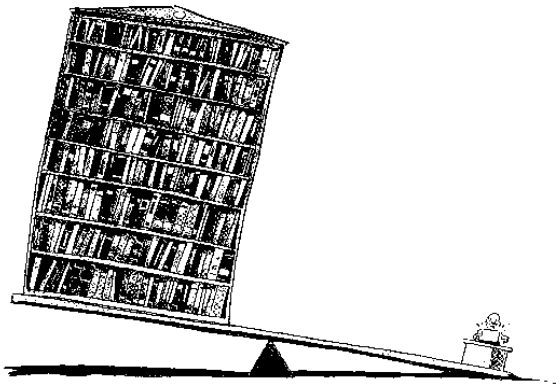
KE Texpress



Texhtml Guide

KE Software Pty Ltd

Contents

Chapter 1
Introduction

Overview

KE Texhtml is an application built using the KE Texpress Object Oriented Database System. It is designed to accept requests from Internet users in the form of Uniform Resource Locators (*URL*s) and perform queries on one or more KE Texpress databases to produce a document of the results of the query. The derived document is in HyperText Markup Language format (*HTML*) and is automatically transmitted to the Internet user who submitted the original request.

KE Texhtml is a series of processes designed to work in conjunction with the HyperText Transfer Protocol Daemon (*httpd*), a server daemon for the transmission of HTML documents. *httpd* responds to Internet requests from HTML browsers such as Netscape, Internet Explorer and Mosaic.

Using This Manual

This manual, the KE Texhtml Internet Services Guide, is one of a suite of manuals describing the KE Texpress Information Management System.

The remainder of this chapter contains definitions of the terminology used throughout this manual and an overview of the KE Texhtml system.

Chapter 2 contains installation instructions for KE Texhtml. Chapter 3 provides a detailed description of the operation of KE Texhtml while Chapter 4 provides an example of how the system can be applied. Finally, Chapter 5 contains information about the auditing, security and error checking supporting by KE Texhtml.

KE Texhtml Terminology

KE Texhtml uses a specific terminology to describe the various processes and protocols which it supports.

The Internet

The Internet is a loosely structured network of networks of machines which extends to most parts of the world. Its communication is based on the Internet Protocol (IP) which is packet based and supports the concept of unique hierarchically based machine addresses (e.g. **203.5.6.1**). Each packet contains a destination address which is used to forward the packet from machine to machine across the Internet until it reaches its ultimate destination. Forwarding of packets is performed immediately, thus supporting live interaction between two machines several thousand kilometres apart.

Although the Internet refers to the global network of machines, it is possible to establish a local, detached "Internet" using the same networking protocols. Such a smaller scale "Internet", called an Intranet, can still support all of the features described in this manual.

Host name

Each machine on the Internet is assigned a host name. This name consists of a local host name and a domain name (e.g. **kestrel.ke.com.au** where **kestrel** is the local host name and **ke.com.au** is the domain name). Many machines on the Internet act as domain name servers which means they can convert from a host name to an IP address and vice versa. A Domain Name Server is also called a DNS.

Server

A machine connected to the Internet can choose to become a server machine in which case it provides Internet users with some sort of

service on demand. Most services relate to holding a variety of files which can be down loaded to an Internet user's machine when requested. Other services allow users to establish an interactive connection to a server machine and execute commands on that machine.

Client

A user on a machine connected to the Internet is often referred to as an Internet client. The user can use one of a suite of programs available to Internet clients to extract information from Web Servers.

Communication Protocol

A client and a server can only communicate if they both support the same set of rules for communication, i.e. protocol. This essentially describes the format in which the request is submitted and the response provided. Some examples of Internet communication protocols include **File Transfer Protocol** or **FTP**, **Wide Area Information Service** or **WAIS**, **Gopher** and **HyperText Transfer Protocol** or **HTTP**.

Uniform Resource Locator, URL

A URL is essentially an address of the following format:

```
protocol://host name/resource name
```

where ***protocol*** specifies the communication protocol to be used between the client and the server machine. Depending on the client process, or browser, being used, this can be one of many different protocols. For KE Texhtml the protocol must be **http**. The ***host name*** is the local host name and domain name of the server machine, e.g. **kestrel.ke.com.au**. The ***resource name*** is the name of a resource on that machine. For *http*, it is a file or the name of a program, and its arguments, which will generate a file. This latter option is used to invoke KE Texhtml.

Netscape, Internet Explorer, Mosaic

Netscape, Internet Explorer and Mosaic are browsers for client machines available for MS-Windows, Macintosh and X terminals. Mosaic is in the public domain. These browsers support a number of communication protocols including FTP, WAIS, Gopher and HTTP. KE Texhtml can be used in conjunction with any of these or any other suitable browser. For convenience, this document uses Netscape in all of its examples but any suitable browser could be substituted.

httpd

httpd is a server program supporting the HTTP protocol. It responds to *URL* requests from Netscape. It extracts the file name component and returns to the client the contents of the file. Alternatively, if the file is in fact a program, it executes the file to produce information to be sent to the client. Both NCSA and CERN offer public domain versions of *httpd* while commercial versions are also available.

HyperText Mark-up Language, HTML

HTML is a definition of a series of markers which can be placed in ASCII files. When viewed with an appropriate browser, such as Netscape, these markers are processed to provided a variety of display facilities. These facilities include formatting of the displayed text, hypertext links within and between documents and the embedding of non-textual information in a document, such as images, voice, etc. For more information on HTML refer to the HyperText Markup Language Internet Draft and the HTML+ Discussion Document, both available from various server machines on the Internet.

Overview of KE Texhtml

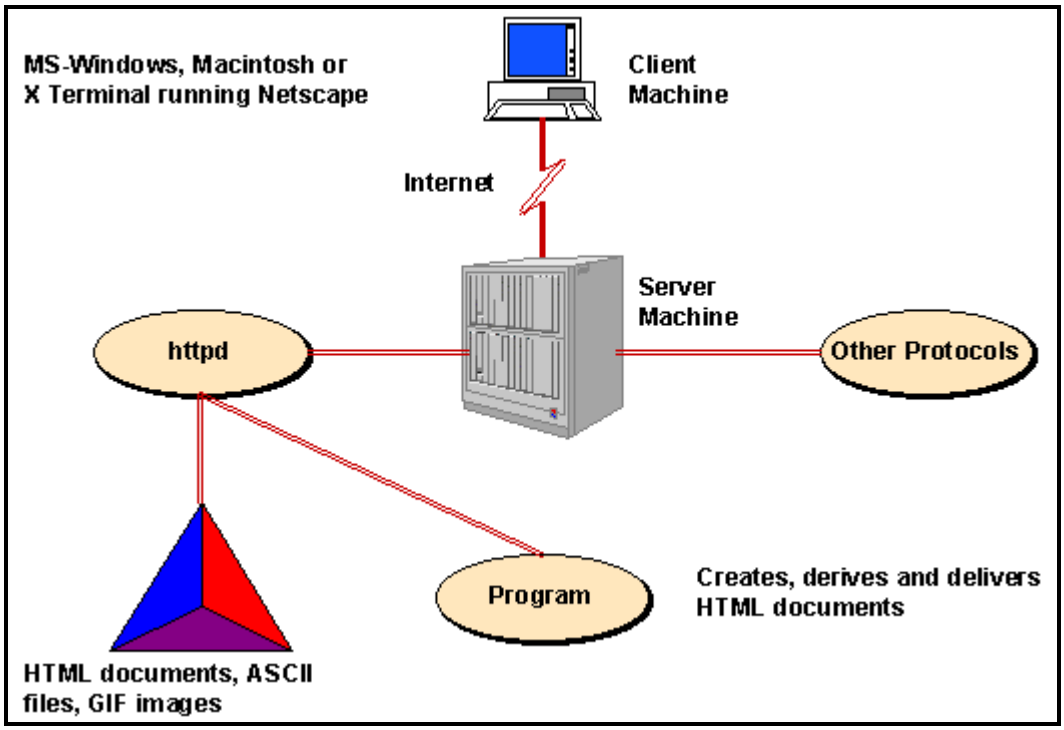
KE Texhtml is designed to support an HTTP server machine to provide HTML documents to Netscape Internet clients.

A Netscape client selects a *URL* to connect to a server machine and retrieve a resource from that machine. Although the resource can be in any format, Netscape is designed to support HTML documents and correctly interpret the markers contained within them.

HTML documents can contain a variety of formatting markers which make the presentation of the document more visually appealing. They can also contain **anchors** which are hypertext links to positions within the same document or to other documents on the Internet. References to other documents are made by including their URL in the anchor. In this way, users need only identify a starting URL after which they can navigate through a series of documents, perhaps on a series of different server machines on the Internet, by use of the embedded URLs. This gives rise to the concept of the **World Wide Web** (WWW) where files spread across server machines throughout the world are linked together in a variety of ways, like a spider web.

The HTTP server daemon, *httpd*, provides a means of delivering static HTML documents to Netscape clients upon request. However, it is also extensible, such that a request can result in the execution of a program on the server machine which will generate an HTML document.

The figure below gives an overview of this process.



