based search feature may be used to search a particular unit account’s content for specified keywords. This ability serves as a great benefit in identifying lectures tutorials and examples containing specified keywords. It eliminates the need to browse through numerous pages to find the page one is looking for. Advanced features include, the ability to search based on regular expressions and the ability to highlight specified keywords in established pages.
A example search page returned after searching for a specified keyword:

A sample page showing the highlight feature of the search utility, with the specified keyword being highlighted:

### 2.5 Listings

*Listings* refer to a set of CGI scripts, each script designed to deploy the content of a Web based directory structured in a specific way via the Web. Listings give the ability to deploy Web content (e.g. lectures, tutorials) with ease. Dynamically generated Web pages with headers, footers and hyper links to content material can be created with the simple inclusion of the correct listing in a directory. Listings allow content developers to concentrate on what they do best, the development of content. Thus, eliminating the need for content developers to diversify into the field of Web development.

In previous versions of the unit template, listings consisted of stand alone CGI scripts. Each CGI script consisted of numerous lines of embedded source
code, making amendments a tedious process.

To generate the dynamic web-pages which present the content of a directory, each CGI script had to be copied into the desired unit account Web directory. This dispersion of CGI scripts with embedded source code provided a headache in terms of maintainability.

Version 0.2 of the template addresses the problems of maintainability by the use of Perl modules. All listings have moved to a Perl module, named UT::Listing.

The use of modules has removed the need for each CGI script to contain embedded source code. Each standalone CGI script requires code only which makes reference to required modules and its methods.

With the use of Perl modules amendments now may be made in one central location, within the module, eliminating the need to make changes amongst widely dispersed CGI scripts.

Customization is another area addressed. Unit based global headers and footers may be created by placing files named header.tmpl and footer.tmpl
into the ~/UT/Tmpl directory.

Directory based headers and footers may also be applied, by including the header and footer files in the directory of the particular listing.

Arbitrary HTML can now be placed just below the title of each listing page via the creation of a file named notice.tmpl containing arbitrary HTML markup, this is ideal for entities like notices.

The use of a unit based global configuration file named ~/UT/Cfg/ut.cfg has been introduced. The unit based global configuration file allows the configuration of font-size, font-color, background-color and open/closed image icons settings throughout a unit. These same changes may be made locally, per listing, by setting the appropriate arguments to each listing. For example,

```perl
$ut->seriesListing(
    font_color => 'black',
    font_size => 5,
    background_color => 'pink',
    open_img => '/~wwwdev/Img/greendiamond.gif',
    closed_img => '/~wwwdev/Img/reddiamond.gif',
);
```

would set the font-color to black, font size to five and the background color of the page to pink and the open and closed image indicators to greendiamond.gif and reddiamond.gif respectively. Styles may also be applied via CSS constructs in the ~/public_html/UT/Css/ut.css file.

Further features include the ability to apply filters to listings. Filters are used to hide files and, or directories with particular characteristics from being displayed by a particular listing. For example, a filter might be used to stop fileListing from displaying all makefiles in a directory, or to display only PDF files.
2.6 On-line Documentation

It is generally agreed that software offering functionality is great, and, by this, the more functionality the better. However, a lack of documentation can seriously undermine the efficiencies of improved functionality.

Documentation can now be found by typing man ut at the command prompt, this documentation is a starting point which gives directions on how and where to obtain further help relating to the Unit Template. A Website, providing documentation, news, bulletin board can be found at http://mcs.une.edu.au/~wwwdev.

Addressing the needs of a variety of users, was another issue which needed to be addressed. Current implementation of the template focused mainly on facilitating users who publish content using the LaTeX [13] typesetting system, and obeying certain file naming conventions. To add flexibility three new listings have been developed:

- seriesDirListing
- seriesUnitListing
- seriesRegexListing

With the addition of the new listing, there are now eight types of listings available:

- seriesDirListing
- seriesUnitListing
- seriesRegexListing
- dirListing
- documentListing
CHAPTER 2. CURRENT CHANGES

- fileListing
- seriesListing
- simpleListing

A screen capture of each of the listings can be seen below.

The **seriesDirListing** listing of a sample unit looks like:

![Figure 2.4: The seriesDirListing for a sample unit.](image)

Use **seriesDirListing** when a directory contains a series of documents in one or more formats. For example a directory may contain the following files:

index.cgi
Lecture01.ppt
Lecture02.ppt
Lecture03.ppt
Lecture04.ppt
The \texttt{seriesUnitListing} listing of a dummy unit looks like: Use \texttt{seriesUnitListing}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig2-5.png}
\caption{The \texttt{seriesUnitListing} for a sample unit.}
\end{figure}

when a particular unit has differing content for each level at with its offered. For example, a unit may have different lectures for its 100 level and 500 level students. For example:

\begin{verbatim}
index.cgi
lecture100.tex
lecture100.ps
lecture100.pdf
lecture100
lecture500.tex
lecture500.ps
lecture500.pdf
lecture500
\end{verbatim}

The \texttt{dirListing} listing of a sample unit looks like:
Figure 2.6: The dirListing for a sample unit.

