

BANGALORE INSTITUTE OF TECHNOLOGY

Affiliated to Visvesvaraya Technological University

K.R.Road, V.V.Puram, Bangalore-560004

**Department of
INFORMATION SCIENCE & ENGINEERING**

**CGI Programming Laboratory
2007-2008**

CGI LABORATORY

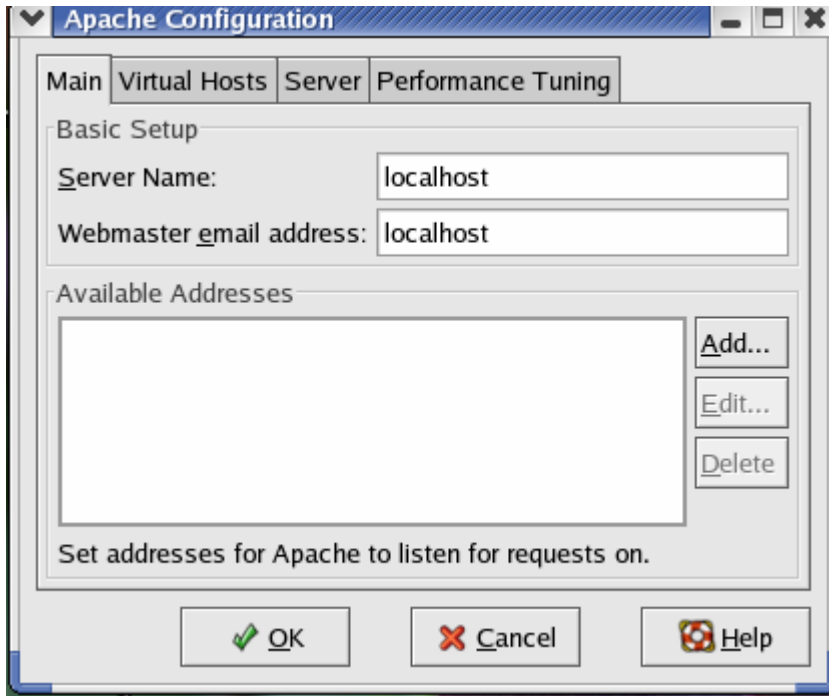
B S SANTOSH
Dept of ISE
BIT, Bangalore
2007-2008



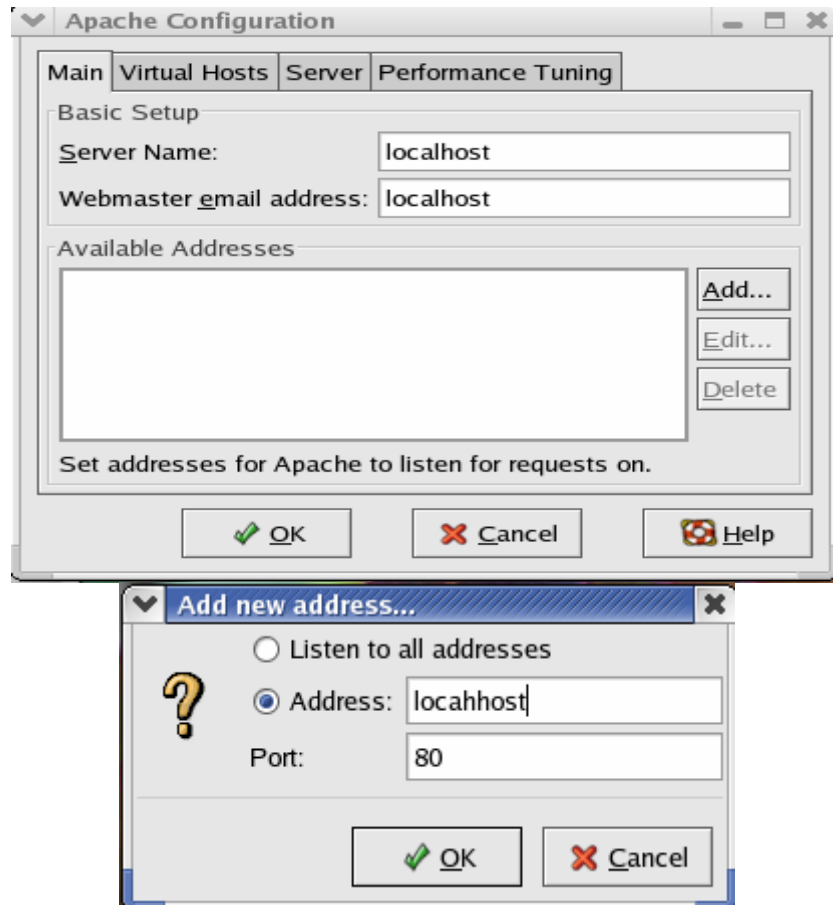
1. Apache Server Configuration

The configuration is to be done on Linux 2.6.9 kernel or any other Linux platform.

1.1 Run HTTP Server by Going to **Start→System Settings→Server Settings→http Server**. Set Server Name=localhost and webmaster email address=localhost as shown in the below snap-shot.



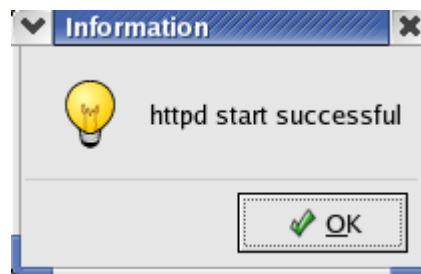
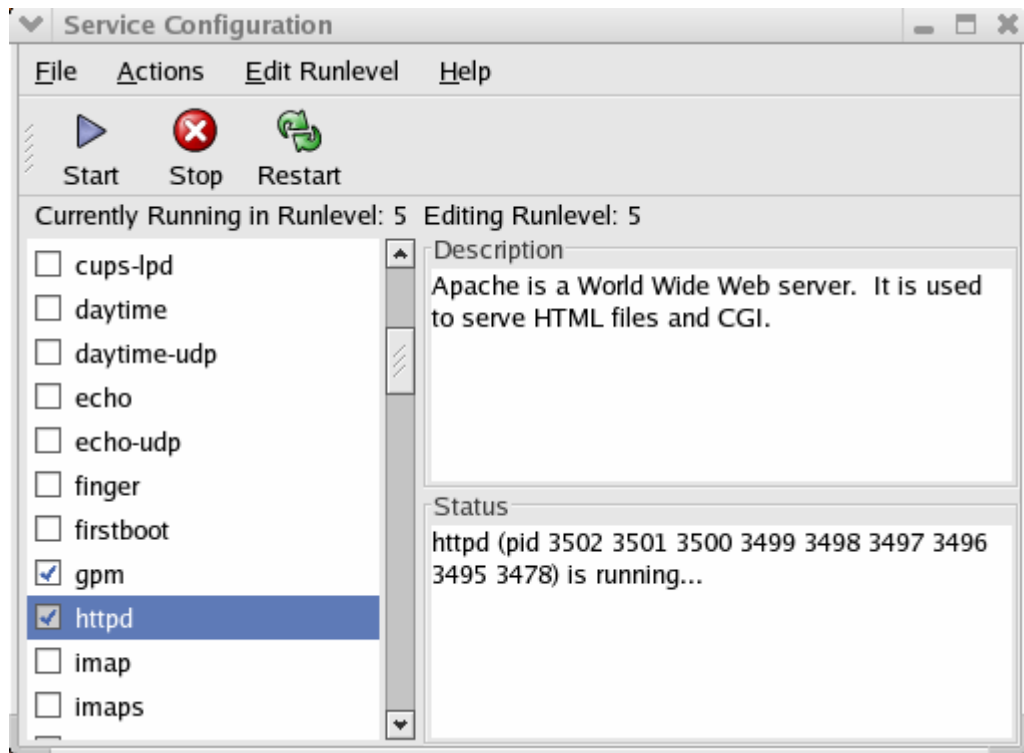
1.2 Click on the Add... button



Set Address to localhost and port number to 80 as show in the above snap-shot.

2. Running Apache Web Server

2.1 To run Apache Web Server...Start→System Settings→Server Settings→Services
Select httpd from the list and then press Start as shown in the below Snap-Shot.



3. Where to Keep HTML files and Perl scripts

3.1 HTML FILES:

Place all html files in the following Directory.

/var/www/html

3.2 PERL SCRIPTS:

Place all Perl scripts in the following Directory.

/var/www/cgi-bin

3.3 HEADER FILE (cgi-lib.pl):

Cgi-lib.pl is a header script. This script is required for some of the Perl scripts when it is necessary to send some information to the Perl script along with the request for the script. This header contains the some built in functions, for example **&readparse()** is a function defined in the header which can access the data specified in the text boxes of the web page. So all the Perl files which access the data specified in the text boxes of the web page should include the above said function. This is done by using the 'require' statement as `require cgi-lib.pl`.

WHERE TO PLACE THE CGI-LIB.PL:

Place this Perl script also in the same directory where the entire Perl files are placed. i.e. in `/var/www/cgi-bin`

This copy this file form `/bitserver/downloads/cgi-lib/cgi-lib.pl`

3.4 ERROR CHECKING:

To check the type of Error in the Perl Script open the `error_log` file present within the following directory.

`/var/log/httpd/error_log`

4. Using Mysql

In case of problems connected database, please adopt the following steps:

To check the path of Mysql edit the following command in the terminal.

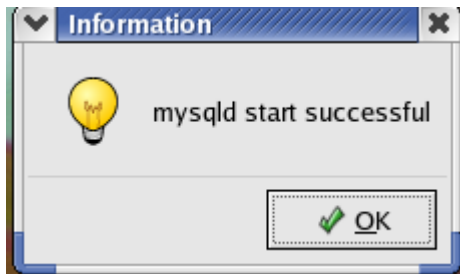
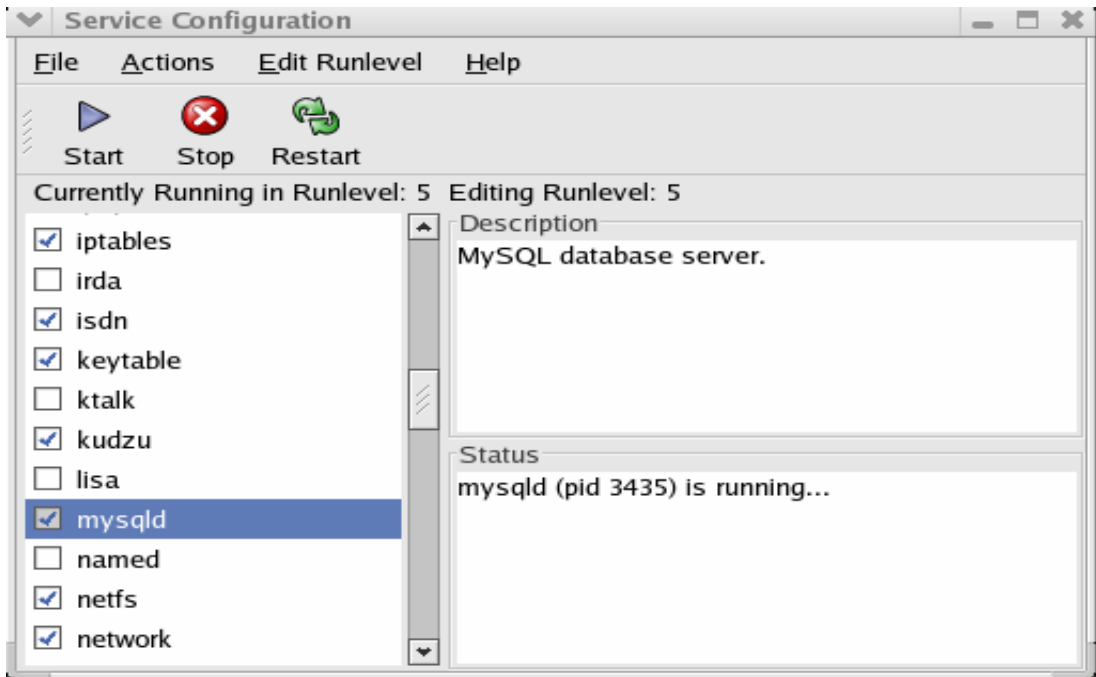
```
$whereis mysql
```

```
Mysql: /usr/bin/mysql
```

If mysql is already installed on your machine then you will get the path as shown above. Otherwise mysql has to be installed.

4.1 TO RUN MYSQL DATABASE SERVER:

Goto: Start→System Settings→Server settings→Services



Select **mysqld** and then press **Start** as shown in the above Snap-shot.

After starting the mysql database server go to the terminal and enter into the following directory.

```
[root@localhost root]#cd /usr
```

```
[root@localhost usr]#cd bin
```

```
[root@localhost bin]# mysql
```

```
Mysql>
```

After entering into the mysql one has to create his own account.

4.3 CREATING THE NEW USER ACCOUNT:

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'some_user_name'@'localhost'  
-> IDENTIFIED BY 'some_pass' WITH GRANT OPTION;
```

```
mysql>exit
```

The above command will create the new account with some user specified username and password. After creating the new account exit from mysql using the exit command.

4.4 LOGGING TO NEW USER ACCOUNT:

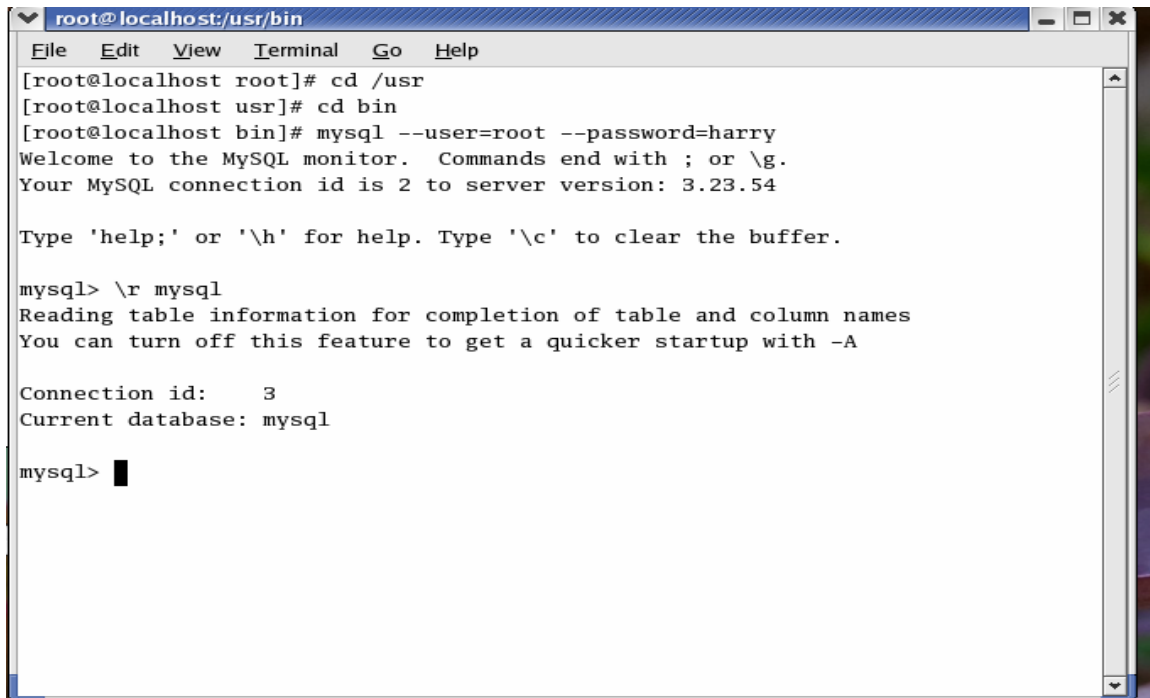
First run the mysql database server as shown already in the section 4.1. Then log onto your account using user name and password as shown below.

```
[root@localhost bin]# mysql - - user="username" - - password="password"␣
```

```
Mysql> \r mysql␣
```

```
Mysql>
```

The following Snap-Shot shows the process of logging to user account.