HyperText Transfer Protocol Daemon

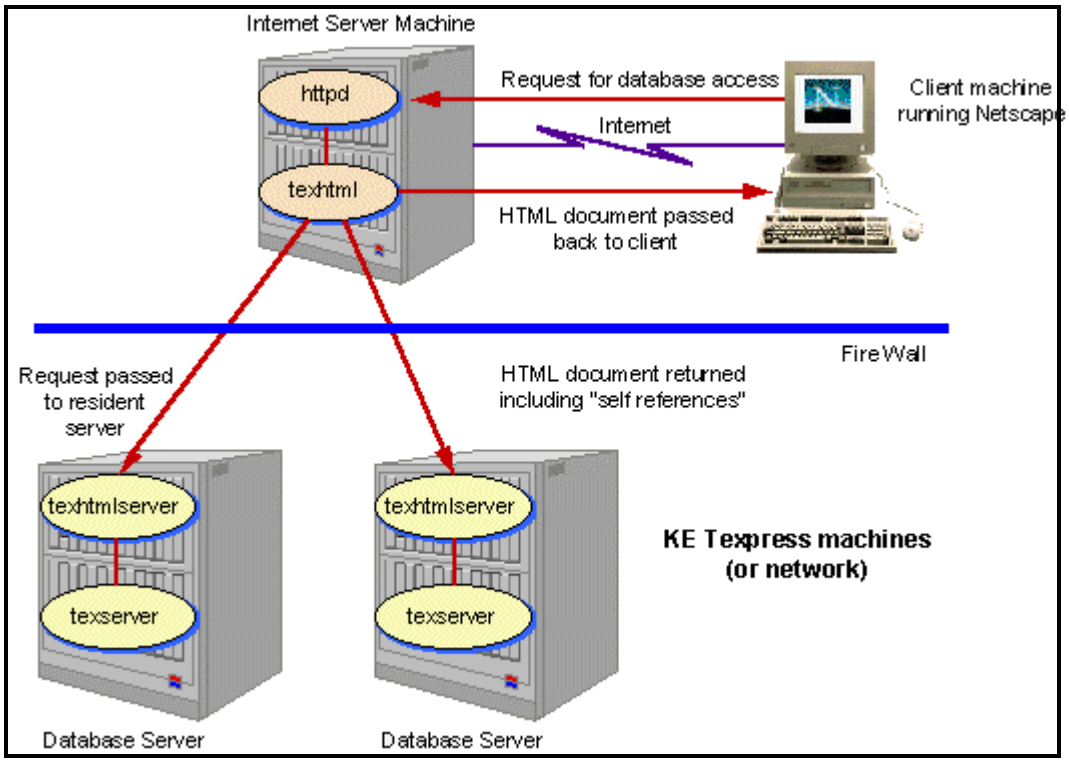
KE Texhtml has been designed to be invoked by *httpd*. When a URL with a file name of **/cgi-bin/texhtml** is requested, the request is transmitted across the Internet to the server machine's *httpd* which then invokes *texhtml*. Depending on the associated arguments, one of two actions is taken:

1. If invoked with one argument, KE Texhtml determines the appropriate host machine at the server site from its configuration files and generates HTML document (or downloads an existing document) from that host. This document is generally in the shape of a query form for a database or databases.

2. If invoked with more than one argument, KE Texhtml uses its first argument to determine, from its configuration files, the host on which the database exists, the template *Texql* statement to be performed and the template HTML document to be produced. It then substitutes the remaining arguments into the *Texql* query before it is executed.

KE Texhtml consist of three processes. *texhtml* is invoked by *httpd* for every Internet request. For each of the above actions, it connects to one of possibly several resident *texhtmlserver* processes on one of possibly several machines. This process is a KE Texpress C-API based application and so on start-up, it in turn connects to a *texserver* process.

The separation of these processes enables the construction of a *fire wall* providing secure access to machines containing databases and restricting Internet access to one machine which contains no privileged data. This fire wall security approach is shown in the figure below.



KE Texhtml Process Model

Chapter 2
Installing KE Texhtml

Overview

This chapter describes the installation process required for KE Texhtml.

KE Texhtml consists of three processes. In most cases, all of these processes will run on the same machine. However, it is possible to spread the system across a range of machines. These installation instructions assume the latter.

Planning Your Installation

KE Texhtml consists of three processes, *texhtml*, *texhtmlserver* and *texserver*. In most cases, all of these processes will run on the same machine. However, it is possible to spread the installation throughout a network of machines.

texhtml is a small process invoked by *httpd* directly. It is invoked for every Internet request requiring database access and must reside on the same machine as *httpd*. This machine is called the **Web Server**.

texserver is the server component of KE Texapi, the KE Texpress Client/Server Application Programming Interface or API. It must reside on each machine that contains databases which are to be accessed via the Internet.

Finally, there must be a machine running *texhtmlserver* for every machine running *texserver*. Generally, this will be the same machine and is referred to as the **Database Server**.

The Web Server and the Database Server can be the same machine. Alternatively, there can be one Web Server connected to many Database Servers. In this scenario, the Web Server may also be a Database Server.

Each Database Server can run several *texhtmlserver* processes. This enables the Database Server to respond to several Internet requests simultaneously.

Before Installing KE Texhtml

KE Texhtml requires that KE Texpress be installed on the machine or machines on which the KE Texpress databases reside, the Database Server(s). Consult the KE Texpress Installation Guide for assistance.

texhtmlserver is a KE Texapi application and so runs as a client/server application. It conforms to the conventions of KE Texapi applications by accepting configuration parameters for the API, including the connection to the server, *texserver*, as on-line arguments. If *texhtmlserver* and *texserver* are to run on different machines, then it is necessary that *texserver* is configured to be invoked by **inetd**. Again, refer to the KE Texpress Installation Guide for assistance.

It is also necessary that *httpd* be installed and running on the machine connected to the Internet, the Web Server.

If fire wall security is to be installed (possibly through the use of an intelligent router), then it should be placed between the Web Server (which is open to the Internet) and the Database Server(s).

The Web Server and the Database Server(s) should all be connected via TCP/IP. Note that it is possible for all of these processes to reside on the one machine, if desired.

Finally, a machine connected via TCP/IP to the Web Server should also have a suitable HTML browser, such as Netscape, already installed.

Installing texhtmlserver

texhtmlserver is a process normally started when a machine is booted. It establishes a connection to *texserver* and then listens on a port for requests from *texhtml* processes. It must be installed on each Database Server.

Installation of *texhtmlserver* requires first that the line:

```
texhtml   5982/tcp
```

be placed in the file **/etc/services**. It can then be invoked using a command of the form:

```
texhtmlserver [-T...] [-bdir] [-hh1:h2:...] [-nn]
              [-qc] [-ssrv]
```

where the option **-T...** is used to pass arguments to the KE Texpress API. The option **-b*dir*** specifies a directory containing configuration files used by *texhtmlserver*. This option may be repeated thus specifying multiple directories which *texhtmlserver* will search for configuration files. The default directories, **/usr/local/etc/texhtml** and **/usr/local/lib/htmldocs**, are always searched.

The option **-h*h1*:*h2*:...** instructs *texhtmlserver* to accept connections only from hosts ***h1***, ***h2***, etc. By default, *texhtmlserver* accepts connections from any hosts.

The option **-n*n*** specifies the number of *texhtmlserver* processes to be launched. The default is **1**.

The option **-q*c*** specifies the quote character used by *texhtmlserver*. The default is #.

If more than one *texhtmlserver* is launched, then all listen on the same port (number 5982) for requests from *texhtml*. In this way, more than one simultaneous Internet request can be serviced. If an Internet request arrives while all *texhtmlserver*s are busy servicing prior requests, then it is placed in a queue. Should the queue be considered full, then the request is denied and *texhtml* automatically generates an HTML document indicating that the servers are all busy.

The option **-s*srv*** indicates that the service ***srv*** should be used rather than the default **texhtml**. This service must be registered in */etc/services*. This option enables multiple sets of *texhtmlserver* processes to be running, possibly

under different Unix User Ids, accessing different databases or using different KE Texpress options, such as supporting different languages. It can also be useful to provide secure (password protected) access to some databases while allowing global access to others.

KE Texhtml includes three programs to launch, monitor and terminate *texhtmlserver* processes. These are:

texhtmlstart [*srv*] [*options*]

This program launches *texhtmlserver* with the additional arguments specified as ***options*** on the service, ***srv***. With no arguments, it launches one server on the default *texhtml* service. It terminates any servers running on the service first, thus also performing a restart.

This program searches for an appropriate base directory for document templates, trying:

1. **~texpress/texhtml/*srv***
2. **/usr/local/etc/*srv***
3. **/usr/local/lib/htmldocs**

and adds a **-b…**argument to *texhtmlserver* if none is given. It then determines the owner of the base directory. If the current user is not the owner of the directory or the superuser, then *texhtmlstart* terminates. Otherwise, it runs as the owner of the directory. The directory generally should be owned by a restricted Unix user account, such as **www** or **daemon**.

texhtmlstatus [*srv*]

This program reports the status of the servers running on the service, ***srv***. With no arguments, it reports all running servers on all services.

texhtmlstop [*srv*]

This program attempts to terminate the servers running on the service, ***srv***. With no arguments, it terminates all running servers on all services

Generally, these three programs are run as the superuser.

texhtmlstart should be included in the start-up procedure for the machine. On most versions of Unix, this means that a script called something like **/etc/rc2.d/S90texhtml** should be created containing commands similar to the following:

```
~texpress/bin/texhtmlstart -n10
~texpress/bin/texhtmlstart texhtmls -n5
          (Expand ~ to home directory)
```

This example starts ten *texhtmlserver* processes on the default *texhtml* service. It then starts a further five processes on the *texhtmls* service, which may be running as a secure service.

Installing texhtml

texhtml is a process invoked for every Internet request for database access. It determines the host containing the database and then establishes a connection to *texhtmlserver* on that machine.

texhtml must be installed on the machine on which *httpd* is installed. Generally, *httpd* is installed in the directory, **/usr/local/etc/httpd**. *texhtml* can then be installed in a script sub-directory of this directory (generally, **cgi-bin**). This directory must be configured within *httpd* as containing executable programs. This is achieved, using the NCSA *httpd*, by adding a line of the form:

```
ScriptAlias /cgi-bin/    /usr/local/etc/httpd/cgi-bin/
```

to the *srm.conf* configuration file of *httpd*. Consult the *httpd* installation instructions for more information.