Use dirListing to provide access to the contents of directories, ideal in situations in which a directory contains arbitrary file types. Note, it is also possible to place other listing types (e.g. fileListing) into each of the directories referred to by dirListing. For example a directory may contain the following subdirectories:

```
index.cgi
Assignment_01
Assignment_02
Assignment_03
Assignment_04
```

also, the Assignment_01 subdirectory may contain the fileListing listing, indicated by the index.cgi script:

```
index.cgi
Q1.pl
Q2.pl
```
Q3.pl
Q4.pl

The documentListing listing of a sample unit looks like:

![Document Listing Example](image)

Figure 2.7: The documentListing for a sample unit.

Use documentListing when a directory contains a series of documents generated by latex and the latex2html UNIX utility. For example a directory may contain the following files with the unit-description subdirectory containing the HTML version of the document generated by the latex2html UNIX utility:

- index.cgi
- unit-description.tex
- unit-description.ps
- unit-description.pdf
- unit-description

The fileListing listing of a sample unit looks like:
Use `fileListing` when a directory contains a list of files, for example unit software for downloading. For example:

```
index.cgi
perl-5.8.0-55.i386.rpm
```

The `seriesListing` listing of a sample unit looks like:

Use `seriesListing` when you have a series of directories containing documents generated by `latex` and the `latex2html` UNIX utility. For example:

```
index.cgi
Lecture_01
Lecture_02
Lecture_03
Lecture_04
```

also, the `Lecture_01` subdirectory may contain the following files, where `lecture_01` is a subdirectory containing the HTML version of the document generated by the `latex2html` UNIX utility:
CHAPTER 2. CURRENT CHANGES

Figure 2.9: The `seriesListing` for a sample unit.

```
lecture_01.tex
lecture_01.ps
lecture_01.pdf
lecture_01
```

*Note:* by convention the `.tex` document within the sub directories should be named with the same name as the containing directory but in all lowercase and with the `.tex` extension. For example, the directory `Lecture_01` should contain a LaTeX file by the name of `lecture_01.tex`, if you would like to avoid this convention use `seriesRegexListing`. Or it will also work with the trailing digits removed. For example, with `Lecture_01` directory containing the LaTeX file by the name of `lecture.tex`

`seriesRegexListing` is similar to `seriesListing` but allows you to specify a regular expression to identify the document for the directories. For example:

```
index.cgi
```
Lecture_01
Lecture_02
Lecture_03
Lecture_04

Lecture_01 might contain:

n-1.tex
n-1.ps
n-1.pdf
n-1

while Lecture_02 might contain:

n-2.tex
n-2.ps
n-2.pdf
n-2

The \texttt{simpleListing} listing of a sample unit looks like:

Use \texttt{simpleListing} when you want to display straight HTML markup. The HTML markup is added to a file by the name of index.tmpl, in which the special tag \texttt{<tmpl var name=\textquoteleft image\textquoteright>\textgreater} may be used as a place holder for the open close image icon.

\section*{2.7 Board}

A bulletin board is a necessary part of any unit, facilitating communication between student and staff. The unit template makes use of \textit{Psunami Bulletin Board} \cite{5}, with a few tweaks and customizations, mostly related to supporting its out of the box installation.

The \texttt{board}, shown below:
2.8 Marks

On-line units require a system of providing assessment results to students, either by making the marks available via the Web or by notifying the students via email. The marks module facilitates both of the methods, as well as allowing students to view statistics numerically or visually (via a pie graph) via the Web. Unit co-ordinators also have the option of browsing class results on-line in a tabular format. There exists two scripts to facilitate the marking process:

- **ut-markingScheme**

- **ut-marked**

  The **ut-markingScheme** utility is used to assign a marking criteria file to each student. The utility also has the ability to fill in template variables such as login name, study mode and time of submission within the criteria file.
The maintenance of the results database can be a tedious process for staff. To solve the tediousness of maintaining the marks database an utility named ut-marked found in the ~wwwdev/UT_Version_0.2/Bin has been introduced.

The ut-marked utility has the ability to iterate through a list of files containing the filled out marking criteria for each student, extracting each students final mark and inserting them into the database.

The marks page of a sample unit looks like:

The stats page of a sample unit looks like:

The class page, showing the marks obtained by individual students:

2.9 Access Control

The ut-dir utility has the ability to open up directories to certain groups defined in the /UT/ApacheAuth/htgroup file. This is ideal for situations in which for example the solutions to a certain assignment has to be restricted on a UNE campus bases.
CHAPTER 2. CURRENT CHANGES

Figure 2.12: The *marks* page, showing a student's marks.

Figure 2.13: The *stats* page, showing the statistics for a particular assessment.
Figure 2.14: The *class* page, showing unit coordinators class results.
Chapter 3

Foreseen Changes

3.1 XHTML and CSS

All Web based components of the template should incorporate the use of XHTML and CSS. Extensible HyperText Markup Language [6] (XHTML), is the new markup language of the Web, it is much stricter than HTML, and is designed with general user agent interoperability in mind. Cascading Style Sheets [10] CSS, is a simple way to add style and layout to a selected group of Web documents.