```
root@localhost:/usr/bin
File Edit View Terminal Go Help
[root@localhost root]# cd /usr
[root@localhost usr]# cd bin
[root@localhost bin]# mysql --user=root --password=harry
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 3.23.54

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> \r mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Connection id:      3
Current database:  mysql

mysql> █
```

1. Configuring Apache Web Server to Execute PHP Server Scripts

1.1 Follow the following steps:

1. Open your Apache httpd.conf configuration file etc/httpd/conf/httpd.conf on RedHat Linux) in your favorite text editor. Look for lines like the following:

```
LoadModule php4_module lib/apache/libphp4.so
```

```
LoadModule php4_module modules/libphp4.so
```

If these lines are there already, then do not make any changes. If these lines are not found in httpd.conf then add these two lines as it is under any other already existing LoadModule lines without modifying existing ones.

2. Next, look for the line starting with DirectoryIndex. This line tells Apache what filenames to use when looking for the default page for a given directory. You'll see the

usual index.html and so forth, but you need to add index.php and index.php3 to that list as shown below:

```
DirectoryIndex index.html index.cgi ... index.php index.php3
```

3. Finally, go right to the bottom of the file and add the following line to tell Apache what file extensions should be seen as PHP files:

```
AddType application/x-httpd-php .phtml .php .php3
```

4. Save the Changes made to the apache httpd.conf file and restart the apache web server. Apache web server should start without any error messages.

1.2 About PHP:

PHP is a server-side scripting language. The concept of php is very similar to the JavaScripts or VBScripts. PHP server-side scripting language is similar to JavaScript in many ways, as they both allow you to embed little programs (scripts) into the HTML of a Web page.

1.2.1 The key difference between JavaScript and PHP:

The key difference between JavaScript and PHP is that, while the Web browser interprets JavaScript once the Web page containing the script has been downloaded, server-side scripting languages like PHP are interpreted by the Web server before the page is even sent to the browser. Once interpreted, the PHP code is replaced in the Web page by the results of the script, so all the browser sees is a standard HTML file. The script is processed entirely by the server. Thus the designation: server-side scripting language.

1.2.2 HOW TO EMBED PHP INTO HTML:

Let's look at the example today.php shown below.

```
<HTML>
<HEAD>
<TITLE>Today's Date</TITLE>
</HEAD>
<BODY>
<P>Today's Date (according to this Web server) is
<?php
    echo( date("l, F dS Y." ) );
?>
</BODY>
</HTML>
```

The above program shows the PHP code embedded in the HTML. Lines between <?php and ?> indicates the php code.

1. <?php means “begin php code”.
2. ?> means “end php code”.

The Web server is asked to interpret everything between these two delimiters (<?php and ?>) and convert it to regular HTML code before sending the Web page to a browser that requests it. The browser is presented with something like this:

```
<HTML>
<HEAD>
<TITLE>Today's Date</TITLE>
</HEAD>
<BODY>
<P>Today's Date (according to this Web server) is
```

Wednesday, June 7th 2000.</BODY>

</HTML>

1.2.3 WHERE TO KEEP PHP FILES:

Keep all the php files in the following directory by saving them with .php extension.

/var/www/html

1.2.4 HOW TO EXECUTE:

Go to the browser and give <http://localhost/today.php>

SERVLETS

It is assumed that the student knows the basics of Java to understand the Lab programs. The details of the Java Programming language is not explained here.

Before we start with the Lab programs, we need to know what Servlets are. Java Servlets are small programs running on the server side that dynamically extends the functionality of a web server. In the earlier days, for every client request, a separate process was created by the server to handle it. Each process opens a connection to the database to get the latest information about it. These processes interacted with the web server via the Common Gateway Interface (CGI). They are computationally expensive and are generally implemented using C, C++, and Perl.

However, Servlets execute in the same address space of the web server.

Life Cycle of a Servlet: -

- 1) init () – Loads Servlet into the Server’s address Space.
- 2) service () - Services requests and waits for further requests from the clients.
- 3) destroy () – Servlet removed from the Address Space.

When we write Java Servlets, it should extend either the “**GenericServlet**” or the “**HttpServlet**” class.

A GenericServlet class implements the basic life cycle methods of a Servlet. An HttpServlet class works well with HTTP Requests and Responses.

The packages that provide you servlet implementation are **javax.servlet.*** And **javax.servlet.http.***.

How to make the servlet work?

To develop and run servlets both jdk1.5 and jsdk2.0 should be installed in the system.

1. Create servlet source code file using any text editor and save it with extension .java under the directory c:/program files/java/jdk1.5.0/bin.
2. Change working directory to c:/program files/java/jdk1.5.0/bin and set the class path to c:/program files/java/jdk1.5.0/lib.
3. Compile the servlet class source file.
4. Copy both source file and class file generated after compilation into c:/jsdk2.0/examples.
5. Place the html code in the same directory or else anywhere, but make sure that you have set path correctly in form tag.
6. Run the html file in browser.
7. Before running html file make sure that, servletrunner.exe is running
8. To run servletrunner.exe type c:\jsdk2.0\bin\servletrunner.exe

HARDWARE REQUIREMENTS

- 1.) **Processor** : Pentium III or more or AMD
- 2.) **RAM** : 16 MB or more
- 3.) **Hard Disk**: 16 GB or more

SOFTWARE REQUIREMENTS

1. **Operating System**: Windows NT/XP or Linux

2. **JSDK2.0 – Java Servlet Development Kit 2.0** – Java tool which provide environment to develop and run servlets.
3. **JDK1.5 – Java Development Kit 1.5** – Standard Java application development kit which enables us to compile and run Java programs.

PART 1

Make sure that you have gone through the Basics of HTML, Perl, MySQL and the Apache Server provided at the beginning of the manual before getting started with the Lab Programs.

STEPS:

- Login as the super user (root)
- Change the directory to cgi-bin
- `$cd /var/www/cgi-bin`

- Type the program in the vi editor and save with a “.pl” extension”
- `$vi 1a.pl`
 - (type the program here)

- Set the execute permission for all users
- `$chmod a+x 1a.pl`

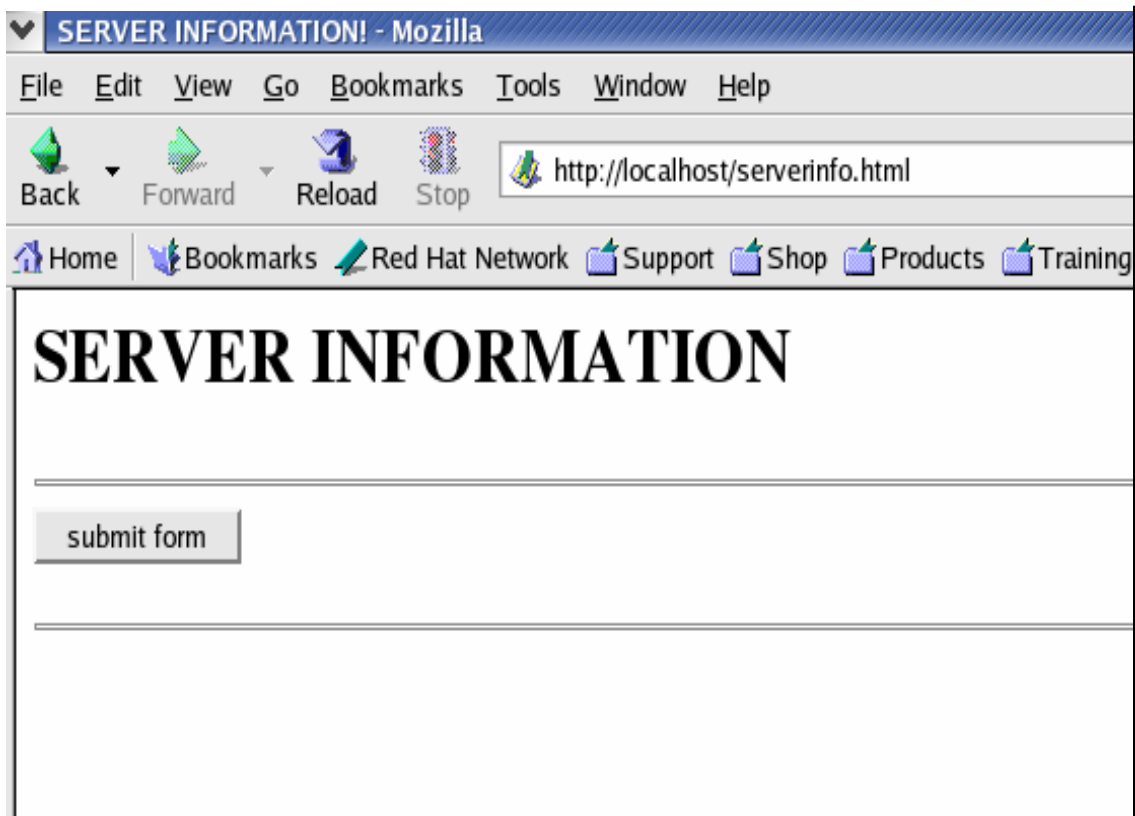
- Start the web server if it is not already started.
- `$/etc/init.d/httpd start`

- Open the Mozilla Web browser and type “http://localhost/cgi-bin/1a.pl”
- The output of the program will be displayed.

1. PROGRAM TO DISPLAY THE VARIOUS SERVER INFORMATION LIKE – SERVER NAME, SERVER SOFTWARE, SERVER PROTOCOL, CGI-REVISION ETC.

HTML FILE:

```
<html>
  <head><title>SERVER INFORMATION!</title></head>
  <body>
    <h1>SERVER INFORMATION</h1>
    <hr>
    <form action="http://localhost/cgi-bin/serinfo.pl" method="get">
    <input type="submit" value="submit form">
    </form>
    <hr>
  </body>
</html>
```



PERL SCRIPT:**Steps: - “1a.pl” –**

- To display the server information, five environment variables, SERVER_NAME, SERVER_PORT, SERVER_SOFTWARE, SERVER_PROTOCOL, and CGI_REVISION are used to print the name of the machine where the server is running, the port the server is running on, the server software, and the CGI revisions.

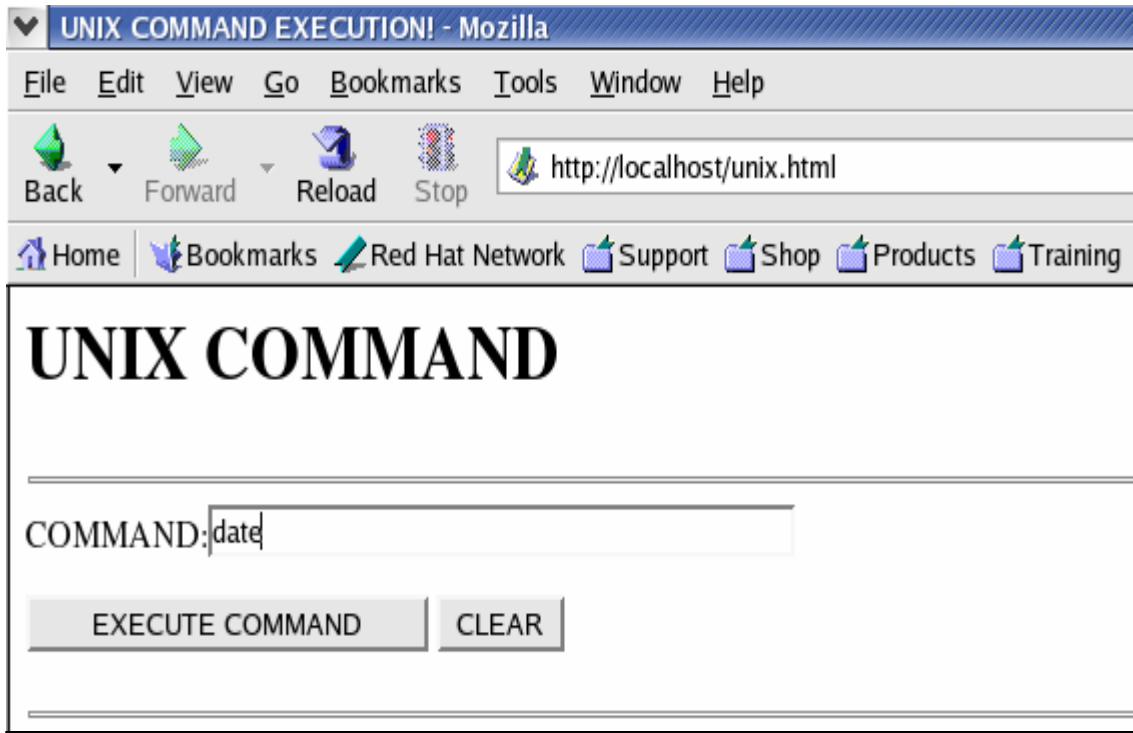
```
#!/usr/bin/perl
require "cgi-lib.pl";
print "content-type: text/html", "\n\n";
print "<HTML>", "\n";
print "<HEAD><TITLE>ABOUT THIS SERVER</TITLE></HEAD>",
"\n";
print "<BODY><H1>ABOUT THIS SERVER</H1>", "\n";
    print "<HR><PRE>";
print "server name:   ", $ENV{'SERVER_NAME'}, "<br>", "\n";
print "Running on the port:   ", $ENV{'SERVER_PORT'}, "<br>", "\n";
print "Server software:      ", $ENV{'SERVER_SOFTWARE'}, "<br>",
"\n";
print "Server protocol:      ", $ENV{'SERVER_PROTOCOL'}, "<br>",
"\n";
print "CGI VERSION:          ", $ENV{'GATEWAY_INTERFACE'},
"<br>", "\n";
print "root document:        ", $ENV{'DOCUMENT_ROOT'}, "<br>",
"\n";
print "<HR></PRE>", "\n";
print "</BODY></HTML>", "\n";
exit(0);
```



1 B. PROGRAM TO ACCEPT UNIX COMMAND FROM HTML FORM AND TO DISPLAY THE OUTPUT OF THE COMMAND EXECUTED.**HTML FILE:**

```
<html>
  <head><title>UNIX COMMAND EXECUTION!</title></head>
  <body>
    <h1>UNIX COMMAND</h1>
    <hr>
    <form action="http://localhost/cgi-bin/unix1.pl" method="get">
      COMMAND :< input type="text" name="command" size=40>

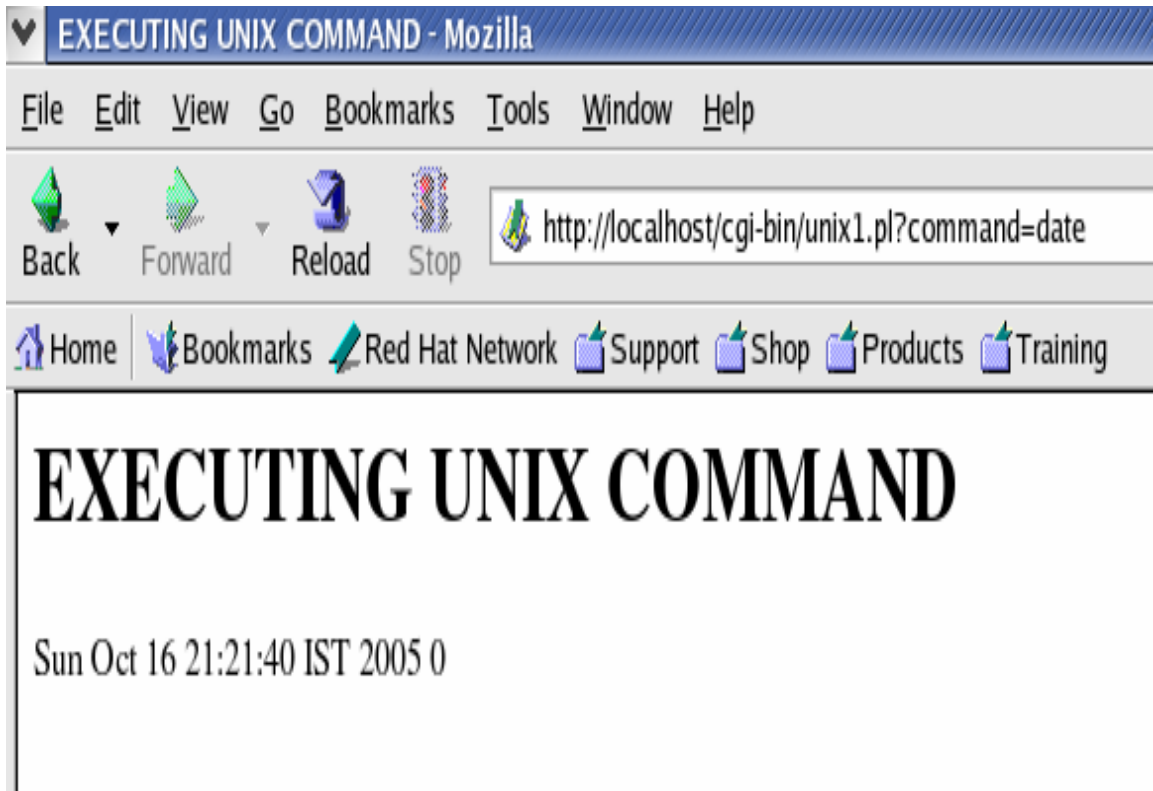
      <input type="submit" value="EXECUTE COMMAND">
      <input type="reset" value="CLEAR">
    </form>
    <hr>
  </body>
</html>
```



PERL FILE:**Steps:”1b.pl”**

- The HTML Page provides a simple interface to accept the UNIX command from the user. After the command is entered, the Perl program is invoked. The command entered by the user is sent via the Query String .
- The Perl program has a “use CGI ‘: standard’”. This “use” pragma ensures that the program has access to all the methods that CGI.pm (Perl Module) provides. The “standard” signifies that only appropriate functions are used for the wide range of browsers. An alternate of this could be “use CGI ‘: all’”
- The param() function is used to capture the parameters that are received from the HTML page. This value is stored in a variable called “\$comm”. To execute the command, the back ticks (`) are used. An alternate approach to this would be to use the system () function.