All URLs referencing *texhtml* would then need to be in the form:

```
http://host name/cgi-bin/texhtml[?args]
```

Any arguments which form part of the URL are merely appended to the program name, separated by ampersand characters (**&**).

To install *texhtml* in the directory, **/usr/local/etc/httpd/cgi-bin**, simply change directory to the texpress bin directory and enter the command:

```
cp texhtml /usr/local/etc/httpd/cgi-bin
```

Note that the cgi-bin directory may not exist and may need to be created.

texhtml uses its name to determine the service to which it should connect. If a service name other than *texhtml* is to be used, a link to *texhtml* should be created by that name and that link should be invoked within a URL. For example, if it is desirable to have two invocations of *texhtmlserver*, one which responds to Chinese language requests and searches Chinese language databases while the other responds to Japanese language requests and searches Japanese language databases, then a second port, say *texhtmlj*, could be used for the Japanese databases. The Japanese language invocation of *texhtmlserver* would be invoked with the option **-stexhtmlj** (e.g. *texhtmlstart texhtmlj*). It would also have the KE Texpress option **langcode=sjis** or **langcode=euc** set while the Chinese invocation would need either **langcode=big5** or **langcode=gbcode** (refer to the KE Texpress User Guides). A link to *texhtml* called *texhtmlj* would also need to be created. Internet requests accessing *texhtmlj* would then be serviced by the Japanese language invocation of *texhtmlserver*. (Note that both the Chinese

and Japanese language *texhtmlserver* invocations could access English databases.)

Installing the KE Texhtml Configuration Files

KE Texhtml uses a series of configuration files to determine how to respond to Internet requests. These files generally reside in the base directory, by default **~texpress/texhtml**.

An example set of configuration files is provided with KE Texhtml and can be copied to the appropriate directory by performing the following commands:

```
cd ~texpress/etc/texhtml
find texhtml -depth -print | cpio -pduv ../..
```

The contents and function of the configuration files is discussed in Chapter 3 of this manual.

An audit database is included in the release and can be installed by typing the following commands:

```
cd ~texpress/etc/texhtml
find htmlaudit -depth -print | cpio -pduv ~/data/local
texperms htmlaudit
```

This *htmlaudit* database is discussed in Chapter 5 of this manual.

Chapter 3
Using KE Texhtml

Overview

This chapter describes the operation of KE Texhtml. It also defines the structure and purpose of each of the configuration files and document templates used by the system.

KE Texhtml Arguments

URLs where the file name component represents a program on the Web Server result in *httpd* invoking that program on the specified machine. Furthermore, a URL can contain arguments for the program. These arguments are appended to the file name component of the URL, following a question mark character and are separated by the ampersand character. So for example, the URL:

```
http://www.ke.com.au/cgi-bin/texhtml?form=AustWines
```

invokes the program, *texhtml* on the Web Server and passes it the single argument, **form=AustWines**. (Note that Netscape forms can use a method called **POST** which sends arguments to programs via their standard input. This method is also supported by KE Texhtml.)

KE Texhtml can be invoked in one of two formats:

1. If invoked with one argument, KE Texhtml assumes that the argument corresponds to an HTML query form for a database.

2. If invoked with more than one argument, KE Texhtml assumes that it must perform a Texql statement to create an HTML document to deliver to the Internet user. The Texql statement is generally a query on one or more databases but it can also use the Texql DML (data manipulation language) to insert, update or delete data in a database.

The remainder of this chapter describes each of these actions in more detail.

Determining the Database Server

When *texhtml* is invoked, it first consults its configuration file. It looks for one of the following files:

1. **${TEXHTMLDIR}/admin/config**
   where ${TEXHTMLDIR} is the standard Unix environment variable. This variable should be set up before *httpd* is invoked so that *texhtml* inherits it every time it is invoked.

2. **~texpress/texhtml/*service*/admin/config**.

3. **/usr/local/etc/*service*/admin/config**.

4. **/usr/local/lib/htmldocs/admin/config**.

The configuration file, if present, should consist of lines of the form:

```
host:doc1,doc2,...:Description
```

where *host* is the host name of the machine running a *texhtmlserver* which can respond to requests referencing document names, *doc1*, *doc2*, etc. (see below).

The *Description* component is free text describing the associated database(s).

Delivering Query Forms

If *texhtml* is invoked with one argument, *form=docname*, then it consults the *config* file described above for a document by that name. If the name is found, then it uses the associated host name. Otherwise it assumes **localhost**. It then establishes a connection to *texhtmlserver* on that host. It passes *docname* to *texhtmlserver*.

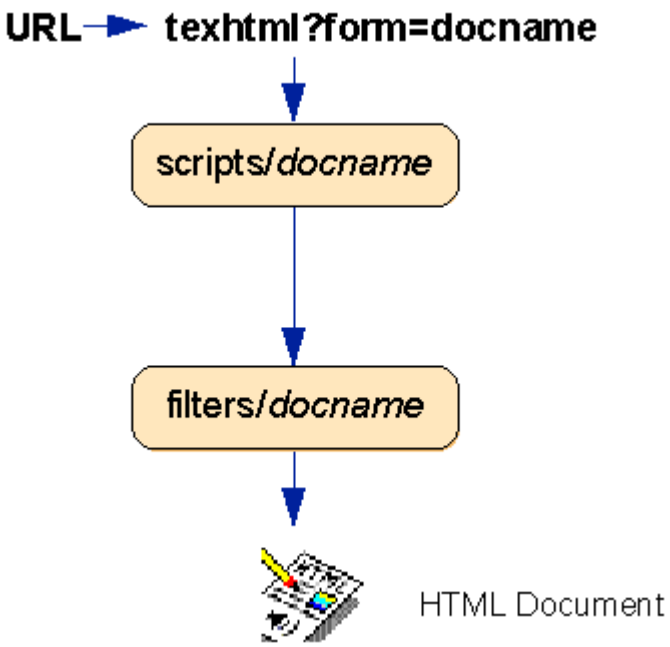
The entire environment of *texhtml* excluding its PATH variable is also passed to *texhtmlserver* which overlays this on its own environment before servicing the request.

texhtmlserver searches its configuration files (in **~texpress/texhtml/*service*** by default) for the file, **queries/*docname*.** This file should contain an HTML document describing a query form for a database. *texhtmlserver* then returns this to the Internet user.

HTML forms are defined to have an action, which is a URL. They can also contain input boxes, check boxes, etc. Any information entered by the user into these input regions is appended to the URL as additional arguments. If the form's *submit* button is selected, the URL with all of its arguments is submitted to the Web Server. This is the mechanism used to perform searches and updates on databases.

Before returning the document to the user, *texhtmlserver* searches for an executable file, **filters/*docname***, in its configuration directories. If one is found, the HTML query form is passed through this filter before being returned. Otherwise if the executable file, **filters/default**, exists then the query form is passed through this filter and returned to the user. As *texhtmlserver* merges the environment of *texhtml* into its own before servicing the request, these filter programs can access environment variables set up by *httpd*. Such environment variables can include *REMOTE_HOST* and *REMOTE_USER*. Refer to the httpd documentation for more information.

This process is as follows:



This operation is audited as discussed in Chapter 5.

Performing Operations on Databases

texhtml can also be invoked with more than one argument, generally as a result of a user having submitted a query after filling in an HTML query form but also from completing an HTML insertion or update form. In this case, it uses its first argument, *form=docname*, to search the *config* file for a document by that name. If the name is found, then it uses the associated host name. Otherwise it assumes **localhost**. As before, it then establishes a connection to *texhtmlserver* on that host. It passes *docname* and any other arguments to *texhtmlserver*. The entire environment of *texhtml* excluding its PATH variable is also passed to *texhtmlserver* which overlays this on its own environment before servicing the request.

The arguments passed to *texhtmlserver* should be of the form:

```
name=value
```

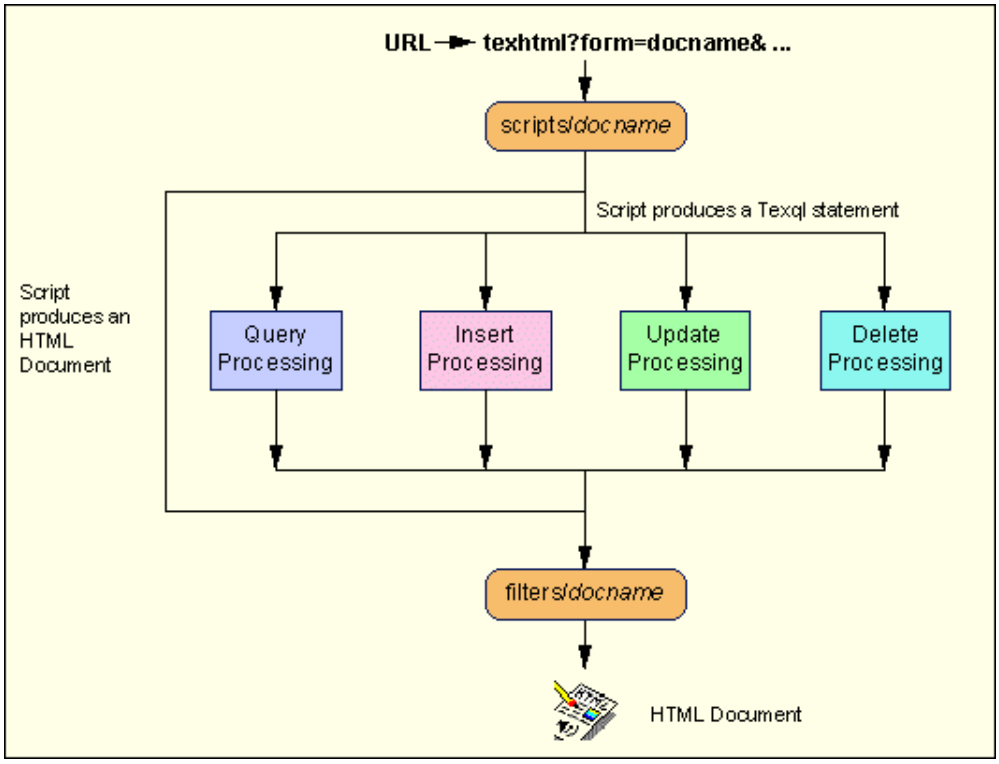
In broad terms, *texhtmlserver* uses the script file, **scripts/*docname***, to generate a Texql statement which is then submitted to the *texserver*. The Texql statement may be a DML (data manipulation language) statement. An HTML document template can be created for the results of each type of DML statement (insert, update and delete). This is generally used simply to report the success of the operation.