The incorporation of XHTML and CSS technology will ensure that all Web pages are accessible from a wide variety of browsers and devices. Separation of content and presentation will be achieved, eliminating the need to generate presentation components dynamically. This separation of content and presentation will allow the maximal customization of layout and appearance of all components, eliminating the need for example, the unit based configuration file, ut.cfg used to set page attributes unit wide. Printer friendly versions of pages can be created (e.g. class marks) eliminating the need to cut, paste and reformat data before printing.
3.2 Installation

Integration of the install utility with the Index module. For example, when a particular listing is installed in a units Web based directory, the 

"~/UT/Index/index.xml" file could be modified in some way to reflect the change. This feature would eliminate the need to add an entry to the index.xml each time a particular listing is created. On the other hand, each listing upon installation might need to be invoked with additional argument(s) indicating the category to which the particular listing should be added, as well as the name of the hyperlink to be created on the front page. For example, the creation of a listing for Lecture Notes might require the following:

```
    ut-install --index --title "Teaching Material" --link "Lecture Notes"
```

which in turn would make the following addition to the index.xml file:

```
    <table>
    <title>Teaching Material</title>
    <image>~/comp315/Images/perlGodSmall.jpg</image>
      .
      .
    <link text="Lecture Notes" href="/comp315/Lectures/" />
      .
      .
    </table>
```

where `<link text="Lecture Notes" href="/comp315/Lectures/"/>
`, indicating the addition made to the index.xml file. The --link parameter may be optional, as its argument may be derived from the installation directory name, for example the link Lecture Notes may be derived from the directory named LectureNotes.
CHAPTER 3. FORESEEN CHANGES

3.3 Search

The Search utility should provide the ability to search a variety of file types (by mime-type, **File::MMagic**) other than HTML and text. File types such as MS Word, PDF, PPT should also be able to be searched, if functionality to a wide variety of users is to be attained. Additional features such as ranking, spell checking should be considered, a good starting point would be *Panoptic* [4], the search engine, used at the UNE.

3.4 Board

The only way to truly centralize the bulletin board is to develop one from scratch. Writing a bulletin board from scratch will allow the addition of features such as:

- Search posts by user-name, date, category and keyword.

- Code pasting ability, eliminate the inclusion of source code in a post, rather store the code on the file-system providing a hyper link to it.

- Enable code formating and highlighting.

The use of a database to build the bulletin board would make most of the features trivial to incorporate.

3.5 Marks

Ideally the results of students should be stored in a real database. The use of a real database would make statistics easier to compute and the source code much cleaner, the use of a database would also eliminate hacks used to bypass the shortcomings of using CSV files. The use of a database would not be
security issue as the database is used only as intermediate store facilitating students access to their marks and statistics, students marks would still in turn be located on the file-system, i.e. within the marking criteria files.

Some robustness also needs to be inbuilt, for example each marking related script (e.g. `ut-marked -a Assignment_01`) should check that it is passed a correct assessment as an argument.

3.6 Access Control

Access control to "solutions" is achieved with the `ut-dir` script, which allows content to be open to individuals on UNE campus basis, via the use of a Apache handler. It has become to sight, that this functionality may be possible to achieve with Unix groups and with standard Apache authentication.

3.7 Technology

- Examine the advantage and disadvantage between use of CGI and `mod_perl, FastCGI, JSP` and similar technologies.

- Examine the advantage and disadvantage between the use of a database (`PostgreSQL, SQLite`) and plain files.

- Examine the benefits caching (e.g. `CGI::Cache`).

- The use of version management (e.g. `CVS`) software, the ability to recover to "functional versions" will become critical as the user base of the Unit Template increases
3.8 Analysis

A greater analysis of the subject matter has to take place. Needs and expectations of students and staff has to be looked at in depth. Prominent Course software on the market

- WebCT [7]
- Blackboard [9]
- Moodle [3]

has to be evaluated, compared and contrasted. The setting up of a Web site for feedback, comments, news and updates would be critical for future development of the software.

3.9 Conclusion

In conclusion, the new version of the unit template has made amendments to certain characteristic of previous versions, it also added new and useful features, for both unit-coordinators as well as students. This and previous related work has provided an insight into what can be achieved with regards to simplifying the creation and every day maintenance of a on-line unit with freely available tools. Currently, there are over thirteen units using the unit template at UNE, some of which include:

- comp132, Computer Science 132
- amth140, Discrete Mathematics
- amth142, Applied Mathematics 142
- comp285, Intro 2 OOP in Java
Insight into areas needing further investigations has been provided for future developers. For development to continue and succeed, it is vital for users i.e. unit-coordinators as well as students to participate in its development and bug reporting, tracking. Due to the necessity of user involvement a Web site has been set up at http://mcs.une.edu.au/~wwdev at which, comments, bug fixes can be made.
Chapter 4

Documentation

4.1 Installation

Installation of the unit template modules is accomplished with the help of the utility script named ut-install located in the `~wwwdev/UT_Version_0.2/Bin` directory. This section will take a closer look at the ut-install utility and how it facilitates the installation of the unit template modules.