```
#!/usr/bin/perl
require "cgi-lib.pl";
print "content-type: text/html", "\n\n";
print "<HTML>", "\n";
print "<HEAD><TITLE>EXECUTING UNIX
COMMAND</TITLE></HEAD>", "\n";
print "<BODY><H1>EXECUTING UNIX COMMAND</H1>", "\n";
$query_string=$ENV{'QUERY_STRING'};
($field_name,$command)=split(/=/,$query_string);
$com=system($command);
if($com==-1)
{
    print "command not found! error";
}
else
{
    print $com;
}
print "</body></html>";
exit(0);
```

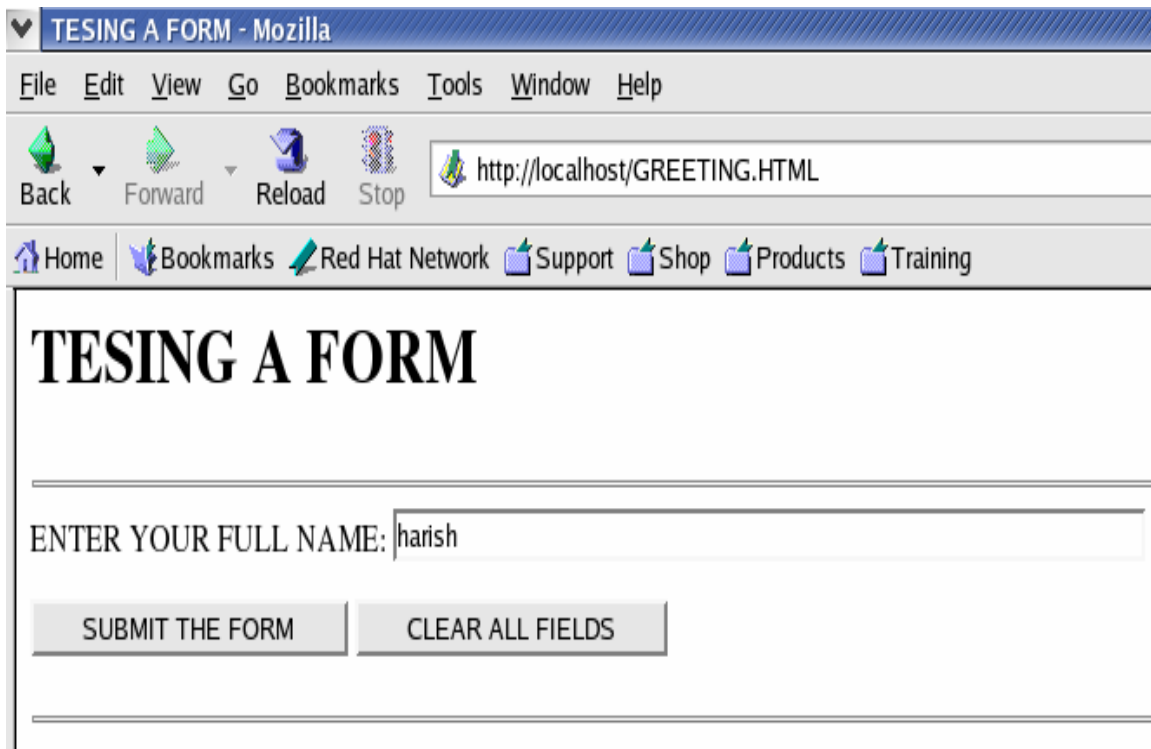


The screenshot shows a Mozilla browser window titled "EXECUTING UNIX COMMAND - Mozilla". The address bar contains the URL "http://localhost/cgi-bin/unix1.pl?command=date". The browser's navigation buttons (Back, Forward, Reload, Stop) are visible. Below the browser window, the main content area displays the output of the CGI script: "EXECUTING UNIX COMMAND" in large, bold, black letters, followed by the date and time "Sun Oct 16 21:21:40 IST 2005 0".

2 A. PROGRAM TO ACCEPT THE USER NAME AND DISPLAY A GREETING MESSAGE.

HTML FILE:

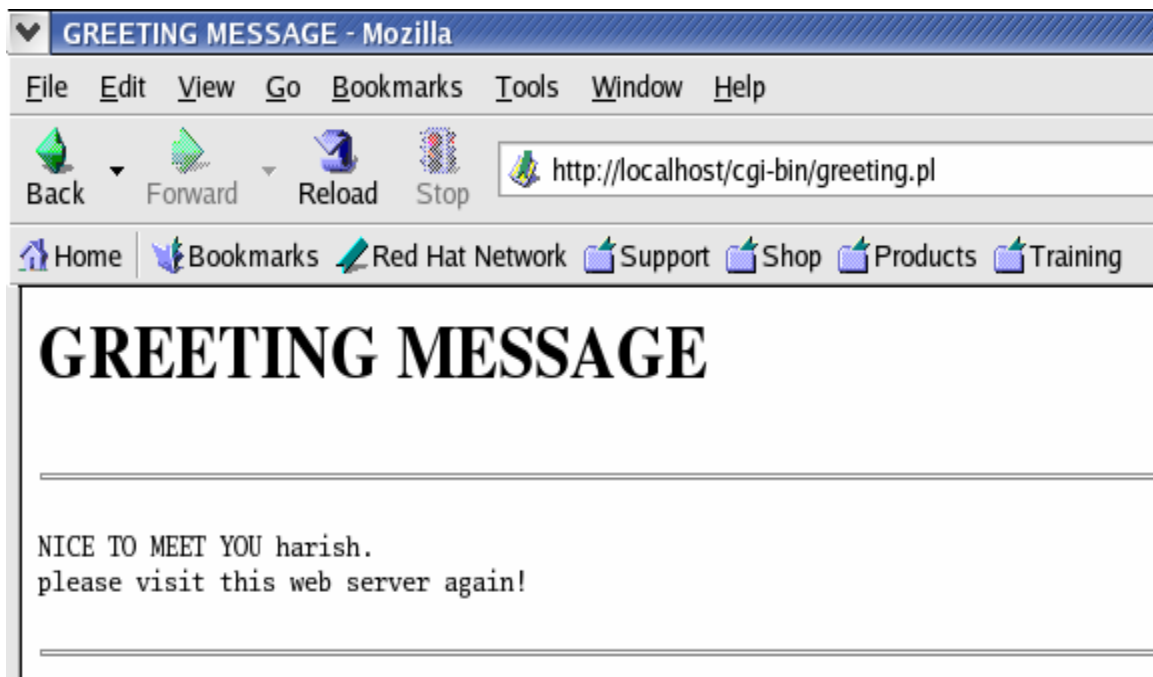
```
<HTML>
  <HEAD><TITLE>TESING A FORM</TITLE></HEAD>
  <BODY>
    <H1>TESING A FORM</H1>
    <HR>
    <FORM ACTION=http://localhost/cgi-bin/greeting.pl
      METHOD="post">
      ENTER YOUR FULL NAME: <INPUT TYPE="TEXT"
        NAME="user" SIZE=60><BR>
      <INPUT TYPE="SUBMIT" VALUE="SUBMIT THE FORM">
      <INPUT TYPE="RESET" VALUE="CLEAR ALL FIELDS">
    </FORM>
    <HR>
  </BODY>
</HTML>
```



PERL SCRIPT:**Steps: “2a.pl”**

- Create html page to enter user name in the form and click on “Submit”.
- The name entered by the user is then used by the Perl program using the param () function. The greeting “Hello <name>” is printed as the output.

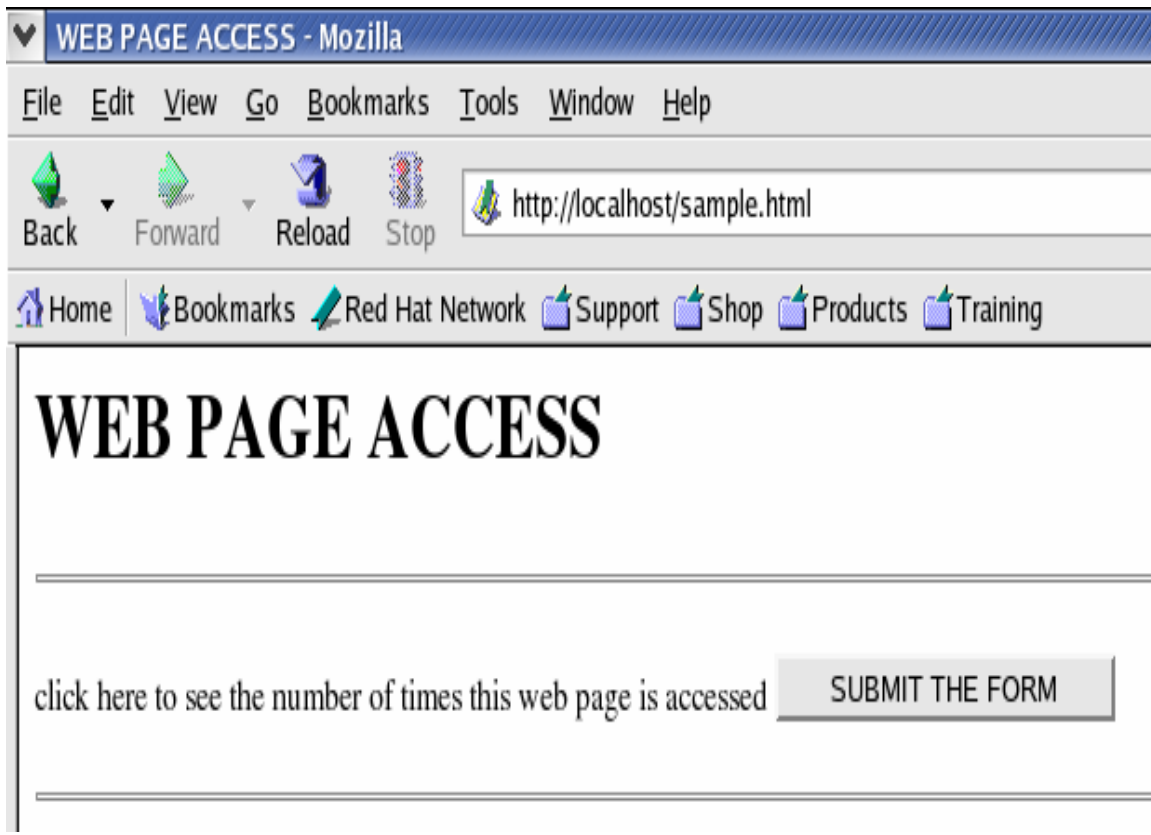
```
#!/usr/bin/perl
require "cgi-lib.pl";
&ReadParse(*simple_form);
$user=$simple_form{'user'};
print "content-type: text/html", "\n\n";
print "<HTML>", "\n";
print "<HEAD><TITLE>GREETING MESSAGE</TITLE></HEAD>", "\n";
print "<BODY><H1>GREETING MESSAGE</H1>", "\n";
print "<HR><PRE>";
print "NICE TO MEET YOU ", $simple_form{'user'}, ". ", "\n";
print "please visit this web server again!", "\n";
print "<HR></PRE>", "\n";
print "</BODY></HTML>", "\n";
exit(0);
```



2 B. PROGRAM TO KEEP TRACK OF THE NUMBER OF VISITORS, VISITED THE WEB PAGE AND DISPLAY THE COUNTER WITH PROPER HEADINGS.

HTML FILE:

```
<HTML>
  <HEAD><TITLE>WEB PAGE ACCESS</TITLE></HEAD>
  <BODY>
    <H1>WEB PAGE ACCESS</H1>
    <HR>
    <form action="http://localhost/cgi-bin/counter.pl" method="post">
      click here to see the number of times this web page is accessed
      <INPUT TYPE="SUBMIT" VALUE="SUBMIT THE FORM">
    </FROM>
    <HR>
  </BODY>
</HTML>
```



PERL SCRIPT:**Steps: - “2b.pl”**

- Create a file called “count.txt” in /var/www/cgi-bin as follows: -

```
# cd /var/www/cgi-bin
# cat > count.txt
1
<Ctrl D>
```

chmod a+rwx count.txt [You can either do this or right click on the “count.txt” file icon, click on properties->permissions and check read, write and execute permissions for the Owner, Group and Others].

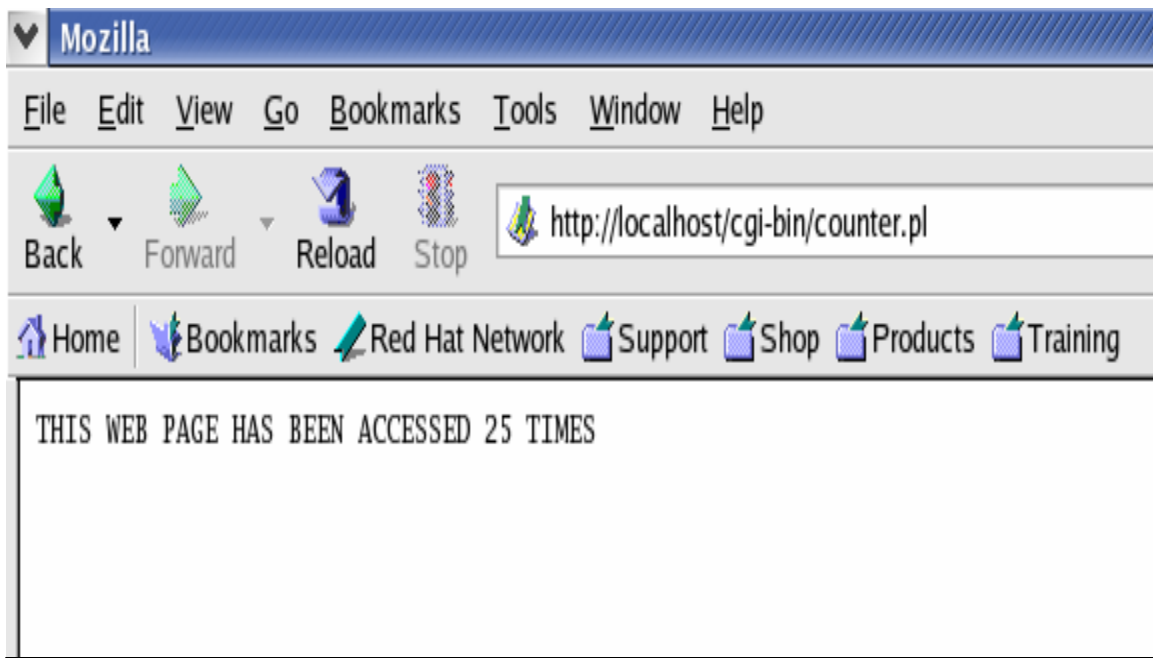
- The program aims to count the number of times a user visits a page. Therefore each time the page is accessed, the count is incremented. The count value is stored in a file called “count.txt”.

Make sure that this file is already created and has a value in it.

- Initially the value “1” is stored in it. *Make sure that all the permissions are given to this file.* The open () function opens the “count.txt” file and reads it into a File Handler. This value is then assigned to a variable called \$count. After reading the value into the variable, its value is incremented. Now the incremented value is once again stored back in the “count.txt” file after invoking the open () function again. The incremented value is printed using the print () and the file handlers are closed.