Alternatively, if a Texql query is generated by the script, then the query is performed.

In each case, *texhtmlserver* searches for an executable file, **filters/*docname***, in its configuration directories. If one is found, the created document is

passed through this filter before being returned. Otherwise if the executable file, **filters/default**, exists then the document is passed through this filter and returned to the user. The script also has its environment set to the merged environments of *texhtmlserver* and *texhtml*.

The following diagram demonstrates the various actions resulting from a Texql statement.



Building Texql Statements

In order to build the Texql statement to be performed, *texhtmlserver* searches for a file, **scripts/*docname***, which is generally a multi-line *Texql* statement. Within this statement, it looks for occurrences of the string **#*name*#**, searches its arguments for the ***name*** and replaces those occurrences with the associated ***value***. If a referenced *name* does not appear in the arguments, then the entire line is omitted. In this way, only input boxes where values have been entered form part of the resulting statement.

KE Texhtml supports the HTML *TEXTAREA* data type. This data type supports multi-line text strings (sequences containing newline characters). By default, the sequence, **#*name*#**, strips newlines from the data before making the substitution. Alternatively, the sequence, **#$*name*#**, can be used to make the substitution *without* stripping the newline characters. Finally, the sequence, **#*name*:*n*#**, can be used to substitute only the ***n***th line of the input text.

Alternatively, if the file, **scripts/*docname***, is an executable file, then *texhtmlserver* executes it to produce the statement to be performed. It passes

all but its first argument, *form=docname*, to the script. The script also has its environment set to the merged environments of *texhtmlserver* and *texhtml*.

If the "Texql statement" produced as a result of this step commences with the sequence:

```
Content-type:
```

or

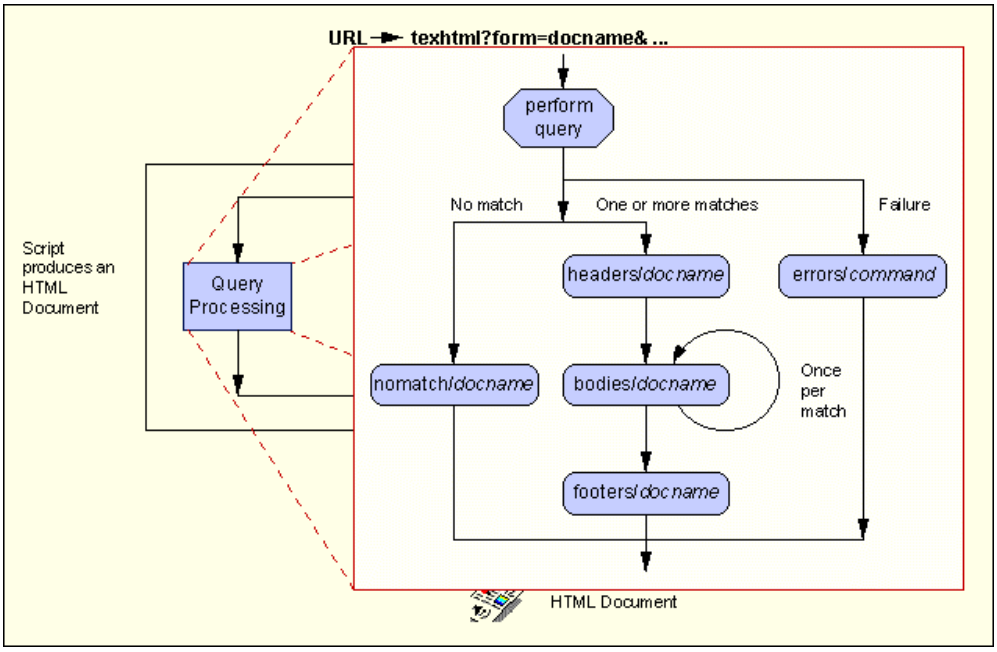
```
Location:
```

then it is considered to be a valid HTML document. No attempt is made to perform the statement but rather it is returned to the user (via the filter, if any). This can be useful when it is necessary to perform validation on the data entered by the user before deciding whether to perform the operation.

Otherwise, the statement produced is submitted to the database engine, *texserver*, for processing.

Building HTML Documents from Queries

If the script produces a Texql query, then the query is performed. Processing is as described below:



The query statement will result in one of the following three actions:

1. **An error occurred while processing the query**

   The HTML document template, **errors/Command**, is returned. Processing of built-in functions is performed as described below.

2. **No records match the query**

The HTML document template, **nomatch/*docname***, is returned. If this document does not exist, the document, **nomatch/default**, is returned. Processing of built-in functions is performed as described below.

texhtmlserver filters the matching records through a *body* HTML document template, with the results surrounded by a header and footer HTML document template. The following diagram demonstrates the various actions resulting from a Texql statement.

3. At least one record matched the query

For query statements resulting in one or more matching records, *texhtmlserver* creates an HTML document by assembling the following files:

headers/*docname* at the top with the first matching record loaded

bodies/*docname* once for each matching record

footers/*docname* at the end with the last matching record loaded

All of the HTML document template files should contain text and valid HTML markers only.

texhtmlserver searches these files for occurrences of the sequence, **#*colid*#**. It then searches the appropriate matching record for a column of that name and substitutes the data in that column for the entire sequence.

If the selected column is a nested tuple (e.g. a date item), then the individual columns of the tuple are substituted into the document separated by spaces. If the selected column is a nested table (e.g. a list of names), then the individual rows of the tuple are substituted into the document separated by newlines. Note that *texforms* style field references, e.g. *name_2*, are also supported enabling individual rows to be extracted from nested tables.

The optional URL operator, **!**, can appear directly after the opening # (e.g. **#!name#**), in which case the data from the record has a series of transformations applied to it, including the following:

space	becomes	**+**
!	becomes	**%21**
#	becomes	**%23**
%	becomes	**%25**
&	becomes	**%26**
*****	becomes	**%2A**

```
+       becomes   %2B

?       becomes   %3F
```

This sort of transformation should be applied if the data is to form part of an embedded URL in the document being created (the reverse transformation is applied when the original arguments are received).

If the selected column is a nested tuple (e.g. a date item), then the individual columns of the tuple are substituted into the document separated by the plus character (+). If the selected column is a nested table (e.g. a list of names), then the individual rows of the tuple are substituted into the document separated by the pipe character (|).

The optional highlight operator, +, can appear directly after the opening # (e.g. #+**name#**), in which case the data from the record has matching term (highlighting) information embedded in it, including the following:

<-- MATCH COUNT *name count* -->
: This HTML comment is embedded at the start of the data and includes the name of the column and the number of sequences of data matching the query (e.g. individual words or phrases).

<-- MATCH ON -->
: This HTML comment is embedded within the data at the start of a word or phrase matching the query.

<-- MATCH OFF -->
: This HTML comment is embedded within the data at the end of a word or phrase matching the query.

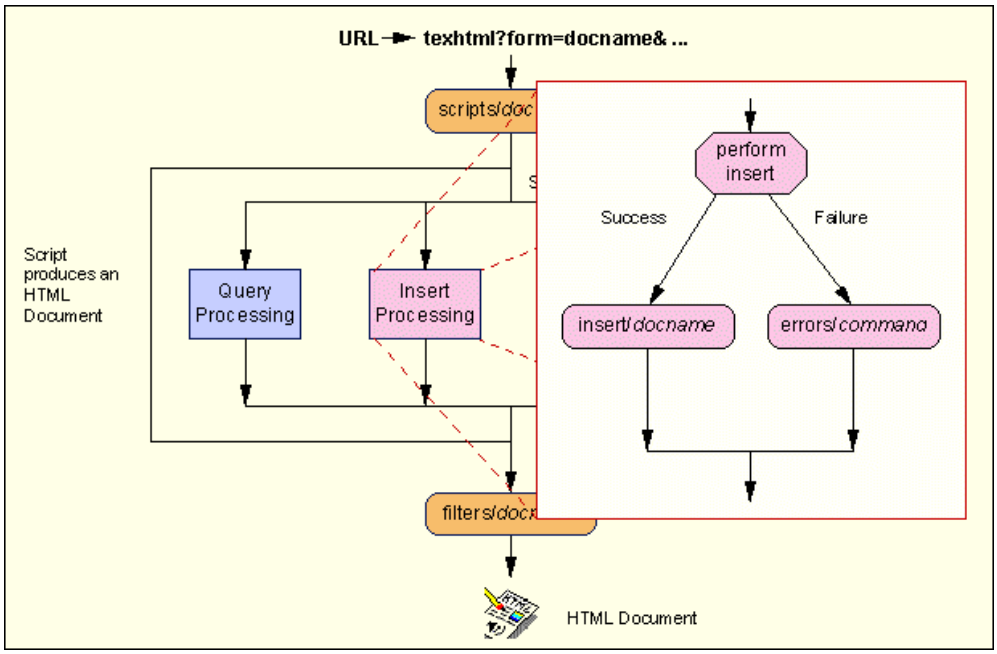
Other utilities, such as the filter described above, can further process these comments, e.g. substituting bold markers (**<B>** and **</B>**) to highlight the matching terms.