4.2 Index

The home page of the unit is constructed by calling the ut-install utility as follows,

```
    ut-install --index
```

from within any directory of the unit account. The ut-install utility performs the following steps on the installation of the unit home page:

- The creation of the local `/public_html/index.cgi` script, corresponding to the front page of the unit.
The creation of the front page configuration file, ~/.UT/Index/index.xml.

The creation of unit based headers and footers, ~/.UT/Tmpl/header.tmpl and ~/.UT/Tmpl/footer.tmpl.

The creation of the search utility configuration file, ~/.UT/Search/search.cfg, necessary if the search feature is activated via the index.xml configuration file.

Customization of the unit home page can be achieved via the modification of the front page configuration file, ~/.UT/Index/index.xml, the configuration options include:

- The ability to change the unit name and code to an arbitrary value.
- The ability to place student notices on the home page.
- Enabling or disabling the units search feature.
- Enable or disable search feature logging, search strings entered may be logged for further analysis.
- The ability to create arbitrary categories, containing a title, image and various links to materials.

The index.xml file, shown below, attribute values are shown in red and black.

4.3 Listings

Once content (e.g. lectures, assignments) have been uploaded to their designated directory, the type of listing required has to be decided, see Current
Figure 4.1: The index.xml file, used to configure the home page of the unit.

Changes 2.5 for a detailed description of the different types of listings available.

Listing pages are constructed by calling the ut-install utility as follows,

```
ut-install --listing dirListing
```

in the desired directory, where dirListing, may be replaced with any other of the six listing types. The ut-install utility performs the following steps on the installation of a listing:

- The creation of the local index.cgi script, corresponding to the particular listing type.
- The creation of the listing configuration file, ~/UT/Cfg/ut.cfg, where attributes values, font-size, font-color, background-color and open/closed image icons may be set.
- The creation of unit based headers and footers, ~/UT/Tmpl/header.tmpl and ~/UT/Tmpl/footer.tmpl.
4.4 Board

The bulletin board for the unit is constructed by calling the \texttt{ut-install} utility as follows,

\begin{verbatim}
  ut-install --board
\end{verbatim}

from within any directory of the unit. The \texttt{ut-install} utility performs the following steps on the installation of the bulletin board:

- The creation of the \texttt{~/BulletinBoard} directory, containing bulletin board associated files.

- Creation of the default threads named, \textit{General discussion}, \textit{Exam}, \textit{Bouncing Mail} and \textit{Web page \etc}. As well as the creation of a thread per assessment, as indicated by the \texttt{project} utility. Manual administration of the bulletin board can be performed via the Web at \url{http://mcs.une.edu.au/~compXXX/BulletinBoard/admin.cgi}, where \texttt{compXXX} is the name of the unit.

- The creation of the \texttt{admin} group in the \texttt{~/UT/ApacheAuth/htgroup} file, used to restrict access to the administration feature of the bulletin board to those people contained in the local \texttt{.rhosts} file.

4.5 Marks

Before the marks module is set up it is recommended that the unit is first set up to allow for the submission of assignments, whether it be via the UNIX shell or the Web. A unit is set up ready to accept submissions with the \texttt{project} program located on \texttt{turing}. \texttt{root} access may be required for this, see your system administrator for help. The \texttt{project} program creates a file named \texttt{Projlist} located in either the units \texttt{~/submit}, or \texttt{~/Submit}
directory, which contains a list of assessments available for the particular unit, which is used by the install utility. An example Projlist file:

```
+Assignment_01
+Assignment_02
+Assignment_03
+Assignment_04
+Assignment_05
+Project_01
=Project_02
```

To set up the marks module for the unit, the `ut-install` utility is called as follows,

```
ut-install --marks
```

from within any directory of the unit. The installation script performs the following steps:

- Create the directory `~/UT/` for later use.

- Create the directory `~/public_html/Marks`, this is the directory from which students will be able to access their marks via the Web. The directory is populated with the marks `index.cgi` script and a `.htaccess` file used for authentication.

- The user is prompted for the maximum obtainable mark for each assessment.

- The file `assessment.csv` is created in the `~/public_html/Marks` directory, representing each assessment and the maximum obtainable mark for the particular assessment.
The headers for the file `marks.csv` file is created in the `~/public_html/Marks` directory, representing each user, assessment, and the mark obtained for each assessment.

The directory `~/Marked` is created, with subdirectories as per the `~/submit` or `~/Submit` directory. This directory is where the marking criteria for each student shall be placed.

The creation of unit based headers and footers, `~/UT/Tmpl/header.tmpl` and `~/UT/Tmpl/footer.tmpl`.

After the installation you should be able to access the unit marks page with the appropriate URL `http://mcs.une.edu.au/ compXXX/Marks/`. 
Chapter 5

How To

5.1 Listings

Customization of the listings module can be broken down into two sections:

5.1.1 Unit based customization

Unit based customization is achieved by the modification of the unit based configuration file `~/UT/Cfg/ut.cfg` and the unit based header and footer template files `~/UT/Tmpl/header.tmpl` and `~/UT/Tmpl/footer.tmpl`.

Some of the attributes which may be set unit wide via the `ut.cfg` file include:

- **open_img** and **closed_img**, allows the setting of the visual indicator (i.e. image), indicating accessibility of a particular file or directory. The visual indicator, image, has to be a valid URL.