```
#!/usr/bin/perl
require "cgi-lib.pl";
print "content-type: text/plain", "\n\n";
$count_file="/usr/local/bin/count.txt";
if(open(FILE, "<" . $count_file))
{
    $no_accesses=<FILE>;
    close(FILE);
    if(open(FILE, ">" . $count_file))
    {
        $no_accesses++;
        print FILE $no_accesses;
        close(FILE);
        print "THIS WEB PAGE HAS BEEN ACCESSED ";
        print $no_accesses;
        print " TIMES";
    }
}
else
```

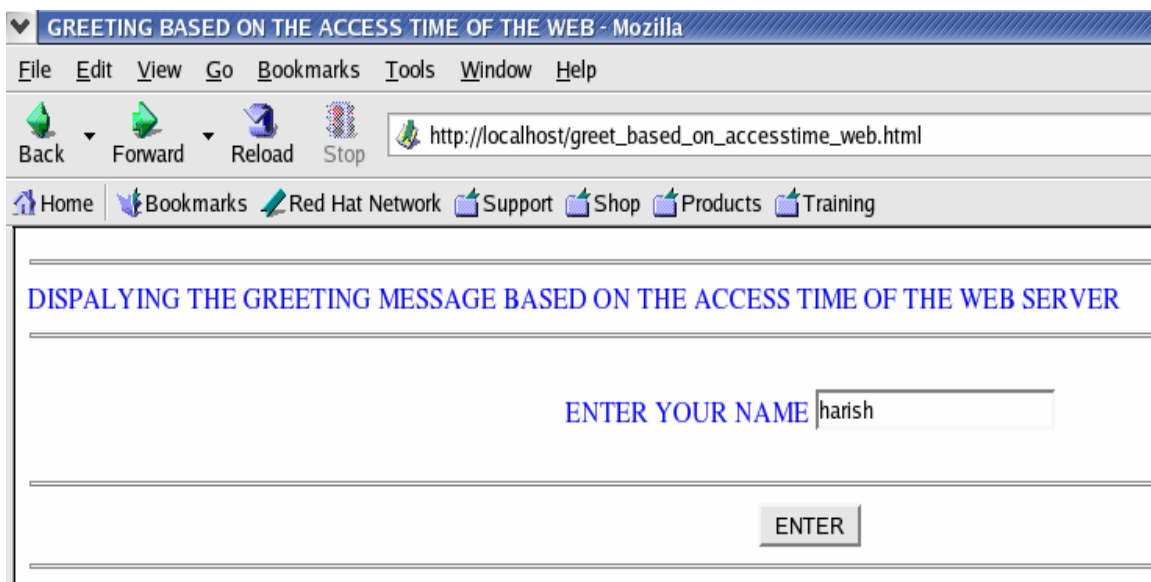
```
        {  
            print "cannot open the data file", "\n";  
            print $no_accesses;  
        }  
    }  
else  
{  
    print "sorry cant read from file";  
}  
exit(0);
```



3. PROGRAM TO DISPLAY A GREETING BASED ON THE ACCESS TIME OF THE WEB SERVER. ALSO TO VERIFY WHETHER THE WEB MASTER IS CURRENTLY LOGGED IN OR NOT.

HTML FILE:

```
<html>
  <head><title>GREETING BASED ON THE ACCESS TIME OF THE
  WEB</title></head>
  <body>
    <form action="http://localhost/cgi-bin/greet1.pl" method="get">
      <hr>
      <font color="BLUE">DISPALYING THE GREETING
      MESSAGE BASED ON THE ACCESS TIME OF THE WEB
      SERVER
      <hr>
      <br>
      <CENTER>ENTER YOUR NAME
      <INPUT TYPE="TEXT" NAME="NAME"
      SIZE="20"></CENTER>
      </font>
      <BR>
      <hr>
      <CENTER><input type="submit" value="ENTER"></CENTER>
      <hr>
    </form>
  </body>
</html>
```



PERL SCRIPT:**Steps: “3.pl”**

- Here our primary concern is to get access to the system clock. This is obtained by the code:

```
($seconds,$minutes,$hour)=localtime(time);
```

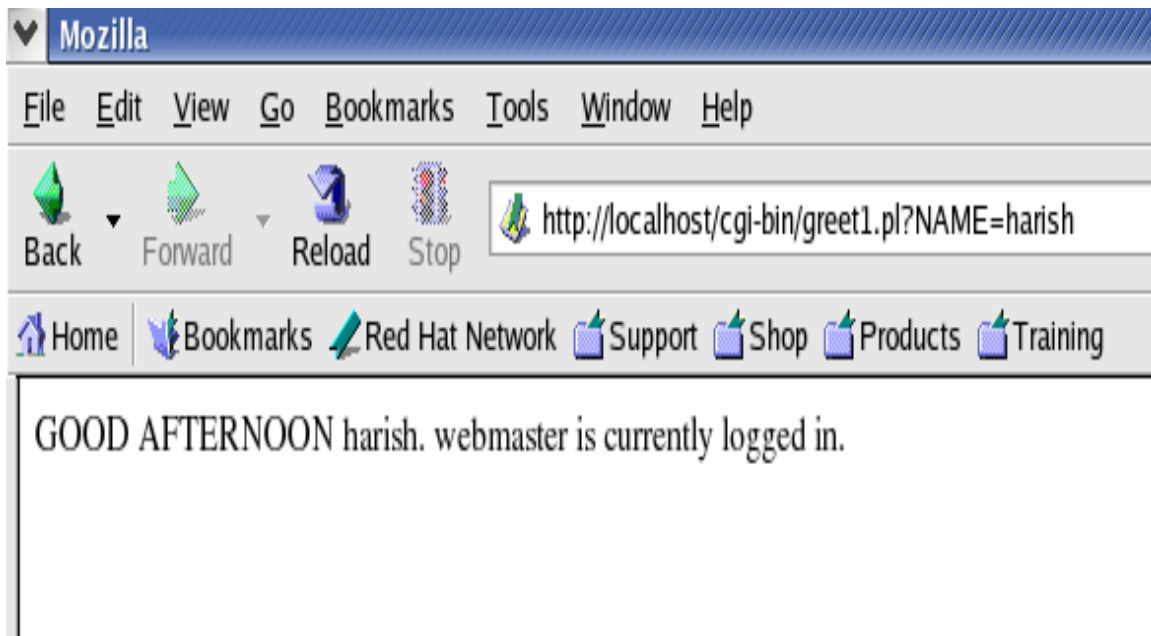
- Now once we have the time in our hand we may frame the “if conditions” and get the task done. For example, to print Good Morning, we do it as follows:

```
if( ($hr>=0) && ($hour<12) )
{ print “Good Morning!\n”; }
```

- The “Webmaster” is the “root” or the “super user”. To check if he/she is currently logged on, we make use of the “who” command. This command lists all the users currently logged on. Then we make use of the filter “grep” to search for the “root” in the output of the ‘who’ command. This is done by making use of the pipeline (|). The “grep” with the “c” option counts the number of occurrences the word “root” appears. If this count is equal to ‘0’, then the webmaster is not logged on else the webmaster is currently logged on.
- The entire piped command is executed by making use of the back ticks(`). An alternate approach is to make use of the system (`) function to execute the command.

```
#!/usr/bin/perl
require "cgi-lib.pl";
&ReadParse(*simple_form);
$name=$simple_form{ 'NAME' };
print "content-type: text/html", "\n\n";
$webmaster="root";
($seconds, $minutes, $hour)=localtime(time);
if ( ($hour>=23) || ($hour<=6) )
{
    $greeting="wow, you are up late";
}
elsif ( ($hour>6) && ($hour<12) )
{
    $greeting="GOOD MORNING";
}
elsif ( ($hour>=12) && ($hour<=18) )
{
    $greeting="GOOD AFTERNOON";
}
else
{
    $greeting="GOOD EVENING";
}
```

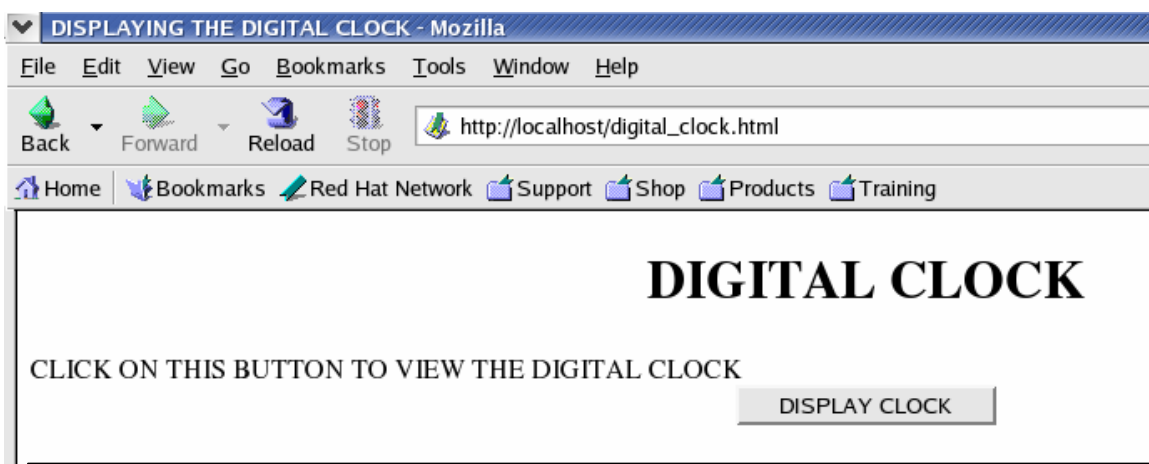
```
    }
    print $greeting , "\n";
    print $name, ".";
    open(CHECK, "/usr/bin/w -h -s $webmaster |");
    if(<CHECK>=~/$webmaster/)
    {
        $in_out=" webmaster is currently logged in.";
    }
    else
    {
        $in_out=" webmaster is just stepped out.";
    }
    print $in_out;
exit(0);
```



4. PROGRAM TO DISPLAY A DIGITAL CLOCK WHICH DISPLAYS THE CURRENT TIME OF THE SERVER.

HTML FILE:

```
<html>
  <head><title>DISPLAYING THE DIGITAL CLOCK</title></head>
  <body>
    <form action="http://localhost/cgi-bin/dig_ck.pl" method="post">
      <h1><center>DIGITAL CLOCK</center></h1>
      CLICK ON THIS BUTTON TO VIEW THE DIGITAL CLOCK
      <br>
      <center><input type="submit" value="DISPLAY
      CLOCK"></center>
    </form>
  </body>
</html>
```



PERL SCRIPT:**Steps: - “4.pl”**

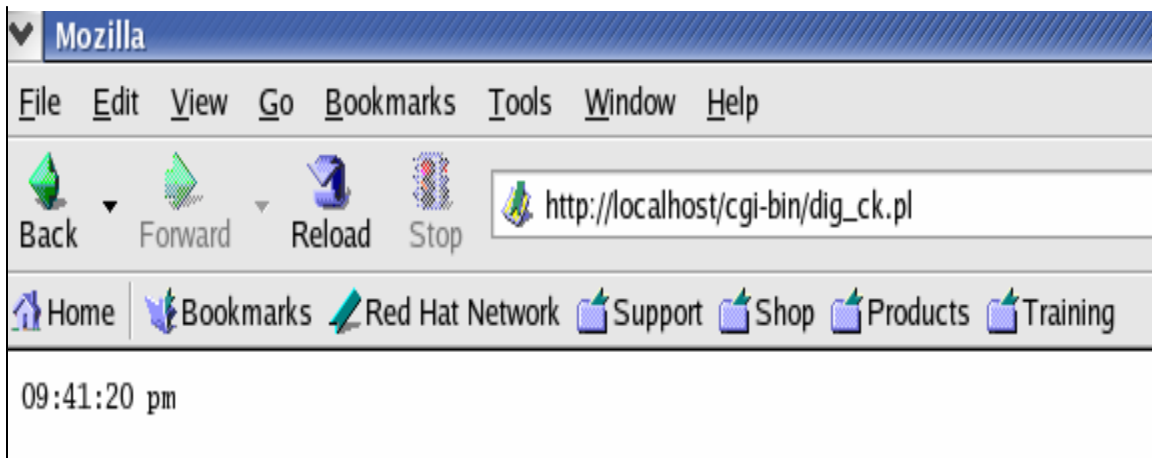
- Firstly, the system time is obtained by making use of the localtime () function

```
($sec,$min,$hr)=localtime(time);  
print "<meta http-equiv=\`refresh\` content=\`1\`>";  
print "The Current time is:$hr:$min:$sec<br>";
```

- . The concept of Meta tags in HTML is to be known to understand the program.
Meta tags are hidden html tags of an html document that are not displayed in a browser but provide a browser or search engine robot with useful information. Meta tags used by the browser define particular things about the document, like what character set to use when displaying the characters on the page, or what language the html document is written it, or how many seconds should be waited before refreshing the page or redirecting to another webpage
- META tags should be placed in the **head** of the HTML document, between the <HEAD> and </HEAD> tags. META tags with an HTTP-EQUIV attribute are equivalent to HTTP headers. Typically, they control the action of browsers, and may be used to refine the information provided by the actual headers. Tags using this form should have an equivalent effect when specified as an HTTP header, and in some servers may be translated to actual HTTP headers automatically or by a pre-processing tool.
- The meta tag used in the program specifies that the action to be performed is “Refreshing” the page. The content attribute specifies that the pages needs to be refreshed every 1 second.

NOTE: Without using the meta tag, the time is displayed on the page and the “Refresh” button is to be clicked each time to see the updated time. But by making use of the meta tags, the time is updated automatically by the browser. Since the page gets refreshed every second, the “Stop” button in the browser may not be accessible. Hence close the Browser window to stop the program.