Note that the URL and highlight operators cannot be used together.

If the document template file is to contain a # character, it should be preceded by a backslash (\) so that it is not interpreted as a column reference. *Texhtmlserver* can also be invoked with the **-q*c*** argument in which case the character ***c*** should be used to surround column references, rather than #.

Inserting Records into Tables

If the script produces a Texql insert statement, then the insertion is performed. Processing is as described below:



The insert statement will result in one of the following two actions:

1. **An error occurred while processing the insertion**

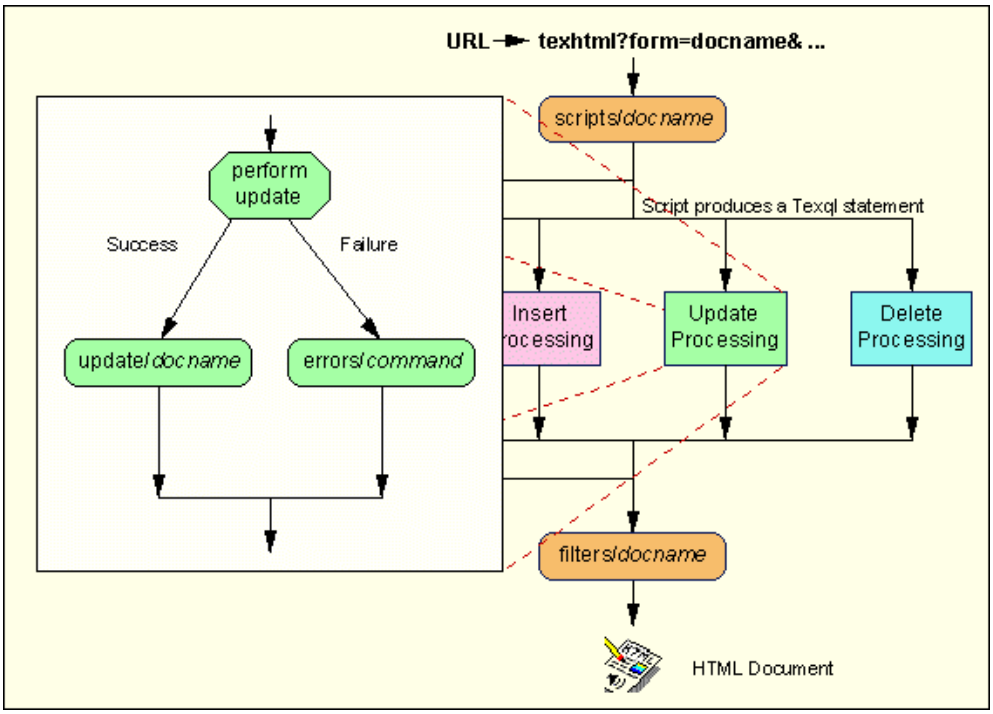
   The HTML document template, **errors/Command**, is returned. Processing of built-in functions is performed as described below.

2. **The insertion or insertions are successfully completed**

   The HTML document template, **insert/*docname***, is returned. If this document does not exist, the document, **insert/default**, is returned. Processing of built-in functions is performed as described below.

Updating Records in Tables

If the script produces a Texql update statement, then the update is performed. Processing is as described below:



The update statement will result in one of the following two actions:

1. **An error occurred while processing the update**

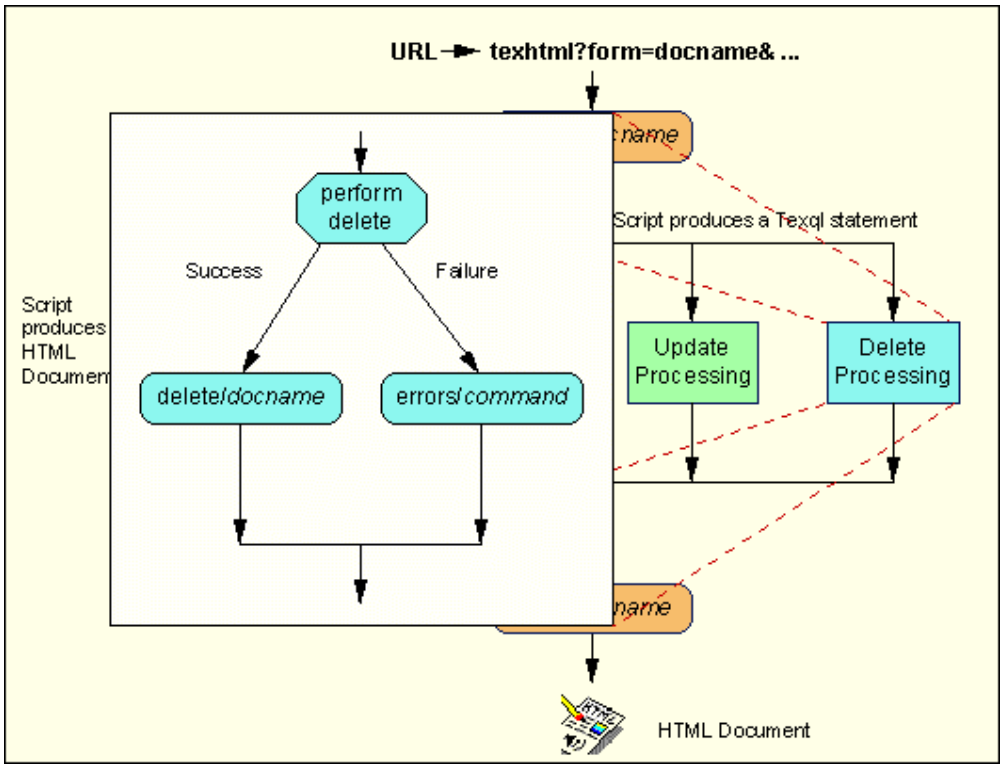
   The HTML document template, **errors/Command**, is returned. Processing of built-in functions is performed as described below.

2. **The update are successfully completed on zero or more records**

   The HTML document template, **update/*docname***, is returned. If this document does not exist, the document, **update/default**, is returned. Processing of built-in functions is performed as described below.

Deleting Records from Tables

If the script produces a Texql delete statement, then the deletion is performed. Processing is as described below:



The delete statement will result in one of the following two actions:

1. **An error occurred while processing the deletion**

   The HTML document template, **errors/Command**, is returned. Processing of built-in functions is performed as described below.

2. **The deletion of zero or more records was successfully completed**

   The HTML document template, **delete/*docname***, is returned. If this document does not exist, the document, **delete/default**, is returned. Processing of built-in functions is performed as described below.

Built-in Functions

texhtmlserver supports a series of built-in functions enabling HTML documents produced by the system to include dynamic data related to the operation performed. These functions include:

#TexCommand#
Display the Texql command which was submitted to the server.

#TexCursorType#
Display the Texql command which was submitted to the server.

#TexErrMsg#
Display the error message generated by the server from the last Texql command.

#TexErrOff#
Display the offset associated with the error message generated by the server from the last Texql command. This represents the point at which the error was discovered in the statement.

#TexError#
Display the error number generated by the server from the last Texql command.

#TexFormName#
Display the Form Name of document template name currently being processed.

#TexRowCnt#
Display the number of records matching the query or affected by the insert, update or delete statement.

#TexRowHits#
Display a (very close) approximation of the number of matching records. This operation is much faster than **#TexRowCnt#**.

#TexRowPos#
Display the number of the current record being processed. This number is from **1** to the number of rows matching the query.

#TexVersion#
Display the version number of the KE Texpress software.

Some of these built-in functions are used by some of the standard document templates. For example, the **errors/Command** document uses the *#TexError#*, *#TexErrMsg#* and *#TexErrOff#* functions, while the **insert/default**, **update/default** and **delete/default** documents use the *#TexRowHits#* function**.**

Resolving Form Names

texhtmlserver uses the form name passed to it by *texhtml* to identify one or more files to be used to service the request. These files contain information used for various purposes including:

- an HTML query or data capture form,
- a Texql statement or a program to generate a Texql statement,
- an HTML document template into which data from matching records can be displayed,
- a filter program to *post-process* the data generated by *texhtmlserver*, etc.

KE Texhtml supports a broad range of rules to resolve which files should be used for any particular request. This gives the designer of the Web interface maximum flexibility in the actual location of the set of files making up a form name. In broad terms, it attempts to locate the file within the directory of the database or the package (multi-database application) to which it relates or failing this, in one of the standard directories used by the program.

If *texhtmlserver* requires a file for a particular form name, it uses any full stop characters in the name to parse it into one of the following four formats:

- ***package.database.formname***
  The first component refers to a KE Texpress package installed on the Database Server.

- ***package.formname***
  The first component refers to a KE Texpress package installed on the Database Server.

- ***database.formname***
  The first component refers to a KE Texpress package installed on the Database Server. The difference between this and the previous option is determined by context.

- ***formname***
  If a KE Texpress package called ***formname*** is installed on the Database Server, then this expanded to the form ***package.formname*** where the ***package*** is also set to ***formname***. Otherwise this name indicates that the document is not directly associated with a package or database.

KE Texhtml uses these naming conventions to determine the appropriate components of the name and then attempts to locate that file in the following places:

1. **~*package*/texhtml/*database***

2. **~*package*/data/*database*/**

Chapter 4

KE Texhtml Example Database

Overview

This chapter describes a KE Texhtml interface to a database of wines. It first describes the *AustWines* database and then looks at the structure of each of the configuration files related to this database. It also shows the effect of these configuration files when viewed using the MS-Windows version of Netscape.