- **bodybgcolor**, used to set the background color property of each listing page.

- **font_color**, used to set the font color property of each linking page.
• **font_size**, used to set the font size property of each linking page.

The **bodybgcolor** and **font_color** property may be expressed in either of two ways: as the red, green and blue (RGB) components of the desired color or as a standard color name. For example, using the standard color name:

```plaintext
bodybgcolor => 'lightblue',
```

or RGB

```plaintext
bodybgcolor => '#ADD8E6',
```

The **font_size** attribute must be one of the virtual sizes (1-7), defined as an absolute size or preceded by a plus or minus sign (+ or -) to define a relative font size.

The **ut.cfg** file, shown below, attribute values are shown in red.

```plaintext
# ut.cfg
open_tag = "<!--
home_page = "http://www.example.com"
bodybgcolor = "lightblue"
font_color = "black"
font_size = 5
-->

Figure 5.1: The **ut.cfg** file used to configure the global look of the unit template

**simpleListing** deserves a special mention, as its intent defers to that of other listings. Its purpose, to displays the content of the file named **index.tmpl**, located in the same directory as the listing itself. The **index.tmpl**
file may contain any valid HTML constructs as well as the special `<tmpl_var name="image">` tag, used as a place holder for the current open / closed visual indicator.

### 5.1.2 Local customization

Each listing script (i.e. `index.cgi`) may be customized on an individual basis. All attributes which may be set globally in the `ut.cfg` file may also be set on an individual bases, for example,

```perl
$ut->seriesListing(
    open_img => '/~wwwdev/Img/greendiamond.gif',
    closed_img => '/~wwwdev/Img/reddiamond.gif',
    body_bgcolor => 'lightblue',
    font_color => 'black',
    font_size => 5,
);
```

will set the visual indicator to the image identified by the given URL, used to indicate whether a file or directory is accessible or not. Also, the background color, font color and font size will be set locally (i.e. per directory).

Additionally, settings exists which may only be set locally, these settings include:

- The title method, used to specify the given title for the particular page, for example:

  ```perl
  $ut->title("Arbitrary Directories");
  $ut->dirListing();
  ```

will set the page title to *Arbitrary Directories*, instead of the default, that of the directory name in combination with the unit name.
• The file formats to be displayed and order may be specified by the `show_extension` attribute, in both `documentListing` and `seriesDirListing`.

```perl
$ut->documentListing(
    show_extension => [ qw( pdf ps dvi html ) ]
);
```

will produce HTML links to documents of type `pdf`, `ps`, `dvi` and `html` in that particular order.

• When using `seriesListing` or `dirListing`, there sometimes arises the need to ignore miscellaneous sub-directories (e.g. sub-directories generated by the `latex2html` utility). The `ignore_name` attribute may be used to ignore these unwanted directories,

```perl
$ut->seriesListing(
    ignore_name => [ qw( Graphics lecture slides Images ) ]
);
```

will ignore all all sub-directories having the name of `Graphics`, `lecture`, `slides` and `Images`.

• The `ignore_regex` attribute can be used to ignore files or directories based on Perl style regular expressions. For example:

```perl
$ut->seriesDirListing(
    show_extension => [ qw( ppt pdf ) ],
    ignore_regex => [ qw( ^lect_01\. ) ],
);
```

will ignore all lecture one documents, i.e. any document with the name `lect_01`, irregardless of the file extension.
$ut->fileListing(
    ignore_regex => [ qw( \.cgi$ makefile ) ],
);

will ignore all CGI files and makefiles (strictly speaking, any file whose name containing the string makefile).

5.2 Search

It consists of two programs:

- the indexer
- front end

5.2.1 Indexer

`ut-index` is the back end indexer for the search utility located on the unit home page. The `ut-index` performs the following functions, once given a root directory to index:

- Locates files to be indexed using the `File::Find` Perl module.
- Files not Web accessible are not indexed, i.e.
  - the file itself is not readable by the Web server
  - directories leading to the file cannot be accessed by `apache`
- Unix paths are converted to URL’s and Web accessibility re-examined
  - if `LWP::Simple::get()` is successful in retrieving the page, the page is considered to be accessible
• use HTML::Parser to grab the text (content) of the HTML page

• store desired information in a Berkeley DB file
  
  - document.db, documents-url \rightarrow document-id
  
  - index.db, word \rightarrow document-id(s)
  
  - intro.db, document-id \rightarrow intro (first 200 chars of file)
  
  - mtime.db, file modification time
  
  - size.db, file size
  
  - title.db, HTML title or filename

• Break the text up into words with the following regular expression:

  \( / ( \texttt{[\\w\{,\} ]} ) \texttt{\ b} /ixg \)

  and index it,

  
  recognized \rightarrow 37,53
  
  powerful \rightarrow 37,53,54
  
  vspace \rightarrow 37
  
  smh \rightarrow 37
  
  uk \rightarrow 37
  
  announcements \rightarrow 42,44,65

The index utility should be run on a regular basis, when content is made available or restricted through the Web. Content can be made available in two ways, either with the UNIX command `chmod` or with the `ut-dir` utility, described below. The `ut-index` utility may be configured via the `~/UT/Search/search.cfg` file. By default all HTML and .tmpl files in the Web root (i.e. `public_html`) will be indexed. Configuration options include:
• **directory**, sets the root directory from which indexing begins, default being `public_html`.

• **extension**, files of the specified extensions will be indexed, default being HTML and `.tmpl`.

• **regex**, Perl regular extension indicating paths to ignore.

Some examples `search.cfg` files:

```plaintext
directory = ~/public_html
extension = html
extension = tmpl
```

is the default configuration file.

Once the `search.cfg` file has been created, the `ut-index` utility may be executed in the following two ways:

• **`ut-index --start --verbose`**, to create index, displaying all files being processed, ideal to see if configuration options have been set correctly.

• **`ut-index --start`**, to index in quiet mode.

• Other useful features include, running the indexer in the background, **`ut-index --start &`**, incorporating it into a **makefile** or even running it as a cron job.

### 5.2.2 Front End

The front end of the search utility is responsible for looking up keywords in the database (Berkeley DB file named `index.db`), and presenting the results to the user. The front end performs the following functions:
CHAPTER 5. HOW TO

• extracts first keyword entered in a HTML text field

• if the word is in the DB file, we have the required document_id(s), hence the desired URLs (by looking in document.db)

• based on the document_id(s) extract other information e.g. title, file size from the relevant Berkeley DB file.

• to highlight the keyword within the found document:
  
  − retrieve the desired URL with LWP::Simple
  
  − parse the document with HTML::Parser to obtain the content
  
  − perform the highlighting by placing the HTML span ah around the keywords in the document, e.g.

    $dtext =~ s/($word)/<span style="color: black; background: #ccffcc">$1</span>/gi;


5.3 Board

The bulletin board installs out of the box, there is no need for any modifications. On the other hand if you would like to customize the boards look and feel, here are some hints:

• Stylistic changes may be made in the psunami.conf file.

• Layout changes can be made by modifying the appropriate files in the psunami/templates/en directory.
5.4 Marks

5.4.1 Marking

To assist in the marking process of assessment material, the following scripts have been created:

- `ut-install`
- `ut-markingScheme`
- `ut-marked`

Another group of scripts exists to help control access to material (e.g. assignment solutions), these scripts include:

- `ut-htgroup`
- `ut-dir`

5.4.2 The `ut-install` script

The `ut-install` script with the `--marks` argument is needed to be run once before any marking takes place to create files and directories later used by the other marking related utilities. The `ut-install` script assumes that the unit has been enabled to allow students to submit assignment, whether it be via the Web or a shell account. For more information on how to enable a unit for assignment submission take a look at the `project` utility on `turing`, you might need to contact your system administrator, as root access might be required.

The `ut-install` script upon invocation, performs the following:

- It prompts for each assessments maximum obtainable mark.
$> ut-install --marks
Assignment_01 is out of > 20
Assignment_02 is out of > 20
Assignment_03 is out of > 20
Assignment_04 is out of > 20
Assignment_05 is out of > 20
Project_01 is out of > 50
Project_02 is out of > 50
$>

- It creates the directory named "~/public_html/Marks" containing the following files:
  - index.cgi, is the CGI script used by students to access their marks via the Web. It may also be used by staff to view students marks via the Web. Its content appears as follows:

    ```perl
    #!/usr/bin/perl

    use lib qw(/homes/admin/wwwdev/UT_0.2);
    use UT::Marks;

    $ut = UT::Marks->new();
    $ut->title();
    $ut->showMarks(
        stats => 1
    );
    ```

    The `stats` argument may be used to enable (by setting it to 1) or disable (by setting it to 0) the displaying of statistical information.

  - assessment.csv, is the database table used to store assessment
name and each assessments maximum obtainable mark. If you wish to change the maximum obtainable mark details after the installation this is the file to modify. Here is an example assessment.csv file:

<table>
<thead>
<tr>
<th>assessment, total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment_01, 20</td>
</tr>
<tr>
<td>Assignment_02, 20</td>
</tr>
<tr>
<td>Assignment_03, 20</td>
</tr>
<tr>
<td>Assignment_04, 20</td>
</tr>
<tr>
<td>Assignment_05, 20</td>
</tr>
<tr>
<td>Project_01, 50</td>
</tr>
<tr>
<td>Project_02, 50</td>
</tr>
</tbody>
</table>

- marks.csv, is the database table used to store students marks. Here is an example assessment.csv file:

<table>
<thead>
<tr>
<th>user, assessment, mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>mkovacs, Assignment_01, 15</td>
</tr>
<tr>
<td>mkovacs, Assignment_02, 17</td>
</tr>
<tr>
<td>mkovacs, Assignment_03, 10</td>
</tr>
<tr>
<td>mkovacs, Assignment_04, 12</td>
</tr>
<tr>
<td>mkovacs, Assignment_05, 18</td>
</tr>
<tr>
<td>mkovacs, Project_01, 45</td>
</tr>
<tr>
<td>mkovacs, Project_01, 43</td>
</tr>
</tbody>
</table>

The marks.csv is populated with the ut-marked script, and should not be manually modified.

- It creates the directory named ~/Marked containing for example the following directories:

  Assignment_01
The `~/Marked` directory is the directory in which each student's marking criteria per assignment will be located and filed in.

Each of the `Assignment_xx` and `Project_xx` directories in the `~/Marked` directory correspond to the equivalent directory in the `submit` directory. For example, if the internal student by the user id of `mkovacs` has submit assignment one, he would have the following file in the `submit` directory:

```
~/submit/Assignment_01/int/mkovacs.Z
```

And for easy reference, once the student's assignment has been marked, the marking criteria should be placed in the equivalent `~/Marked` directory, for example:

```
~/Marked/Assignment_01/int/mkovacs.txt
```

The `MarkingScheme` directory is special, it contains a template criteria file for each of the unit assessments, for example:

```
assignment_01.txt
assignment_02.txt
assignment_03.txt
```
The template criteria for each assessment should be modified according to the criteria for the particular assessment. Each criteria may contain placeholders for the following information:

- `##assessment##`, the particular assessment, e.g. `Assignment_01`.
- `##loginName##`, the login name of the person submitted the assignment, e.g. `mkovacs`.
- `##intExt##`, the mode of the person submitted the assignment, e.g. `int` for internal.
- `##timeStamp##`, the date and time when the submission was made.

Here is an example template criteria:

```
Marking scheme for ##assessment##

login name:     ##loginName##
int/ext:        ##intExt##

late penalty:   (0%)
```
Total.........................................................[/20]

Once the marking criteria has been set, it is necessary to assign a marking criteria file to each student. Assigning the marking criteria file to each student is the task of the `ut-markingScheme` script.

Usage:

`ut-markingScheme [--assessment assessment] [--substitute]`

Once the `ut-markingScheme` script is called, it performs the following tasks for the particular assignment:

- It creates a marking criteria file for each user found in the `submit` directory and places it into the equivalent `~/Marked` directory.

- If it is called with the `--substitute` flag, the `ut-markingScheme` script also replaces the given placeholders in the marking criteria file with their equivalent values.

For example, given the internal user `mkovacs` has submitted assignment one:

```bash
$> ls -la ~/submit/Assignment_01/int/
-rw------- 1 comp315 admin 1422 Mar 18 09:35 mkovacs.Z
```

After running the `ut-markingScheme` script with the `--substitute` flag enabled, the following marking criteria file will be produced:

```bash
$> ut-markingScheme -a Assignment_01 -s
copied marking criteria for mkovacs
$> ls -la ~/Marked/Assignment_01/int/
-rw-r--r-- 1 comp315 admin 255 Mar 29 08:49 mkovacs.txt
```
And the content of the created marking criteria file appears as:

Marking scheme for Assignment_02

login name: mkovacs
int/ext: int
time stamped: Mon Mar 29 08:49:24 2004
late penalty: (0%)

Question 1:
Message in BBoard.................................................[/1]
Image in Gallery....................................................[/1]
env_info_pm.cgi works from browser........................[/2]
Subtotal........................................................[/4]

Question 2:
The "use strict" and "-w" flags are present................[/1]
Correct permissions on file.................................[/1]
Uses "split" correctly........................................[/1]
Simple, e.g. uses scalar & list contexts.................[/1]
Subtotal........................................................[/4]

Total...........................................................[/8]
NOTE: The total line in the marking criteria files have to match one of the following formats:

- total 40
- Total 40/50
- Total .... [40/50]
- Total.......[20.5/50]

otherwise the ut-marked script, described later, will not be able to gather total mark given to each student. Once the marking criteria files have been filed in by the markers, the marks should be made available to the students via the Web, and / or emailed out to them. The ut-marked script is responsible for publishing the marks via the Web and / or emailing the marking criteria files out to students.

The main job of the ut-marked script is to gather the marks for a particular assessment (e.g. Assignment_01) from within the marking criteria files, populate the marks database (i.e. the ~/public_html/Marks/assessment.csv file) and generate the pie graph image in the ~/public_html/Marks/ directory.

Example usage of the ut-marked script:

**Usage:**