```
#!/usr/bin/perl
require "cgi-lib.pl";
print "content-type: text/plain", "\n\n";
$GS="/usr/bin/gs";
$|=1;
($seconds, $minutes, $hour)=localtime(time);
if($hour>12)
{
    $hour-=12;
    $ampm="pm";
}
else
{
    $ampm="am";
}
if($hour==0)
{
    $hour=12;
}
$time=sprintf("%02d:%02d:%02d %s", $hour, $minutes,
$seconds, $ampm);
print $time;
exit(0);
```

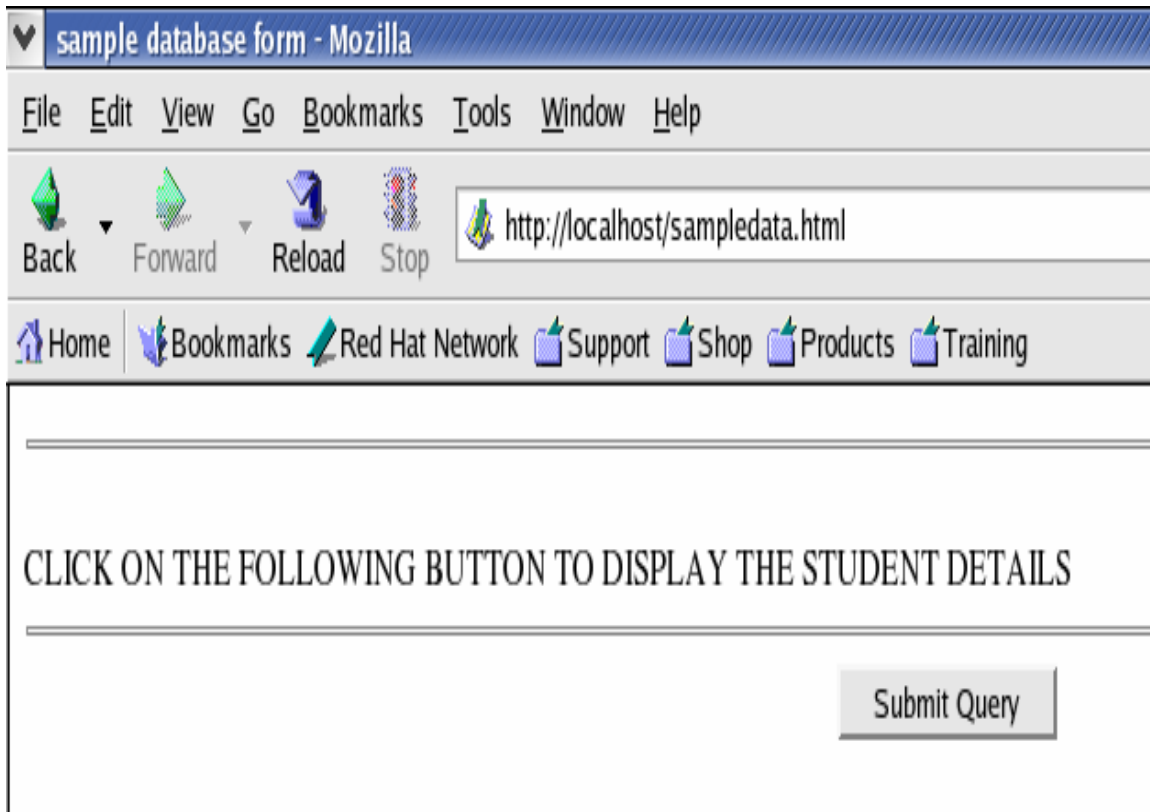


PART 2

5. PROGRAM TO DISPLAY THE CURRENT CONTENTS OF THE TABLE IN A DATABASE.

HTML FILE:

```
<html>
  <head><title>sample database form</title></head>
  <body>
    <form action="http://localhost/cgi-bin/samdatlab.pl" method="post">
    <HR>
    <BR>
    CLICK ON THE FOLLOWING BUTTON TO DISPLAY THE
    STUDENT DETAILS
    <HR>
    <center><input type=submit name="student details"></center>
    </form>
  </body>
</html>
```



PERL SCRIPT:

Steps: "5.pl"

- It is assumed that a MySQL database and the corresponding tables are ready for use. If not, start the MySQL Server and make everything ready. We have called the database "*employee*" and the table as "*emp*" in this case. The password used for the '*apache*' user is '*lamp*'.
- DBI (Database Independent Interface) is a Perl Module that provides methods to manipulate SQL Databases. With DBI, one can connect to a database within a Perl script and issue all kinds of queries like select, insert and delete. The "use DBI" pragma makes use of this Perl module.
- The "use strict" pragma makes the user compulsorily declare a variable before it is used. This pragma overrides the Perl's built in feature that states that variables can be used directly without declaration. Therefore with 'strict', if you use a variable before it is declared, an error is generated. Once this pragma is used, the '*my*' keyword is used to declare variables.
- The variable "\$dbh" serves as a Database Handler. The connect () function is used to connect to the server. The first parameter indicates that the DBI Perl module is being used to the MySQL database and the database name is

‘employee’. The second parameter indicates the user “apache” and the third parameter indicates that the password is “lamp” (Remember that this is the password that we had given with the Grant command. Make sure that this password matches with the one given with the Grant command)

- The variable “\$dbh” is the Statement handler that prepares the SQL statement (Query) to be executed.
- The execute () function is used to execute the SQL query.
- The fetchrow () function is used to loop through each record of the table and the records are stored in the variables “\$a” and “\$b” and these values are printed.
- After this, ensure that you finish the statement handler and disconnect from the server by making use of the finish () and disconnect () functions respectively.

```
#!/usr/bin/perl
require "cgi-lib.pl";
use DBI;
print "content-type: text/html", "\n\n";
$dbh=DBI->connect('dbi:mysql:mysql','root','harry');
$r=$dbh->prepare("select * from stud");
$r->execute();
while(($a,$b)=$r->fetchrow())
{
    print "<HTML>", "\n";
    print "<BODY>", "\n";
    print "<HR><PRE>";
    print "NAME <input type=\"text\" name=\"g1\" value= $a>";
    print "REGNO <input type=\"text\" name=\"g2\" value= $b>";
    print "<HR></PRE>", "\n";
    print "</BODY></HTML>", "\n";
}
$r->finish();
exit(0);
```

Mozilla

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop http://localhost/cgi-bin/samdata.pl

Home Bookmarks Red Hat Network Support Shop Products Training

NAME REGNO

NAME REGNO

NAME REGNO

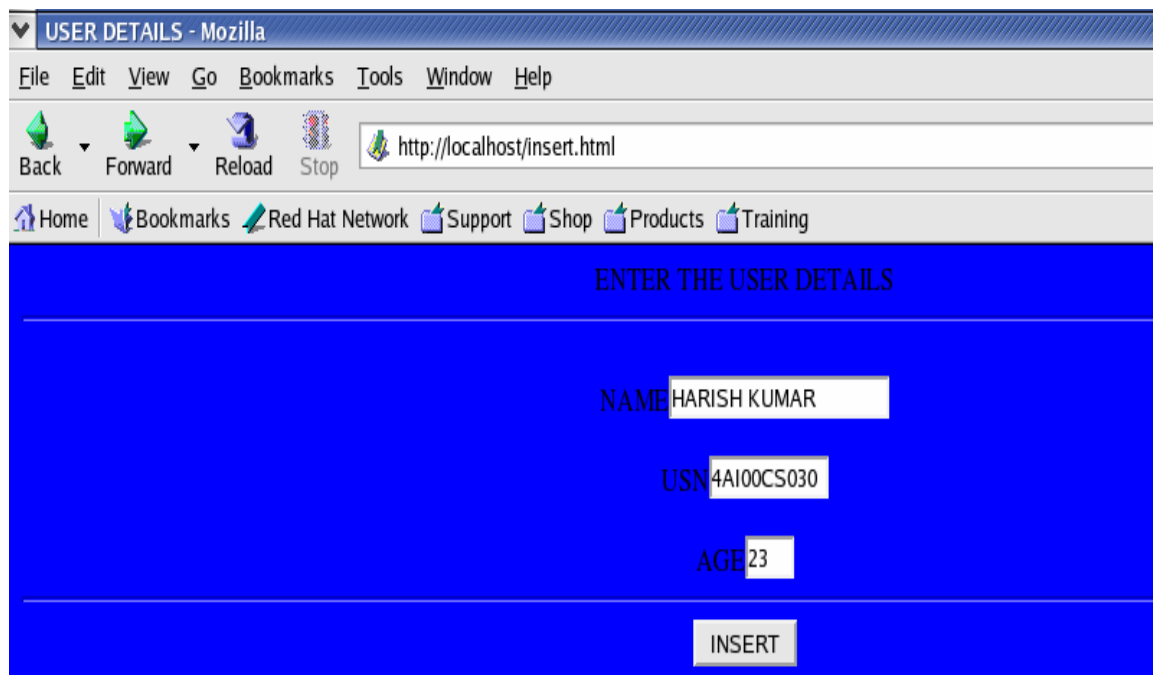
NAME REGNO

NAME REGNO

6. PROGRAM TO INSERT NEW NAME AND AGE INFORMATION ENTERED BY THE USER INTO THE DATABASE.

HTML FILE:

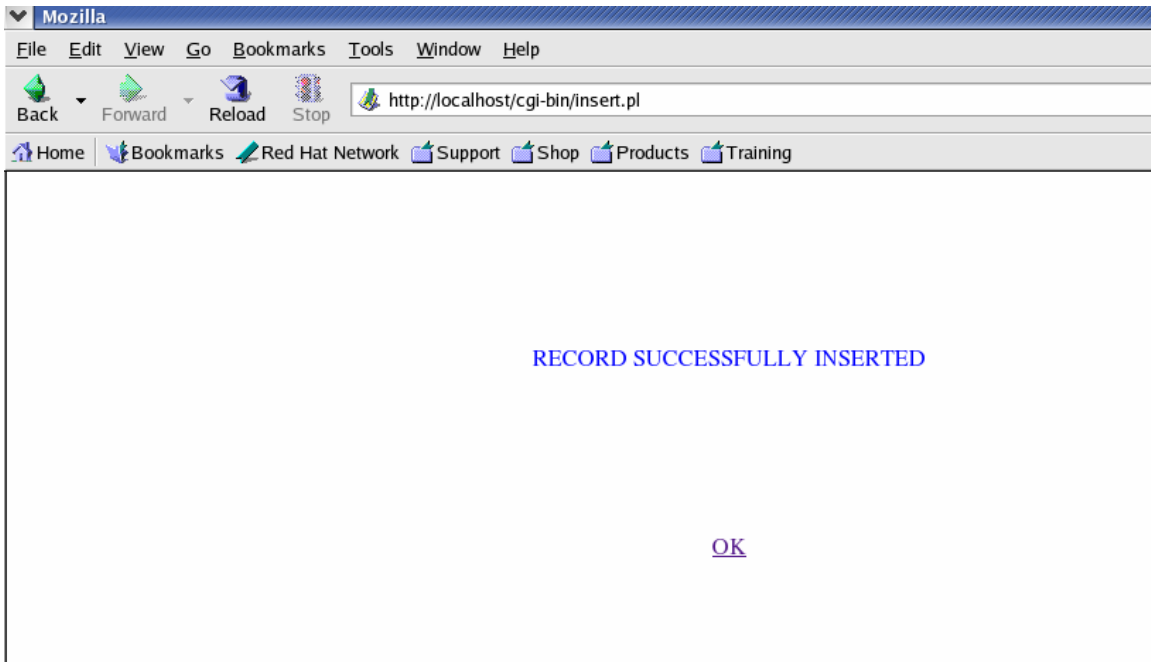
```
<html>
  <head><title>USER DETAILS</title></head>
  <body bgcolor="blue">
    <form action="http://localhost/cgi-bin/insert.pl" method="post">
      <CENTER>ENTER THE USER DETAILS</CENTER>
      <HR>
      <BR>
      <CENTER>NAME<INPUT TYPE="TEXT" NAME="UNAME"
      SIZE="20"></CENTER>
      <BR>
      <CENTER>USN<INPUT TYPE="TEXT" NAME="USN"
      SIZE="10"></CENTER>
      <BR>
      <CENTER>AGE<INPUT TYPE="TEXT" NAME="AGE"
      SIZE="3"></CENTER>
      <HR>
      <CENTER><INPUT TYPE="SUBMIT"
      VALUE="INSERT"></CENTER>
    </FORM>
  </BODY>
</HTML>
```



PERL SCRIPT:**Steps: “6.pl”**

- The ReadParse() function is used to capture the values entered by the user and access it from the Perl script.
- Here, two statement handlers are used. “\$dbh” is used to prepare the “insert” statement and “\$r” is used to prepare the “select” statement. The rest of the code can be easily understood if you have understood the previous program.
- Make sure that you finish both the statement handlers.

```
#!/usr/bin/perl
require "cgi-lib.pl";
use DBI;
print "content-type: text/html", "\n\n";
$dbh=DBI->connect('dbi:mysql:mysql','root','harry');
&ReadParse(*p);
$x=$p{'UNAME'};
$y=$p{'USN'};
$a=$p{'AGE'};
$r=$dbh->prepare("insert into student values('$x','$y','$a')");
$r->execute();
$r->finish();
print "<html><body>";
print "<center>";
print "<font color=BLUE>";
print "<br><br><br><br><br><br>";
print "RECORD SUCCESSFULLY INSERTED";
print "<br><br><br><br><br><br><br>";
print "<a href=http://localhost/insert.html>OK</a>";
print "</font>";
print "</center>";
print "</body></html>";
exit(0);
```



PART 3:

PHP is a server side scripting language that allows your web site to be truly dynamic. PHP stands for Hypertext Preprocessor (recursive acronym). We will be using PHP 4.0 for lab usage. It is freely downloadable from www.php.net.

Conventions:

- Enclose all the PHP parts of the script within the PHP start and end tags - `<?php>` and `?>` OR `<? and ?>`.
- Enter a Semicolon at the end of each line of the PHP code.

A sample PHP Program

PHP Code can be embedded within the HTML pages, tagging the PHP parts of your web pages with php tags tells the web server to start reading the files and parse them according to the php language. Anything between PHP tags is evaluated by the PHP part of the web server. Others are interpreted as HTML tags.

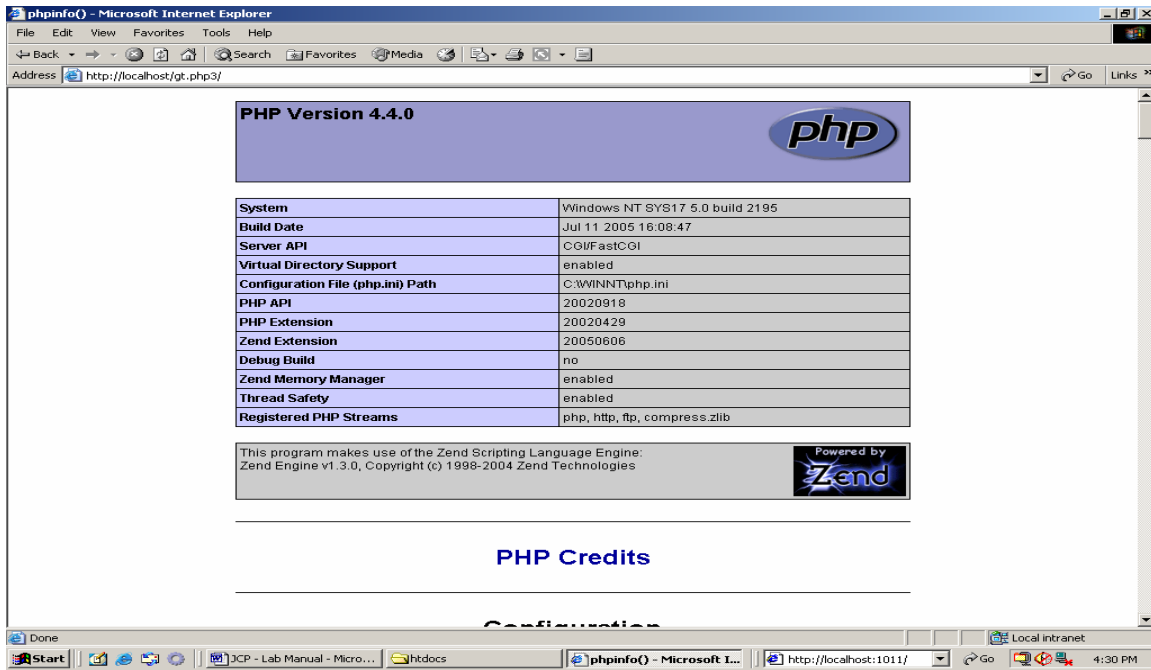
Open a text editor, say a notepad and type the code

```
<html><body>
<? Phpinfo( ); ?>
</body></html>
```

Save this file as info.php in the path

C:/ProgramFiles/Apache Group/Apache/htdocs/<create a folder, say LAB1>/.

Open Internet explorer and type the URL <http://localhost/LAB1/info.php>. If you have successfully installed PHP, you should see the following page displayed:



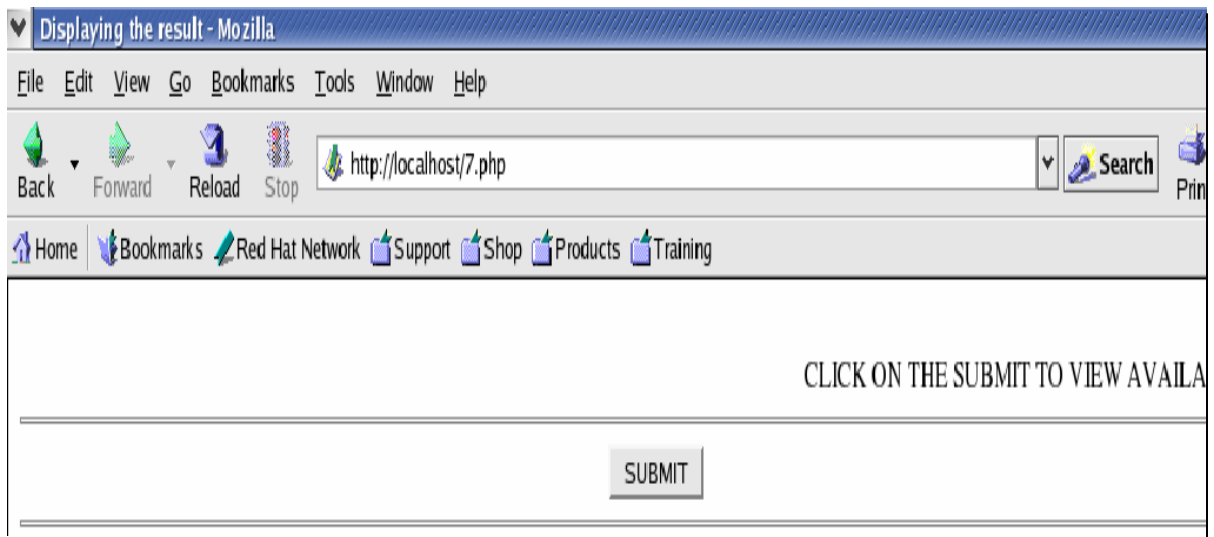
Develop and execute the following programs using HTML and PHP. Create database using MYSQL wherever necessary.

1. Program to query the database and to display the result on a page.
2. Program to accept book information viz. Accession number, title, authors, edition and publication from a web page and to store those in a database.
3. Program to search a book for a title given by the user on a web page and display the search results with proper headings.

7. Program to query the database and to display the result on a page

HTML: 7.php

```
<HTML>
  <HEAD>
    <TITLE>Displaying the result</TITLE>
  </HEAD>
  <BODY>
    <FORM ACTION="<?php echo("7a.php"); ?>" METHOD=POST>
    <P>
      <marquee>CLICK ON THE SUBMIT TO VIEW AVAILABLE
      BOOKS</marquee>
    <hr>
    <CENTER><INPUT TYPE="SUBMIT" NAME="submit"
    VALUE="SUBMIT"></CENTER>
    <hr>
  </BODY>
</HTML>
```



Steps: - “7a.php” –

- Connect to the database.