The *AustWines* Database

The *AustWines* database is a standard KE Texpress database describing a series of wines produced in Australia. It contains fields such as **Wine Name**, **Wine Type**, **Tasting Notes** and **Best Served With**. Its Texql description is as follows:

```
AustWines[
    wineno(
        wineno_1   integer
    ),
    winename   text,
    type       text,
    style      text,
    class      text,
    year       integer,
    priceper   float,
    bestin     integer,
    tastingn   text,
    personal   text,
    bestserv_tab[
        bestserv   text
    ],
    numberof   integer,
    owner      text,
    frontima   text,
    backimag   text
];
```

Of particular note are:

- *wineno*, a key item and nested tuple,
- *bestserv_tab*, a nested table (multi-field item), and,
- *frontima* and ***backimag***, which are references to externally stored images corresponding to the front and back labels of the bottle. These images are in GIF format.

Most of the other columns are of type *text*.

The remaining sections of this chapter describe a set of KE Texhtml configuration files which produce a relatively simple Netscape interface for searching this database.

An Example *config* File

KE Texhtml consults its *config* file to determine which Database Server can service particular requests. This file is generally stored in **/usr/local/etc/texhtml/admin/config**. A sample *config* file follows:

```
kestrel:cities,cities.sum,cities.all:Database describing various cities
    around the world.
localhost:hansard:The complete text of Hansard from Victorian Parliament
    for 9 March 1993 through to 10 March 1994.
kestrel:movies,movies.sum,movies.all:A collection of information about
    the best movies of the last forty years, including the director,
    cast, year in which each was made and a description of the movies'
    highlights.
kestrel:AustWines,AustWines.sum,AustWines.all:Information about some of
    Australia's best wines.
```

This file contains entries for four databases. Three of those databases reside on a Database Server called ***kestrel***, connected to the Web Server (perhaps via a firewall). The other resides on the Web Server itself.

Each line then contains a comma-separated list of document template names related to that database and a description of the database.

The HTML Query Form

When invoked with a single argument of **form=AustWines**, KE Texhtml consults its *config* file to determine the appropriate Database Server (*kestrel*). It then establishes a connection to *texhtmlserver* on *kestrel* and passes to it the document template name (*AustWines*) and the environment, including the host name of the connecting user (e.g. **REMOTE_HOST=john.ke.com.au**).

texhtmlserver then searches for the file **forms/AustWines** in its *config* directory. If the file **filters/AustWines** exists, or failing this the file **filters/default**, the *AustWines* query form is passed through this filter and returned to the Internet client. In this example, no such filter exists so the AustWines query form is returned directly.

A sample *AustWines* query form follows:

```
Content-type: text/html

<HTML>
<HEAD>
<TITLE>Australian Wines</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#23238E" LINK="#990033" ALINK="#2F2F4F"
   VLINK="#990033">
<IMG HSPACE=50 ALIGN=RIGHT SRC="/ke/images/kelogo/kehome.gif">
<H2>
<CENTER>
<IMG HSPACE="8" SRC="/ke/images/texpress.gif">
Australian Wines
</CENTER>
</H2>
<H4><CENTER><A NAME="Top">KE Software Pty Ltd</A></CENTER></H4>
<P>
<HR>
<P>
<TABLE>
<TR><TD><IMG HSPACE=8 SRC="/ke/images/demo/wingrape.gif"></TD>
<TD>
<STRONG>KE Texpress database of 105 records</STRONG>
<P>
Enter your search terms or request
<A HREF="/ke/demohelp.html"><STRONG>help</STRONG></A>.
</TD>
</TR>
</TABLE>
<HR>

<TABLE COLSPEC="L20 L20" WIDTH=100%>
<TR>
<TD>
<FORM ACTION="http:/cgi-bin/texhtml" METHOD="POST">
<INPUT TYPE="hidden" NAME="form" VALUE="AustWines.sum">

<B>Wine Name</B><BR>
<INPUT TYPE="text" NAME="winename" SIZE="40" VALUE="">
<P>

<B>Wine Style</B><BR>
<INPUT TYPE="text" NAME="style" SIZE="40" VALUE="">
```

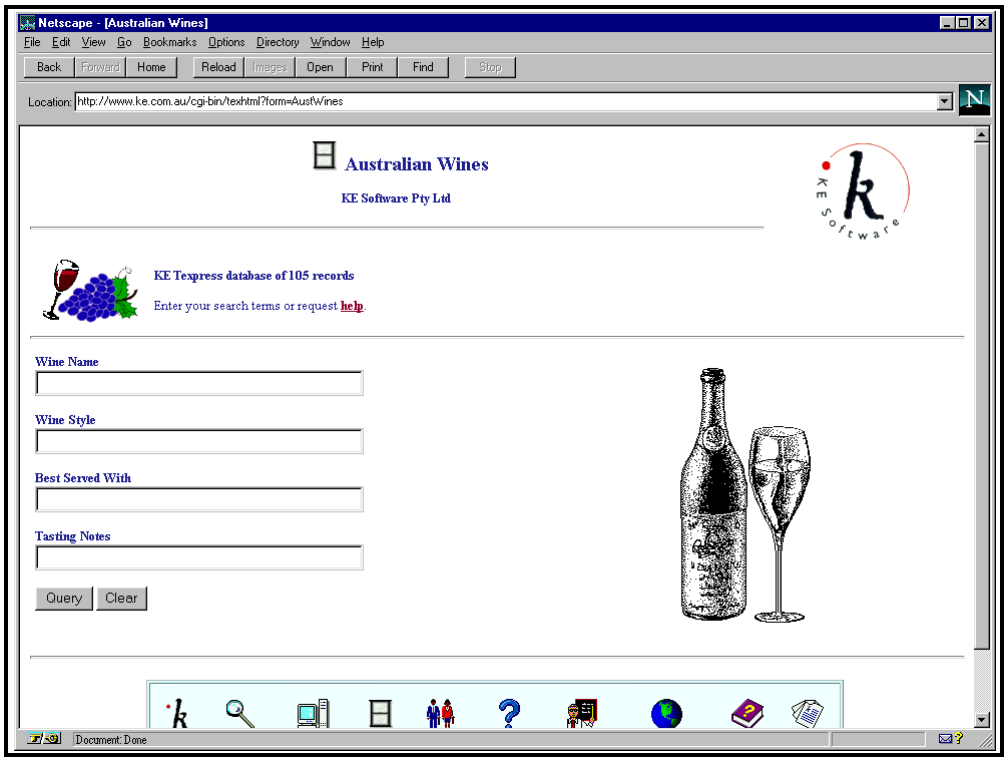
```
<P>

<B>Best Served With</B><BR>
<INPUT TYPE="text" NAME="bestserv" SIZE="40" VALUE="">
<P>

<B>Tasting Notes</B><BR>
<INPUT TYPE="text" NAME="tastingn" SIZE="40" VALUE="">
<P>
<INPUT TYPE=submit VALUE=Query>
<INPUT TYPE=reset VALUE=Clear>
</FORM>
</TD>
<TD>
<IMG SRC="/ke/images/demo/wine.gif">
</TD>
</TR>
</TABLE>

<!---This is the toolbar imagemap--->
<P>
<HR>
<P>
<CENTER>
<IMG BORDER=0 SRC="/ke/toolbar/toolbar.gif" USEMAP="#toolbar">
<MAP NAME="toolbar">
<AREA SHAPE="rect" COORDS="3,4, 59,100" HREF="/index.html">
<AREA SHAPE="rect" COORDS="60,4, 124,100"
   HREF="/cgi-bin/texhtml?form=WebSearch">
<AREA SHAPE="rect" COORDS="125,4, 202,100" HREF="/ke/products.html">
<AREA SHAPE="rect" COORDS="203,4, 264,100" HREF="/ke/demo.html">
<AREA SHAPE="rect" COORDS="265,4, 328,100" HREF="/ke/clients.html">
<AREA SHAPE="rect" COORDS="329,4, 400,100" HREF="/ke/support.html">
<AREA SHAPE="rect" COORDS="401,4, 474,100" HREF="/ke/courses.html">
<AREA SHAPE="rect" COORDS="475,4, 564,100" HREF="/ke/distrib.html">
<AREA SHAPE="rect" COORDS="565,4, 630,100" HREF="/ke/guides/index.html">
<AREA SHAPE="rect" COORDS="631,4, 692,100" HREF="/ke/promo.html">
<AREA SHAPE="rect" COORDS="0,0, 695,133" HREF="/index.html">
</MAP>
</CENTER>
<!---End of toolbar imagemap--->

</BODY>
</HTML>
```

This contains references to a series of images such as *kelogo.gif* which are stored on the Web Server. It makes use of HTML tables to assist in alignment of the images and text and includes a site-wide image map at the end of the form.

The document also defines an HTML form with input boxes for *winename*, *style*, *bestserv* and *tastingn*. Although these names are merely labels used when the query is submitted, for convenience they have been defined to be the same as the column Ids within Texql.

The form contains the hidden input box called *form*. When the *Submit* button is selected, the argument *form=AustWines.sum* is submitted as the first argument to *texhtml*.

This form is displayed in Netscape as below:



***AustWines* Query Form**

This form contains input boxes into which the user can enter any values. When the user selects the submit button (lower left region of the form), the form action is performed. This is a *URL* to run *texhtml*. Any data in the input boxes, including the hidden input box, are appended to the *URL* in the form ***label=value***.