```
ut-marked [--help][--delete][-mail] [--assessment assessment] [user(s)]
```

Options include:

- **--assessment**, Specifies which particular assessment we are working on e.g. Assignment_01.
- **--delete**, Used when Web access to marks / statistics for a particular assessment is not desired.
- **--mail**, Mails each of the marking criteria files to student(s).
• **user(s)**, Mails the specified user his / her marking criteria, as well as
  updating the Web marks / statistics. Use this when a change occurs
  in a particular students mark.

  `ut-marked -a Assignment_01`

  Assignment_01 has been marked, now set up Web marks / statistics for access
  to students.

  `ut-marked -a Assignment_01 -m`

  Assignment_01 has been marked, now set up Web marks / statistics for access
  to students and email each student their marking criteria file.

  `ut-marked -a Assignment_01 mkovacs`

  Changed the mark given to mkovacs, update marks / statistics accordingly
  and mail him his new mark (i.e. marking criteria file).

  `ut-marked -a Assignment_01 -d`

  Close Web access to marks / statistics for Assignment_01.

### 5.4.3 Access Control

Access control to content via the Web may be controlled via the use of two
scripts:

• `ut-htgroup`

• `ut-dir`

  The `ut-dir` script is used to restrict access to a directory to a particular
  group.

  Some examples:
CHAPTER 5. HOW TO

Usage:

ut-dir [--help] --close [directory] | --open [directory] [--group groupName]

Options include:

- **--close**, causes a particular file or directory to be closed up for Web access.

- **--open**, causes a particular file or directory to be opened up for Web access. If also given the argument `group` the file or directory will only be opened up for Web access to the given group (see `ut-htgroup` for more information on groups).

`ut-dir -o`

Will open the current directory up for Web access to all.

`ut-dir -c`

Will close the current directory up for Web access to all.

`ut-dir -o -g admin`

Running the above in for example the `~/Solutions/Assignment_05` directory will restrict Web access to the directory to only members of the admin group.

`ut-dir -o -g admin Project_01`

Opens up the `Project_01` directory for only admin access via the Web.

The `ut-htgroup` script is used to create or add list of user groups used by Apache [8] for authentication. The unit template uses the `~/UT/ApacheAut/htgroup` file to store group information.

Example usage of the `ut-htgroup` script:
ut-
tgroup [--help] --list unitCode | --show mailAliasFile | --delete groupName | --group groupName --members mailAliasFile | admin | file

Example usage:

ut-
tgroup -l comp315
Display the mail aliases for the unit comp315.

ut-
tgroup -s comp315e1
Show the members of the comp315e1 mail aliases file.

ut-
tgroup -g comp315 -m comp315e1
Creates the group comp315 consisting of members contained in the comp315e1 mail alias file, if the group does not already exists. If the group comp315 already exists, members from the comp315e1 mail alias file are added to the existing comp315 group.

ut-
tgroup -g comp315 -m comp315i1
Creates the group comp315 consisting of members contained in the comp315i1 mail alias file, if the group does not already exists. If the group comp315 already exists, members from the comp315i1 mail alias file are added to the existing comp315 group.

ut-
tgroup -g comp315 -m comp315e1 -m comp315i1
Combines the last two examples into a single call.

ut-
tgroup -g admin -m admin
CHAPTER 5. HOW TO

Creates the group admin consisting of members contained in the local .rhosts file, if the group does not already exists. If the group admin already exists, users from the local .rhosts file are added to the existing admin group

    ut-htgroup -g comp315 -m ./extraComp315Students

Creates the group comp315 consisting of members contained in the local ./extraComp315Student file, if the group does not already exists. If the group comp315 already exists, members from the ./extraComp315Students are added to the existing comp315 group.

    ut-htgroup -d comp31

Deletes the comp315 group.

5.5 Sysadmin Notes

The ~/wwdev/UT_Version_0.2/ directory is where all the necessary files for the unit template reside, the contents of the directory in a tree-like format:

    UT_Version_0.2/
    |-- Bin
    |   |-- InstallFiles
    |   |     |-- BulletinBoard.zip
    |   |     |-- Marked
    |   |     |     |-- markingScheme.txt
    |   |     |     |-- Marks
    |   |     |     |-- index.cgi
    |   |     |     |-- marks.csv
    |   |     |-- UT
    |   |     |     |-- ApacheAuth
CHAPTER 5. HOW TO

`-- htgroup
|   |-- Cfg
|   |   |-- ut.cfg
|   |   |   |-- Index
|   |   |   |   |-- index.xml
|   |   |   |-- Search
|   |   |   |   |-- search.cfg
|   |   |-- Tmpl
|   |   |   |-- footer.tmpl
|   |   |   |-- header.tmpl
|   |   |-- dirListing.cgi
|   |   |-- documentListing.cgi
|   |   |-- errorDocument401.cgi
|   |   |-- fileListing.cgi
|   |   |-- index.cgi
|   |   |-- seriesDirListing.cgi
|   |   |-- seriesListing.cgi
|   |   |-- seriesRegexListing.cgi
|   |   |-- seriesUnitListing.cgi
|   |   |-- simpleListing.cgi
|   |   |   |-- simpleListing.tmpl
|   |-- ut-dir
|   |-- ut-htgroup
|   |-- ut-index
|   |-- ut-install
|   |-- ut-marked
|   |-- ut-markingScheme
|   |-- ut-new
|   |-- ut.1
• ~/wwdev/UT_0.2/ Bin directory contains all the command line utilities/

• ~/wwdev/UT_0.2/ Bin/ InstallFiles directory contains template files used but the ut-install command line utility.

• ~/wwdev/UT_0.2/ Pm/ Local directory contains the Perl modules used by the unit template. ~/wwdev/UT_0.2/ Pm/ Cpan directory contains all the locally installed Perl modules obtained from CPAN [2].

There also exists the ~/wwdev/ UT_0.2 directory, which is simply a symbolic link to the ~/wwdev/ UT_Version_0.2/ Pm/ Local/ directory and is used to make the inclusion of the Perl modules user friendly, for example:

use lib qw(/homes/admin/wwwdev/UT_0.2)
Bibliography