```
<?
    $mysql = mysql_connect ( "localhost" , "root" );
    or die ( " cannot connect to mysql" );
    $result = mysql_db_query( "lib" , select * from booksinfo " )
    or die ( "Query failed" );
?>
```

- Create a table to display the query information using <Table> tag.
- Fetch records from database and display

```
<? while ( $array = mysql_fetch_row( $result ) ) : ?>
    <td> <? echo $array[0] ; ?> </td>
    <td> <? echo $array[1] ; ?> </td>
    <td> <? echo $array[2] ; ?> </td>
    <td> <? echo $array[3] ; ?> </td>
```

```
<? endwhile : ?>
```

- Close all Html tags as well as mysql.

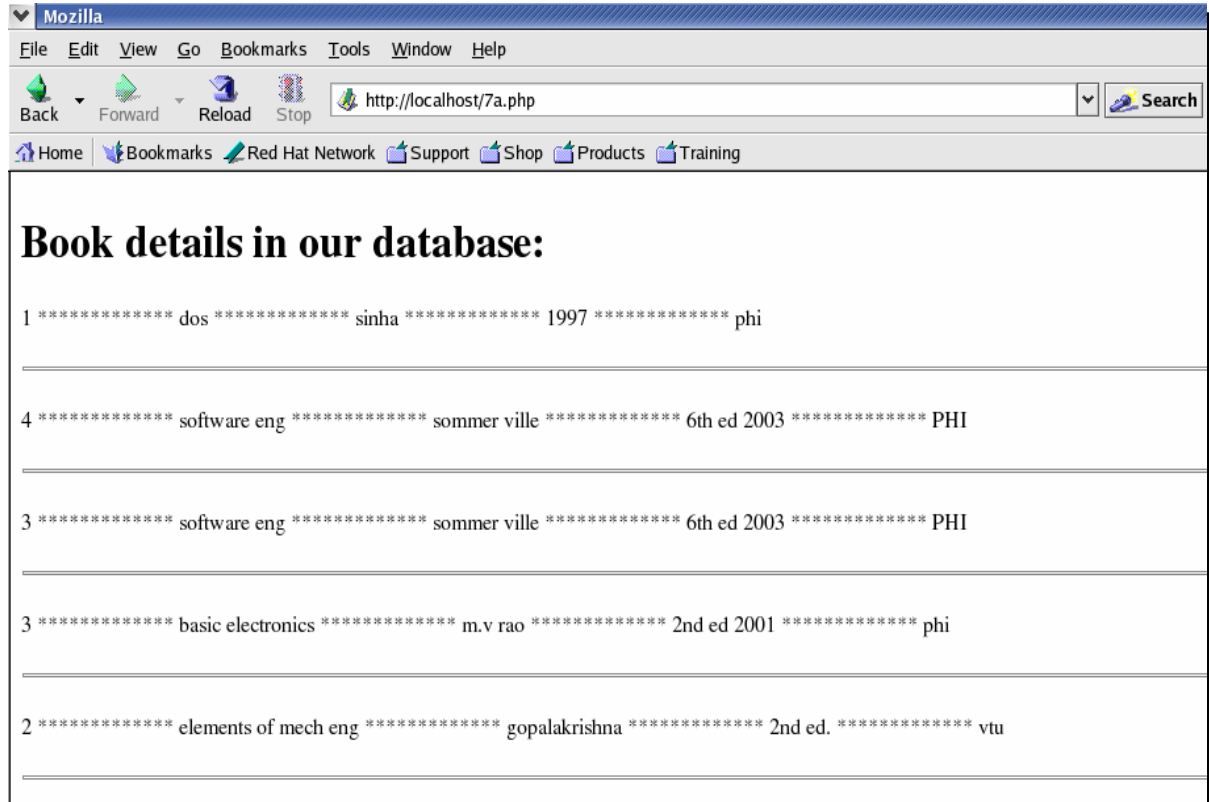
```
<? Mysql_close( $mysql ); ?>
```

```
<?php
```

```
    // Connect to the database server
    $dbcnx = @mysql_connect("localhost","root", "harry");
    if (!$dbcnx)
    {
        echo( "<P>Unable to connect to the " .
            "database server at this time.</P>" );
        exit();
    }
    // Select the mysql database
    if ( ! @mysql_select_db("mysql" ) )
    {
        echo( "<P>Unable to locate the mysql " .
```

```
"database at this time.</P>" );
    exit();
}
?>
<P><h1> Book details in our database:</h1> </P>
<?php
// Request the text of all the book details
$result = mysql_query("SELECT acc_no,title,author,edition,publication FROM
book");
if (!$result)
{
echo("<P>Error performing query: " .
    mysql_error() . "</P>");
    exit();
}
// Display the text of each row
while ( $row = mysql_fetch_array($result) )
{
echo("<p>");
echo($row["acc_no"]);
echo(" ***** ");
echo($row["title"]);
echo(" ***** ");
echo($row["author"]);
echo(" ***** ");
echo($row["edition"]);
echo(" ***** ");
echo($row["publication"]);
echo("<hr>");
echo("</p>");
}
}
```

?>




```

        </FORM>
    </BODY>
</HTML>

```

INSERTING INTO THE BOOK - Mozilla

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop http://localhost/8.php Search

Home Bookmarks Red Hat Network Support Shop Products Training

ENTER THE BOOK DETAILS HERE AND PRESS INSERT

Accession number:

Title:

Authors:

Edition:

Publication:

To create 8a.php: -

- connect to the database
- Get the information from webpage

```

$accession_no = $_GET ["accession_no"];
$edition = $_GET ["edition"];
.
.
.
$publication = $_GET ["publication"];
If( $accession_no == "" or $edition == "" or ..... )
{ print "Wrong or no input data";
  Die (" Record not inserted ");

```

- To insert information into the database

Mysql_query("insert into bookinfo values (\$accession_no, '\$author',)
or die (" Query failed ");

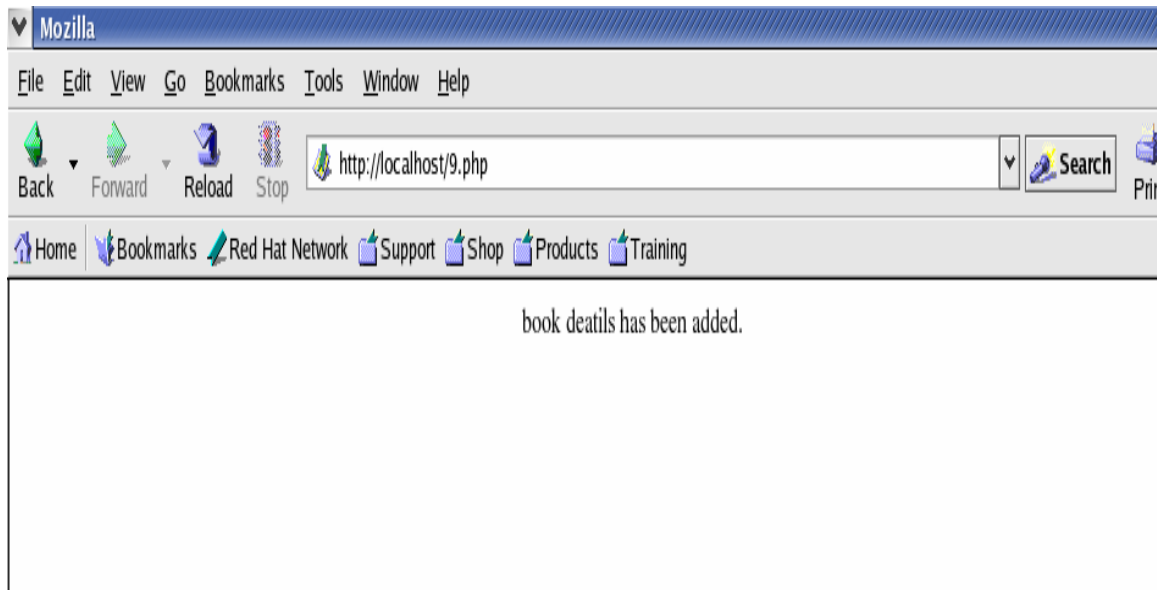
- Close all Html tags as well as mysql.

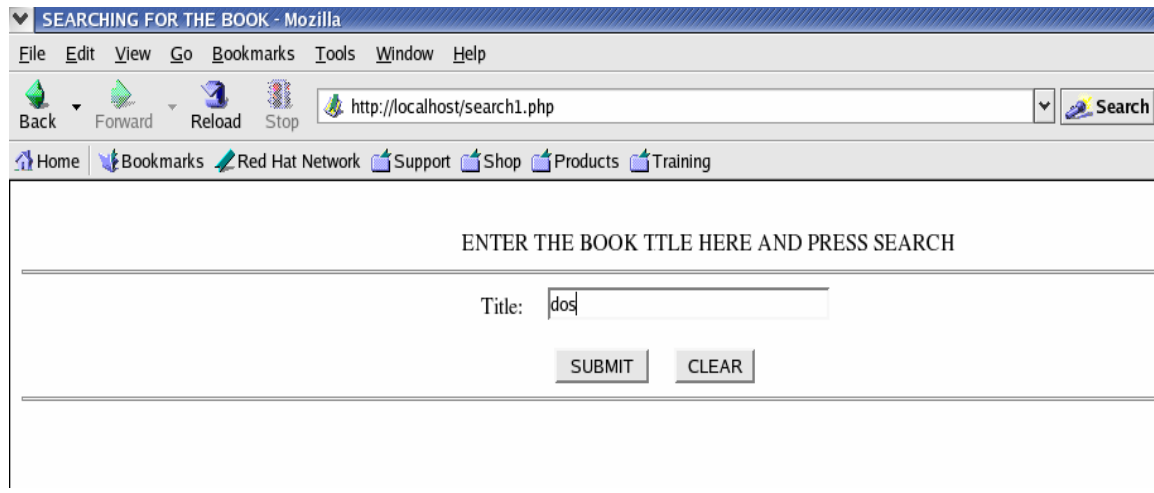
```
<? Mysql_close( $mysql ); ?>
```

```
<?php
// Connect to the database server
$dbcnx = @mysql_connect("localhost", "root", "harry");
if (!$dbcnx)
{
echo( "<P>Unable to connect to the " .
      "database server at this time.</P>" );
exit();
}
// Select the mysql database
if ( ! @mysql_select_db("mysql" )
{
echo( "<P>Unable to locate the mysql " .
      "database at this time.</P>" );
exit();
}
// If a details has been submitted,
// add it to the database.
$acc_no=$_POST["acc_no"];
$acc_no=addslashes($acc_no);
$title=$_POST["title"];
$title=addslashes($title);
$authors=$_POST["authors"];
$authors=addslashes($authors);
$edition=$_POST["edition"];
$edition=addslashes($edition);
```

```
$publication=$_POST["publication"];
$publication=addslashes($publication);
if($acc_no=="")
{
echo("<P><center>Accession Number Not Provided.</center></P>");
$check=1;
}
if($title=="")
{
echo("<P><center>Title Not Provided.</center></P>");
$check=1;
}
if($authors=="")
{
echo("<P><center>Author name Not Provided.</center></P>");
$check=1;
}
if($edition=="")
{
echo("<P><center>Edition Not Provided.</center></P>");
$check=1;
}
if($publication=="")
{
echo("<P><center>Publication Not Provided.</center></P>");
$check=1;
}
if($check!=1)
{
    $sql = "INSERT INTO book SET " .
        "acc_no='$acc_no', " .
```

```
"title='$title', " .  
"author='$authors', "  
"edition='$edition', "  
"publication='$publication";  
}  
if (mysql_query($sql))  
{  
    echo("<P><center>book deatils has been added.</center></P>");  
}  
else  
{  
    echo("<P><center>Unable to add submitted book: " .  
        mysql_error() . "</center></P>");  
}  
?>
```





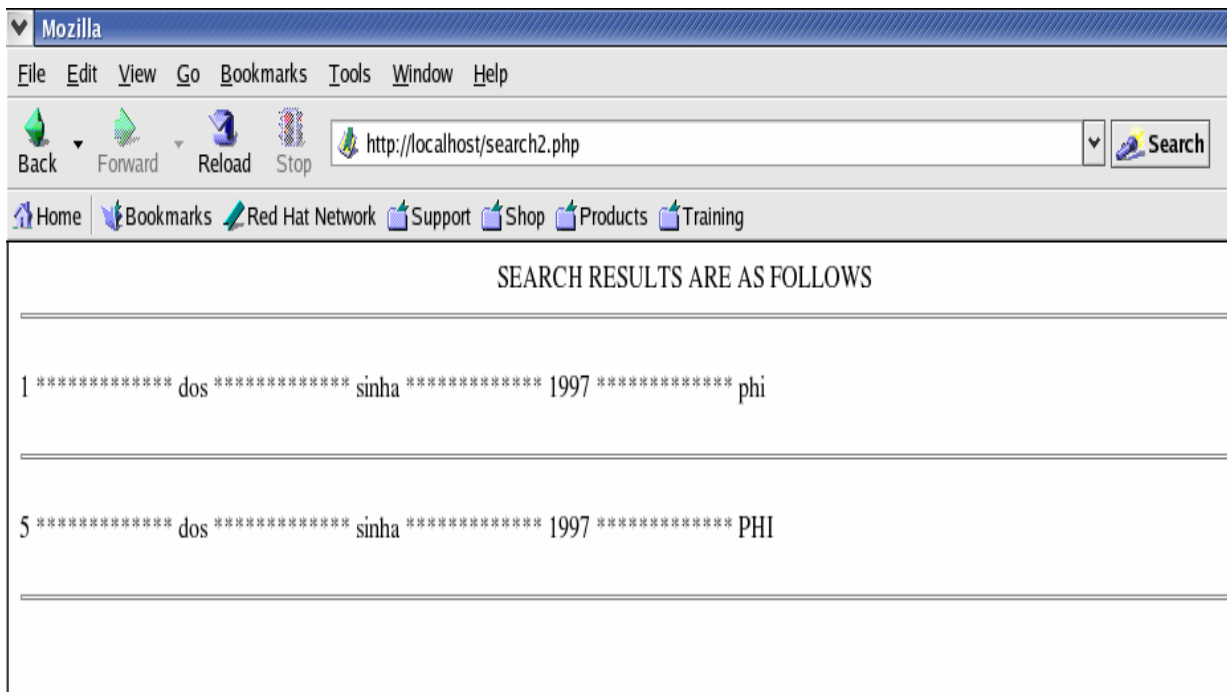
Step to search for the specific record