The HTML Display Forms

When invoked with more than one argument, KE Texhtml consults its *config* file to determine the appropriate Database Server (*kestrel*) for the document template name supplied (**AustWines.sum**). It then establishes a connection to *texhtmlserver* on *kestrel* and passes to it that template name, the remaining arguments and the environment.

texhtmlserver then searches for the file **scripts/AustWines.sum** in its *config* directory. If this file is executable, it is executed with the arguments passed to *texhtmlserver* passed to it. An example Unix shell script which would generate a suitable *Texql* statement follows:

```
:
cat <<EOF
select    all
from      AustWines
where     true
EOF
for i
do
   name=`echo $i | sed 's/=.*$//'`
   val=`echo $i | sed 's/^[^=]*=//'`
   if test "${name}" = "bestserv"
   then
                echo "and    exists(${name}_tab where
```

```
                    ${name} contains '${val}')"
   else
                echo "and    ${name} contains '${val}'"
   fi
done
echo "$REMOTE_HOST `date`" >>/tmp/connections
```

If the file is not executable, then the substitutions of the on-line arguments are made and the query is collapsed, as described in Chapter 3. An example of such a file follows:

```
select  all
from    AustWines
where   true
and     winename contains '#winename#'
and     style contains '#style#'
and     tastingn contains '#tastingn#'
and     exists(bestserv_tab where bestserv contains '#bestserv#')
```

If the user were to enter the single term, **brilliant**, into the Tasting Notes input box, both of the above methods would result in the Texql statement:

```
select   all
from     AustWines
where    tastingn contains 'brilliant'
```

The first method has the side effect of recording the date and connecting host name in the file, */tmp/connections*.

texhtmlserver performs the query and then assembles the results using the three files, **headers/AustWines.sum**, **bodies/AustWines.sum** and **footers/AustWines.sum**. Examples of these files follow:

headers/AustWines.sum

```
Content-type: text/html

<HTML>

<HEAD>
<TITLE>Australian Wines</TITLE>
</HEAD>
<BODY BGCOLOR="FFFFFF">
<IMG HSPACE=50 ALIGN=RIGHT SRC="/ke/images/kelogo/kehome.gif">
<CENTER>
<H2>
<IMG HSPACE="8" SRC="/ke/images/texpress.gif">
Australian Wines
</H2>
<H4>
<A NAME="Top">KE Software Pty Ltd</A>
</H4>
</CENTER>
<HR>
<IMG SRC="/ke/images/demo/vineline.gif">
<IMG HSPACE=15 SRC="/ke/images/demo/redwine.gif">
<BR>
<CENTER>
<I>No. of hits: #TexRowCnt#</I>
<HR>
</CENTER>
```

bodies/AustWines.sum

```
<IMG SRC="/ke/images/demo/bottl.gif">
<A HREF="/cgi-bin/texhtml?form=AustWines.all&wineno=#!wineno#">
#+winename#
</A>
<BR>
```

footers/AustWines.sum

```
<!---This is the query/clear imagemap--->
```

```
<P>
<HR>
<IMG BORDER=0 SRC="/ke/images/buttons/database.gif" USEMAP="#querymap">
<MAP NAME="querymap">
<AREA SHAPE="rect" COORDS="0,0, 110,28"
   HREF="/cgi-bin/texhtml?form=AustWines">
<AREA SHAPE="rect" COORDS="110,0, 192,28" HREF="/ke/demo.html">
<AREA SHAPE="rect" COORDS="0,0, 192,28" HREF="/ke/demo.html">
</MAP>

<!---This is the toolbar imagemap--->
<P><HR><P>
<CENTER>
<IMG BORDER=0 SRC="/ke/toolbar/toolbar.gif" USEMAP="#toolbar">
<MAP NAME="toolbar">
<AREA SHAPE="rect" COORDS="3,4, 59,100" HREF="/index.html">
<AREA SHAPE="rect" COORDS="60,4, 124,100"
   HREF="/cgi-bin/texhtml?form=WebSearch">
<AREA SHAPE="rect" COORDS="125,4, 202,100" HREF="/ke/products.html">
<AREA SHAPE="rect" COORDS="203,4, 264,100" HREF="/ke/demo.html">
<AREA SHAPE="rect" COORDS="265,4, 328,100" HREF="/ke/clients.html">
<AREA SHAPE="rect" COORDS="329,4, 400,100" HREF="/ke/support.html">
<AREA SHAPE="rect" COORDS="401,4, 474,100" HREF="/ke/courses.html">
<AREA SHAPE="rect" COORDS="475,4, 564,100" HREF="/ke/distrib.html">
<AREA SHAPE="rect" COORDS="565,4, 630,100" HREF="/ke/guides/index.html">
<AREA SHAPE="rect" COORDS="631,4, 692,100" HREF="/ke/promo.html">
<AREA SHAPE="rect" COORDS="0,0, 695,133" HREF="/index.html">
</MAP>
</CENTER>
<!---End of toolbar imagemap--->

</BODY>
</HTML>
```

This creates a summary of the matching wines. It is an HTML document containing a marked up list of wine names, behind each of which is a *URL* referring to the *docname*, **AustWines.all**, and using the query, **wineno=*num***. Note that, as the *wineno* field is included in a *URL*, it is transformed via the URL operator.

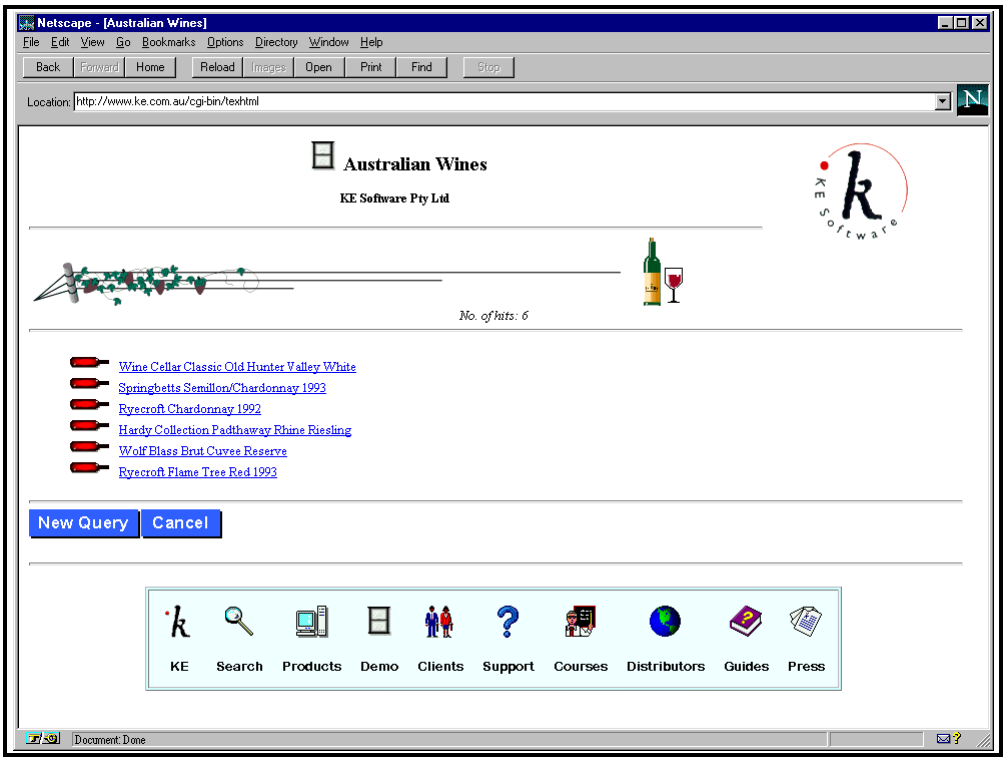
The footer contains an image, behind which is a *URL* to display the *AustWines* HTML query form described in the previous section.

If the file **filters/AustWines.sum** exists, or failing this the file **filters/default**, the derived HTML document is passed through this filter and returned to the Internet client. This example uses a simple filter to convert the matching term markers into HTML bold markers.

filters/AustWines.sum

```
:
sed '
s#<!-- MATCH ON -->#<B>#g
s#<!-- MATCH OFF -->#</B>#g
'
```

An example summary document, for the query of **brilliant** in the *Tasting Notes*, is displayed in Netscape as follows:



AustWines Summary Document

Behind each list entry is a *URL* which submits a different query to KE Texhtml, in this case using the Wine Number to extract extensive information about an individual wine. The *URL* is of the form:

```
http:/cgi-bin/texhtml?form=AustWines.all&wineno="num"
```

This uses the **AustWines.all** Texql query and HTML document template set.

The *AustWines.all* Texql query, stored in **scripts/AustWines.all**, is as follows:

```
wines
where   true
and     wineno = #wineno#
```