```
$TitleSearch = " $_GET["titlesearch"]";
If( $TitleSearch == "" )
{ print (" Wrong or no input data ");
  Die (" Record not Found "); }
$result = mysql_query(" select * from bookinfo where title like ' % $TitleSearch % ' ");
Display the result on web page in the form of a table
```

```
<?php
  // Connect to the database server
  $dbcnx = @mysql_connect("localhost", "root", "harry");
  if (!$dbcnx)
  {
    echo ("<P>Unable to connect to the " .
      "database server at this time.</P>" );
    exit();
  }
  // Select the mysql database
  if ( ! @mysql_select_db("mysql") )
  {
    echo ("<P>Unable to locate the mysql " .
      "database at this time.</P>" );
```

```
        exit();
    }
    // If a details has been submitted,
    // add it to the database.
$title=$_POST["title"];
$title=addslashes($title);
if($title=="")
{
echo("<P><center>Title Not Provided.</center></P>");
$check=1;
}
if($check!=1)
{
    $result = mysql_query("select acc_no,title,author,edition,publication from book WHERE
title LIKE '%$title%'");
}
    if (!$result)
    {
        echo("<P>Error performing query: " .
            mysql_error() . "</P>");
        exit();
    }
echo("<center>SEARCH RESULTS ARE AS FOLLOWS</center>");
echo("<hr>");
// Display the text of each row
while ( $row = mysql_fetch_array($result) )
{
echo("<p>");
echo($row["acc_no"]);
echo(" ***** ");
echo($row["title"]);
```

```
echo(" ***** ");
echo($row["author"]);
echo(" ***** ");
echo($row["edition"]);
echo(" ***** ");
echo($row["publication"]);
echo("<hr>");
echo("</p>");
}
?>
```



PART – 3

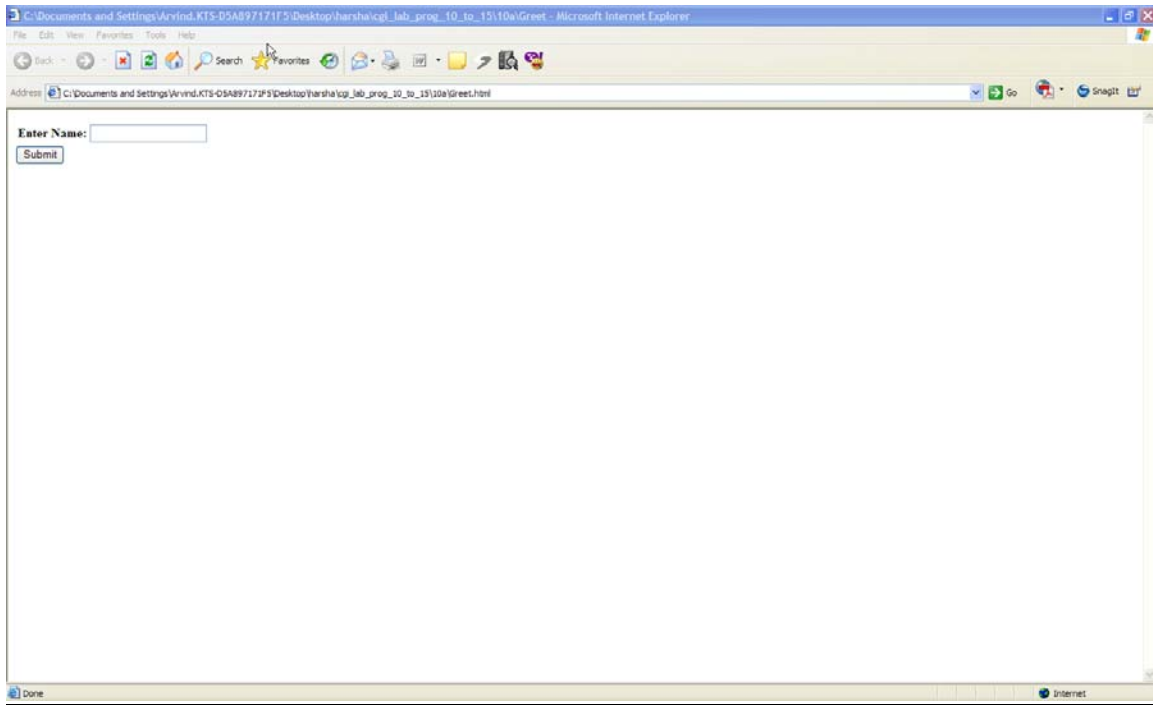
10A) PROGRAM TO ACCEPT USER NAME AND DISPLAY A GREETING MESSAGE.

Steps:“p10a.java” –

- A textbox is used in the HTML file to take the name entered by the user and it is called “t1”. The value present in “t1” textbox can be accessed in the servlet program by making use of the `getParameter()`. This value is stored in a variable called “name”. Next the greeting message is printed on the screen by making use of the `println ()` method. The URL pattern used here is “hi”.

HTML FILE

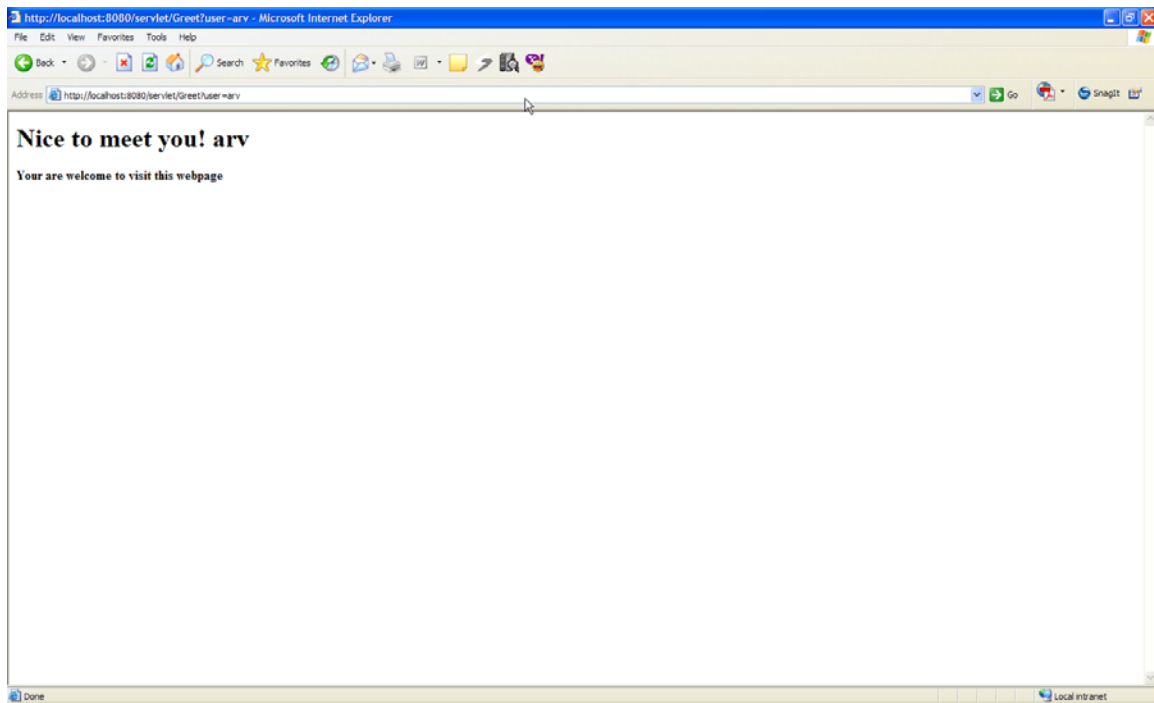
```
<html>
  <body>
    <form action="http://localhost:8080/servlet/Greet" method=get>
      <table>
        <tr>
          <td><b>Enter Name: </b></td>
          <td><input type=textbox name="user"></td>
        </tr>
      </table>
      <input type=submit value="Submit">
    </form>
  </body>
</html>
```



JAVA FILE

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Greet extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        String username = req.getParameter("user");
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        if(username.length() > 0) {
            pw.println("<b><h1>");
            pw.println("Nice to meet you! "+ username);
            pw.println("</h1>");
            pw.println("Your are welcome to visit this webpage");
            pw.println("</b>");
        }
        else {
            pw.println("<h2>You have not entered your name!!!</h2>");
        }
    }
}
```

```
    pw.close();  
  }  
}
```



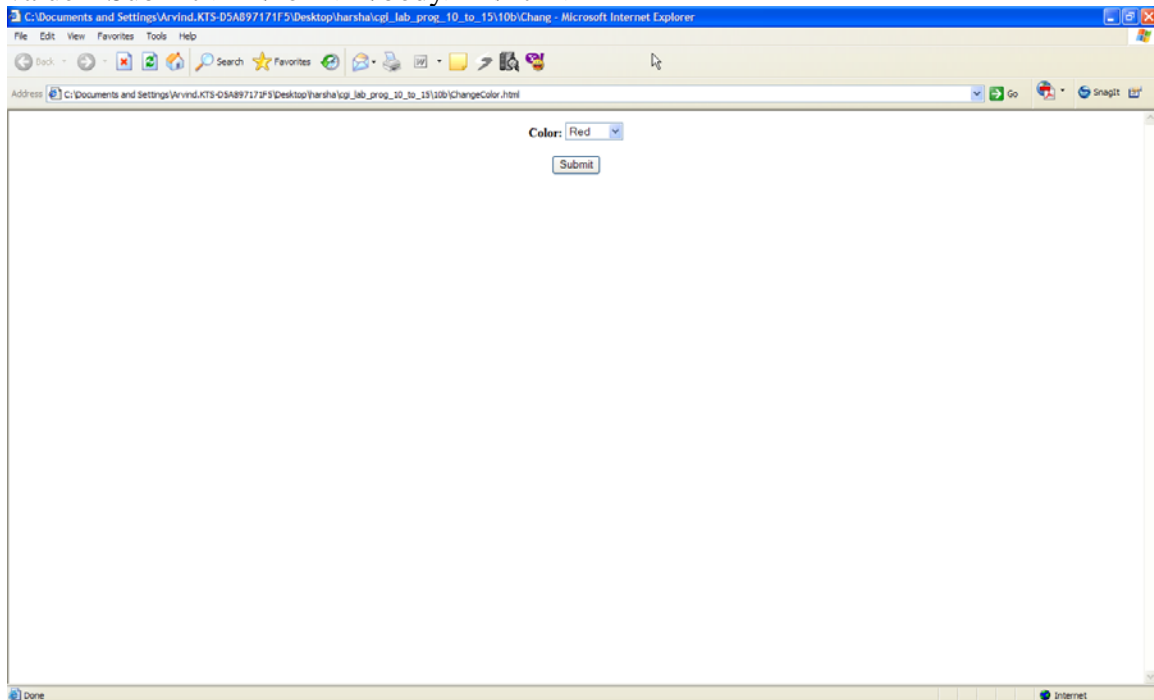
10B) PROGRAM TO CHANGE THE BACKGROUND COLOR OF THE PAGE BASED ON THE COLOR SELECTED BY THE USER

Steps:“p10b.java”

- The name of the colour is entered by the user in the HTML page. The resultant page has a background colour that the user has entered

HTML FILE

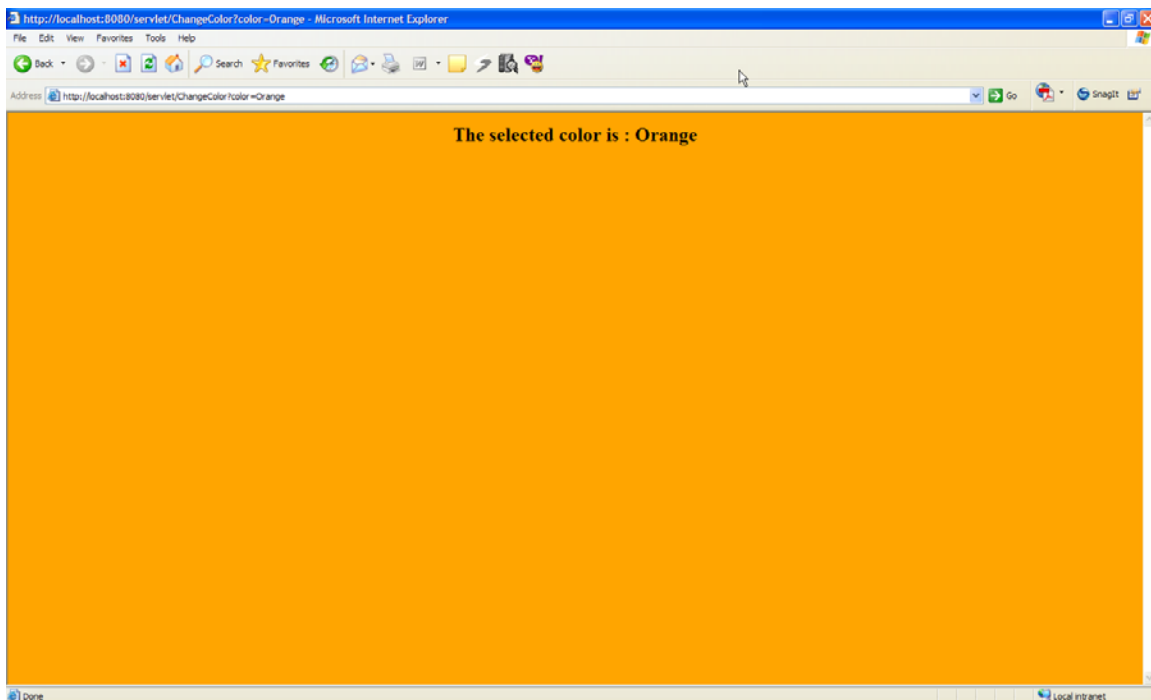
```
<html>
<body>
<center>
<form action="http://localhost:8080/servlet/ChangeColor" method=get>
<b>Color:</b>
<select name="color" size="1">
<option value="Red">Red</option>
<option value="Green">Green</option>
<option value="Blue">Blue</option>
<option value="White">White</option>
<option value="Black">Black</option>
<option value="Brown">Brown</option>
<option value="Gray">Gray</option>
<option value="Orange">Orange</option>
<option value="Yellow">Yellow</option> </select> <br><br> <input type=submit
value="Submit"> </form></body> </html>
```