The HTML document template set is as follows:

headers/AustWines.all

```
Content-type: text/html

<HTML>

<HEAD>
<TITLE>Australian Wines</TITLE>
</HEAD>
<BODY BGCOLOR="FFFFFF">
<IMG HSPACE=50 ALIGN=RIGHT SRC="/ke/images/kelogo/kehome.gif">
<CENTER>
<H2>
<IMG HSPACE="8" SRC="/ke/images/texpress.gif">
Australian Wines
</H2>
<H4>
<A NAME="Top">KE Software Pty Ltd</A>
</H4>
```

```
</CENTER>
<HR>
<IMG SRC="/ke/images/demo/vineline.gif">
<IMG HSPACE=15 SRC="/ke/images/demo/redwine.gif"><BR>
<CENTER>
<I>No. of hits: #TexRowCnt#</I>
</CENTER>
<HR>
```

bodies/AustWines.all

```
<TABLE CELLSPACING=10 COLSPEC="L20 L20" WIDTH=100%>
<TR>
<TD><IMG SRC="/ke/images/demo/vine.gif"></TD>
<TD>
<H3><B>#winename#</B></H3>

#type#. #style#.
Best in <B>#bestin#</B>.
Price per bottle <B>$#priceper#</B>.<P>

<B>Best served with:</B><BR>
#bestserv_tab#
<P>

<B>Tasting Notes:</B><BR>
#tastingn#
<P>

<IMG SRC="/Images/AustWines/#frontima#.gif" ALT="Wine Label (front)">
<IMG SRC="/Images/AustWines/#backimag#.gif" ALT="Wine Label (back)">

</TD>
</TR>
</TABLE>
```

footers/AustWines.all

```
<!---This is the query/clear imagemap--->
<P>
<HR>
<IMG BORDER=0 SRC="/ke/images/buttons/database.gif" USEMAP="#querymap">
<MAP NAME="querymap">
<AREA SHAPE="rect" COORDS="0,0, 110,28"
   HREF="/cgi-bin/texhtml?form=AustWines">
<AREA SHAPE="rect" COORDS="110,0, 192,28" HREF="/ke/demo.html">
<AREA SHAPE="rect" COORDS="0,0, 192,28" HREF="/ke/demo.html">
</MAP>

<!---This is the toolbar imagemap--->
<P><HR><P>
<CENTER>
<IMG BORDER=0 SRC="/ke/toolbar/toolbar.gif" USEMAP="#toolbar">
<MAP NAME="toolbar">
<AREA SHAPE="rect" COORDS="3,4, 59,100" HREF="/index.html">
<AREA SHAPE="rect" COORDS="60,4, 124,100"
   HREF="/cgi-bin/texhtml?form=WebSearch">
<AREA SHAPE="rect" COORDS="125,4, 202,100" HREF="/ke/products.html">
<AREA SHAPE="rect" COORDS="203,4, 264,100" HREF="/ke/demo.html">
<AREA SHAPE="rect" COORDS="265,4, 328,100" HREF="/ke/clients.html">
<AREA SHAPE="rect" COORDS="329,4, 400,100" HREF="/ke/support.html">
<AREA SHAPE="rect" COORDS="401,4, 474,100" HREF="/ke/courses.html">
<AREA SHAPE="rect" COORDS="475,4, 564,100" HREF="/ke/distrib.html">
<AREA SHAPE="rect" COORDS="565,4, 630,100" HREF="/ke/guides/index.html">
<AREA SHAPE="rect" COORDS="631,4, 692,100" HREF="/ke/promo.html">
<AREA SHAPE="rect" COORDS="0,0, 695,133" HREF="/index.html">
</MAP>
</CENTER>
<!---End of toolbar imagemap--->

</BODY>
</HTML>
```

The document includes HTML markers for images, the names of which are determined from the database (*#frontima#* and *#backimag#*). In cases where there is no associated image for a record, it is necessary to remove the marker. To achieve this, the following **filters/AustWines.all** script is used:

```
:
```

```
sed '
/\/.gif/d
'
```

This is a simple *sed* script to remove lines where no image name precedes the *.gif* file name extension.

An example detail document is displayed in Netscape as follows:

AustWines Detail Document

Chapter 5

KE Texhtml Administrative Support

Overview

This chapter describes a variety of administrative support facilities available with KE Texhtml. These include auditing, security and error reporting. There is also a section describing debugging techniques.

KE Texhtml Auditing

KE Texhtml is capable of maintaining comprehensive audit trails of all operations performed by *texhtmlserver*. This information can be held in either a database or a file.

Audit records include:

- The current date and time.
- The requesting user's host name.
- The User Id of the user, if available. If *texhtml* is subject to server security (refer to the administration manual for your *httpd* software), then the browser asks the user for a User Id and password which are validated by the server. The User Id then becomes part of the *texhtmlserver* environment, called **REMOTE_USER** and is included in the audit record for the transaction.
- The CPU time (user and system time) taken to service the request.
- The type of the request - one of **Query form**, **Query statement**, **Row insertion**, **Row update** or **Row deletion**.
- The number of rows matching the query or affected by the insert/update/delete statement.
- The number of bytes of information produced by *texhtmlserver* in servicing the request. This counts the number of bytes fed to the filter, if any, not the number of bytes ultimately produced by the filter.
- The status of the request, one of **Completed**, **Command interrupted** (indicating that the user terminated the request during the search) or **Results interrupted** (indicating that the user terminated the request during the downloading of results).
- The name of the Web server (host of *httpd*). This can be useful for *multi-homing* hosts - hosts responding to different Internet addresses.
- The service name on which the request was processed, e.g. *texhtml* or *texhtmls*
- All of the arguments passed to *texhtmlserver*, including the form or document template name.

If the database, **htmlaudit**, exists and *texhtmlserver* can insert records into it, then it adds a record to this database consisting of the above information.

If *texhtmlserver* cannot insert into this database, it places a copy of the insert statement, together with the error message generated into the file

admin/htmlaudit.err, if it can write to the file. It then attempts to write the audit information, in colon separated form, into the file, **admin/audit**.

To disable auditing, simply do not install the *htmlaudit* database or close the database and set the file permissions on **admin/audit** to read-only.

KE Texhtml Security

Secure Access

Web server software supports a secure access facility. This enables a server administrator to designate a file or all files in a directory to be subject to secure access. When Internet users attempt to access such files, they are requested to supply a valid User Id and password which is authenticated on the Web server by the *httpd* process. The user Id supplied is then made available as the environment variable, **REMOTE_USER**.

Many sites use of this feature by creating an additional script directory called, say, **cgi-secure**. The *texhtml* program is linked into this directory using a different name, such as *texhtmls*, thus creating a new service under this name. Note that this service must be registered in the **/etc/services** file with a unique service number, say, **5983**. An *texhtmlserver* can be launched on this service using the command:

```
texhtmlstart texhtmls -n5
```

This example starts five servers on this service.

Alternatively or in addition to the above, scripts and filters can make use of the *REMOTE_USER* environment variable to implement additional security, e.g. group-based security on a document database.

User Id

texhtmlserver is invoked running as a particular user Id, often **www** or **daemon**. This account must be given appropriate access privileges to any databases which are to be accessed via the Internet.

Note that all Internet user requests are processed under this account on the database server even if the request is via a secure server where the *REMOTE_USER* environment variable is available.

Providing Access to the Database Server from the Web Server Only

The **admin/config** file on the Web server machine indicates to *texhtml* which Database server is to service the request. However, by default, a Database server accepts connections from any Web server. This can compromise the *httpd* security feature which is based on the Web server as another Web server without such security could connect directly to the database server with an appropriate **admin/config** file.

To address this problem, *texhtmlserver* supports the argument:

```
-hhost1:host2:...
```

This ensures that the Database server will only accept requests from *texhtml* processes on ***host1***, ***host2***, etc.

Should an attempt is made to connect to a *texhtmlserver* process from an invalid host, then the request is refused and an entry including the date, time and an error message indicating the connecting host is added to the **admin/warnings** file.

KE Texhtml Error Reporting Facilities

Refused Requests

Many sites run multiple *texhtmlserver* processes on each service. Each process can service one request simultaneously. Additional simultaneous requests are queued (to a depth of up to the number of *texhtmlserver* processes running). Any further requests are rejected with the user given an HTML document containing the message:

> **All KE Texhtml Database Servers are busy at the moment.**
> **Please try again later or contact the server site administrator.**

The refusal is logged in the file, **admin/refused**, together with the date and time and the error message detected.

Error Documents

There are four primary error conditions which are detected by *texhtmlserver*. Each uses a template document in the **errors** directory, substituting the built-in functions as appropriate. These document templates can be changed on site as required.

NoFormName
Used if no **form=*docname*** argument is supplied.

NoQueryForm
Used if the request is for a query form but the query form document does not exist.

BadFormName
Used if the request is for a Texql statement to be generated and performed but no **script/*docname*** exists.

Command
Used if Texql statement fails.

Debugging Techniques

This section discusses a common technique for debugging a KE Texhtml document template set. It considers only the process of deriving an HTML document and populating it from one or more databases. It does not consider the development of query forms, which are generally just HTML documents.

Script Development

While a non-executable script can be used for producing the Texql statement, many developers prefer to use a program such as a shell or perl script to create their Texql statement. Such a script can be run manually, passing to it a range of arguments in the form:

```
arg=value
```

corresponding to input field names and their associated values. (Note that empty fields need not be specified as KE Texhtml automatically removes them before running the script.)

The output of the script can be redirected into a file. This file should contain a valid Texql statement. The most effective way to test a Texql statement is using the *texql* command interpreter. Simply add:

```
:
texql <<EOF
```

to the beginning of the file and:

```
;
EOF
```

to the end. Then run test the statement using a command such as:

```
sh file
```

For command errors, this will report the position within the command at which the error was detected. Otherwise it will produce the output of the command as appropriate.

To verify the structure of the derived table (the statement's output), simply insert the keyword, **describe**, in this file immediately before the Texql statement.

Template Development

When the Texql statement appears to be working correctly, you can test the HTML document templates (header, body and footer documents). First ensure that no filter exists or that the filter is not executable. Then test the script and HTML document templates using a Unix command of the form:

```
texhtml form=docname arg=value ...
```

This performs the Texql statement produced by the script and outputs the result via the appropriate header, body and footer document templates. Check the output for errors, including sequences such as:

```
no such column
```

which indicates that one of the column references refers to an invalid column identifier.

Filter Development

When the document templates appear to be working correctly, repeat the above *texhtml* command, redirecting its output into a file. Test the filter by running it directly from the Unix prompt and taking its input from this file. Any errors in the filter, such as syntax errors will be reported directly to the screen (stderr). When the filter is run in conjunction with *texhtml*, these errors are redirected into the file, **admin/server.out**.

Finally, install the filter in the appropriate location and make it executable. Then repeat the above *texhtml* command to ensure that the correct HTML document is produced. After this step, the whole sequence is ready to be tested from your browser.

Index

/

A

B

C

D

E

F

G

H

I

K

L

M

N

P

Q

R

S

T

U

W