JAVA FILE

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ChangeColor extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        String color = req.getParameter("color");
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        pw.println("<html>");
        pw.println("<body bgcolor=" + color + ">");
        pw.println("<center><h2>The selected color is :  ");
        pw.println(color + "</h2></center>");
        pw.println("</body></html>");
        pw.close();
    }
}
```



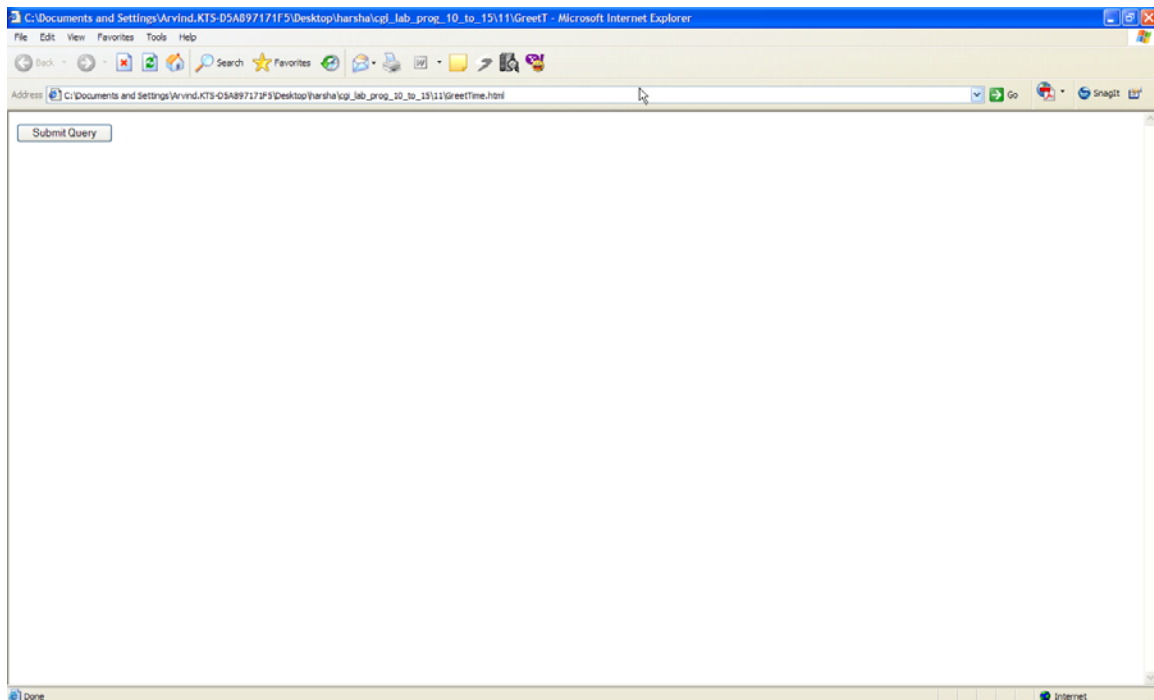
11) PROGRAM TO DISPLAY A GREETING BASED ON THE ACCESS TIME OF THE SERVER

Steps:“p11.java”

- The program is similar to Program 10 (a) except that time is also being printed here. For this purpose the “java.util.*” package is being imported. The getHours () and the getMinutes () methods are being used get the hours and minutes respectively. Based on the value of “hours”, the appropriate greeting message is displayed. The URL pattern being used is “hello”.

HTML FILE

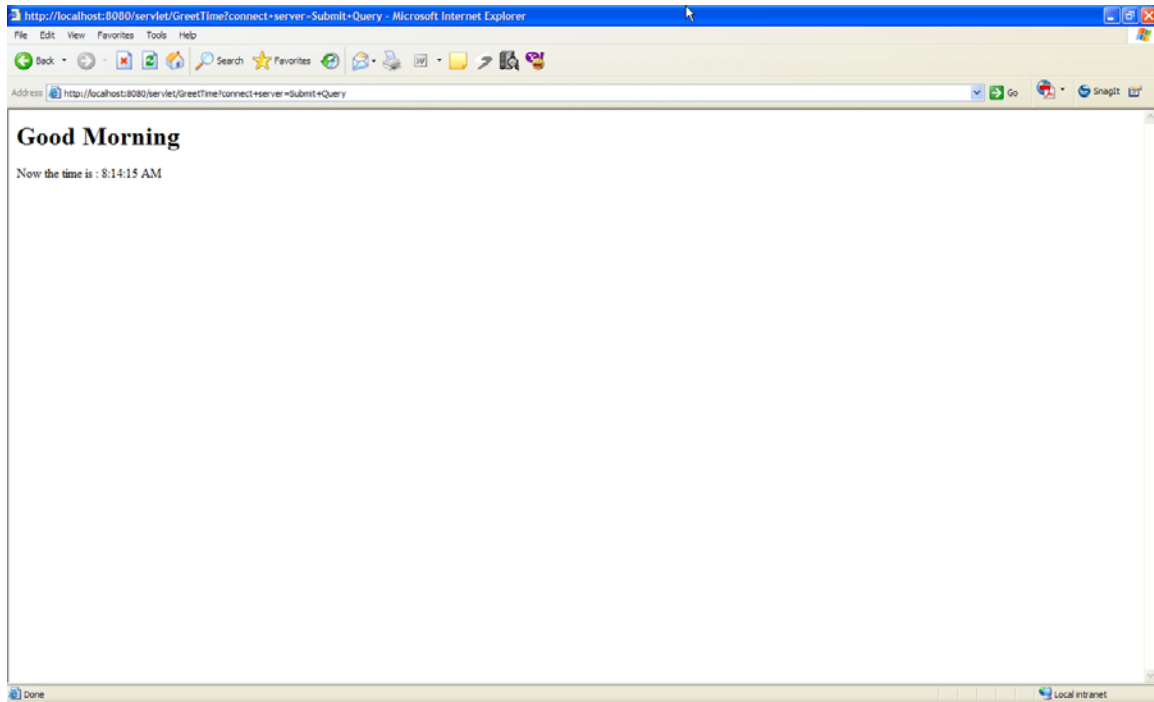
```
<html>
  <body>
    <form action="http://localhost:8080/servlet/GreetTime" method="get">
      <input type="submit" name="connect server">
    </body>
</html>
```



JAVA FILE

```
import java.io.*;
import java.util.Calendar;
import javax.servlet.*;
import javax.servlet.http.*;

public class GreetTime extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        int hour,minute,second;
        String ampm = new String("AM");
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        Calendar calendar = Calendar.getInstance();
        hour = calendar.get(calendar.HOUR);
        minute = calendar.get(calendar.MINUTE);
        second = calendar.get(calendar.SECOND);
        pw.println("<h1>");
        if(hour > 23 || hour <= 6)
            pw.println("You are Up Late");
        else if(hour > 6 && hour <= 12)
            pw.println("Good Morning");
        else if(hour > 12 && hour <= 18)
            pw.println("Good Afternoon");
        else
            pw.println("Good Evening");
        pw.println("</h1>");
        if(hour == 0)
            hour = 12;
        if(hour > 12)
        {
            hour -= 12;
            ampm = "PM";
        }
        pw.println("Now the time is : " + hour + ":" + minute + ":" + second + " "
+ ampm);
        pw.close();
    }
}
```



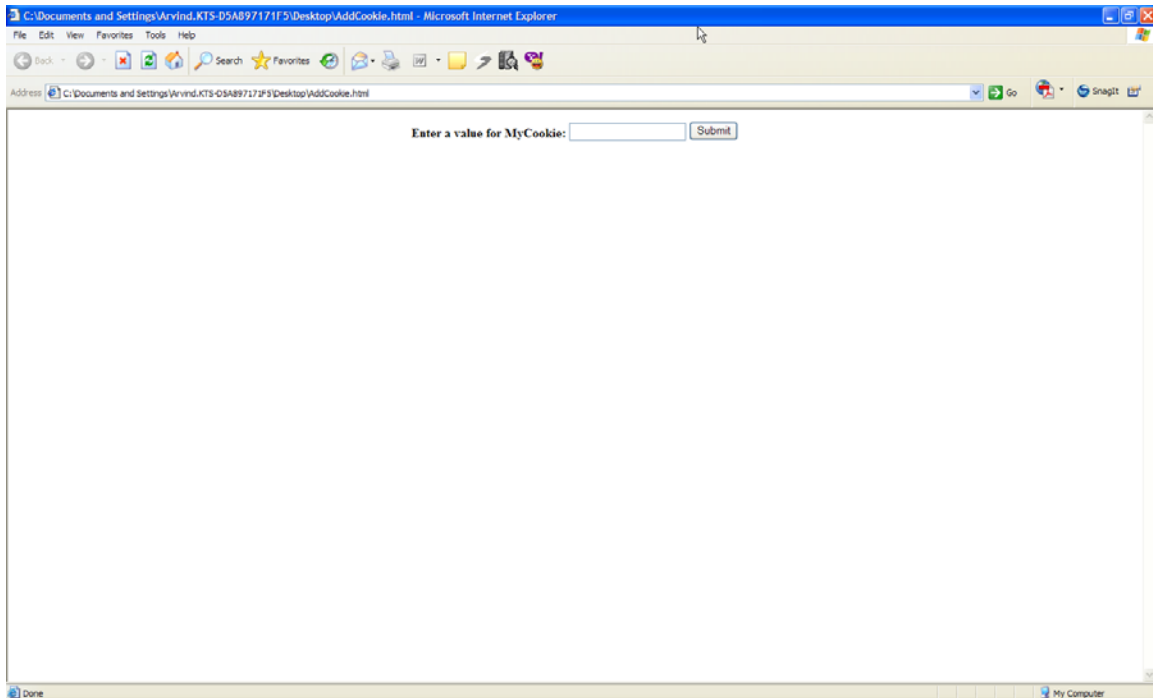
12) PROGRAM TO CREATE AND DISPLAY A COOKIE

Steps “p12.java”

- Here, a single ‘.java’ file is being used. Alternately, an additional HTML file may also be used for taking in values from the user.
- Cookies are small text files that are sent by the server to the browser. This is used for session handling which is very important in web development. They house the client’s state information. Cookies are useful and an example for this is in the “Online Shopping” process. In online shopping, a cookie can store information regarding the client such as his/her name, address etc. This ensures that the client need not have to enter this information again the next time he/she shops online at the same store. For each cookie, the following attributes are present.
- Name, Value, Expiry Date, Domain & Path Information. In this program only the first two attributes are being considered.
- A servlet can write a cookie on the client’s machine with the addCookie () method in the response. The data of this cookie is then included in the header part of the HTTP Response. The “Refresh” button of the Web Browser has to be clicked after submitting the values to view all the cookies.

HTML FILE

```
<html>
  <body>
    <center>
      <form action = http://localhost:8080/servlet/AddCookieServlet
        method="post">
        <b>Enter a value for MyCookie:</b>
        <input type="text" name="data">
        <input type="submit" value="Submit">
      </form>
    </body>
  </html>
```

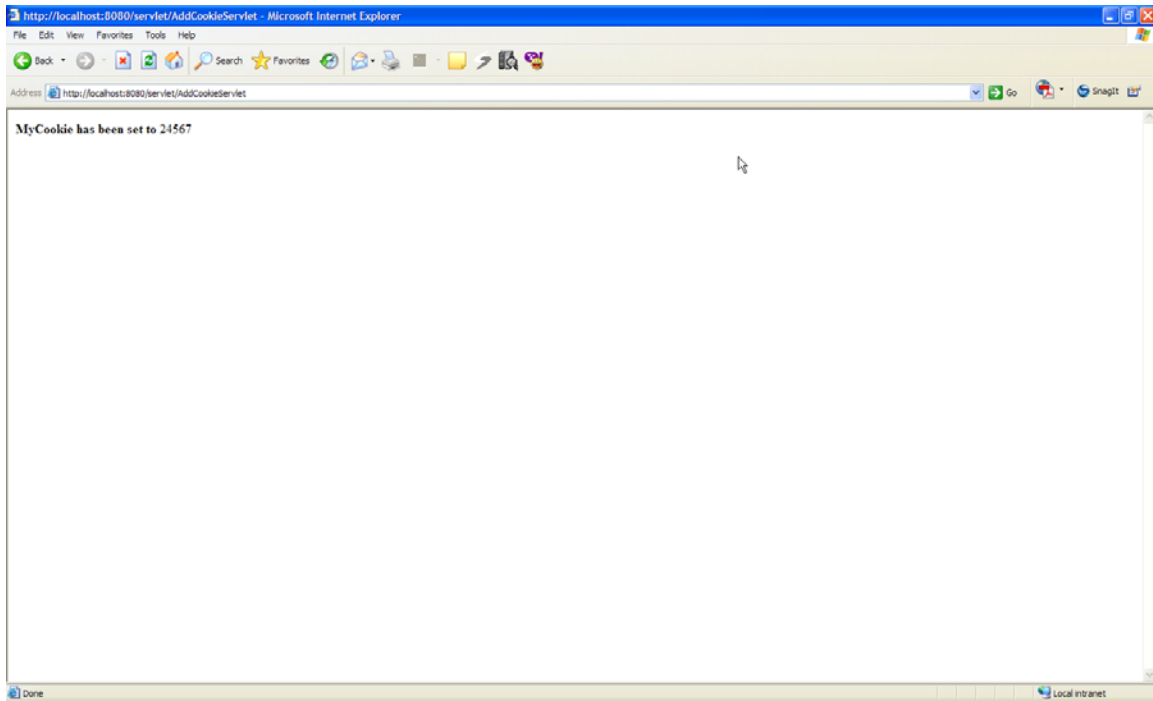


JAVA FILE

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class AddCookieServlet extends HttpServlet
{
    public void doPost(HttpServletRequest req, HttpServletResponse res) throws
    ServletException, IOException
    {
        String data = req.getParameter("data");
        Cookie cookie = new Cookie("MyCookie",data);
        res.addCookie(cookie);
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        pw.println("<b>MyCookie has been set to </b>");
        pw.println(data);
        pw.close();
    }
}
```

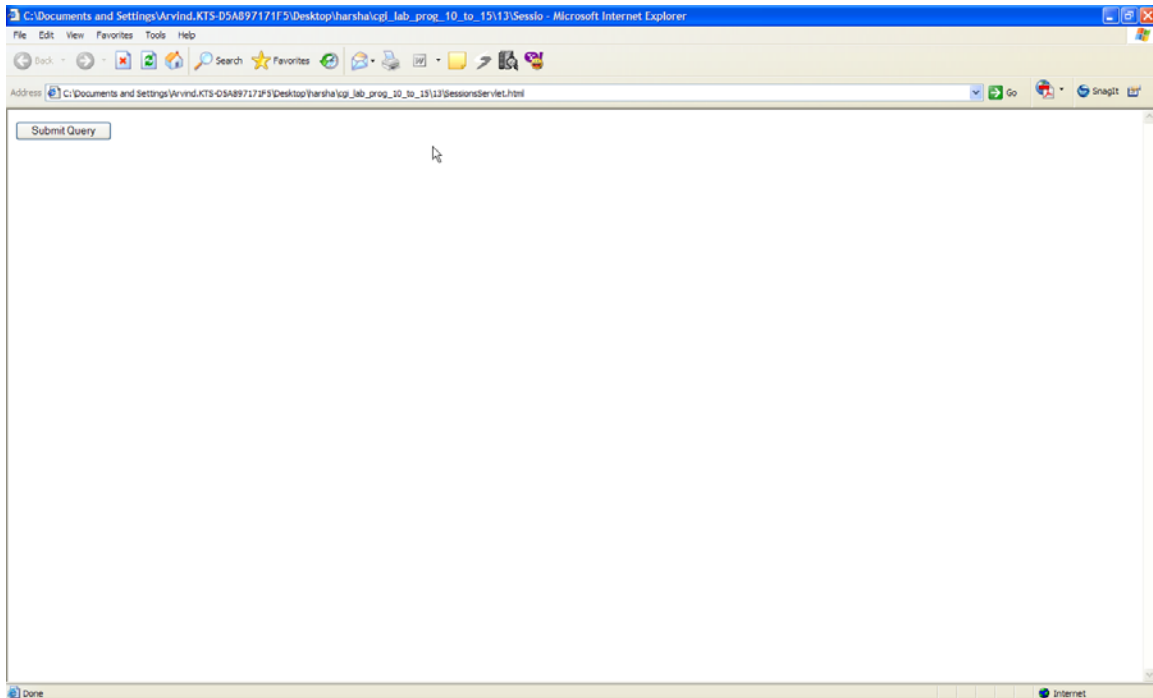


13) PROGRAM TO CREATE A SESSION AND DISPLAY SESSION INFORMATION VIZ SESSION ID CREATION TIME AND LAST ACCESSED**Steps: “p13.java”**

- A Session provides a mechanism to store the state information between multiple interactions between a Client and a Server. They are used as HTTP is a “Stateless Protocol”, i.e. each request is independent of the other.
- Here we try to get the current session first. If there is no session currently, server will automatically create a session when you request. The names of the methods used are self explanatory and we will skip the details of each of these methods.

HTML FILE

```
<html>
  <body>
    <form action="http:\\localhost:8080\\servlet\\SessionsServlet"
      method="get">
      <input type="submit" name="connect server">
    </body>
</html>
```



JAVA FILE

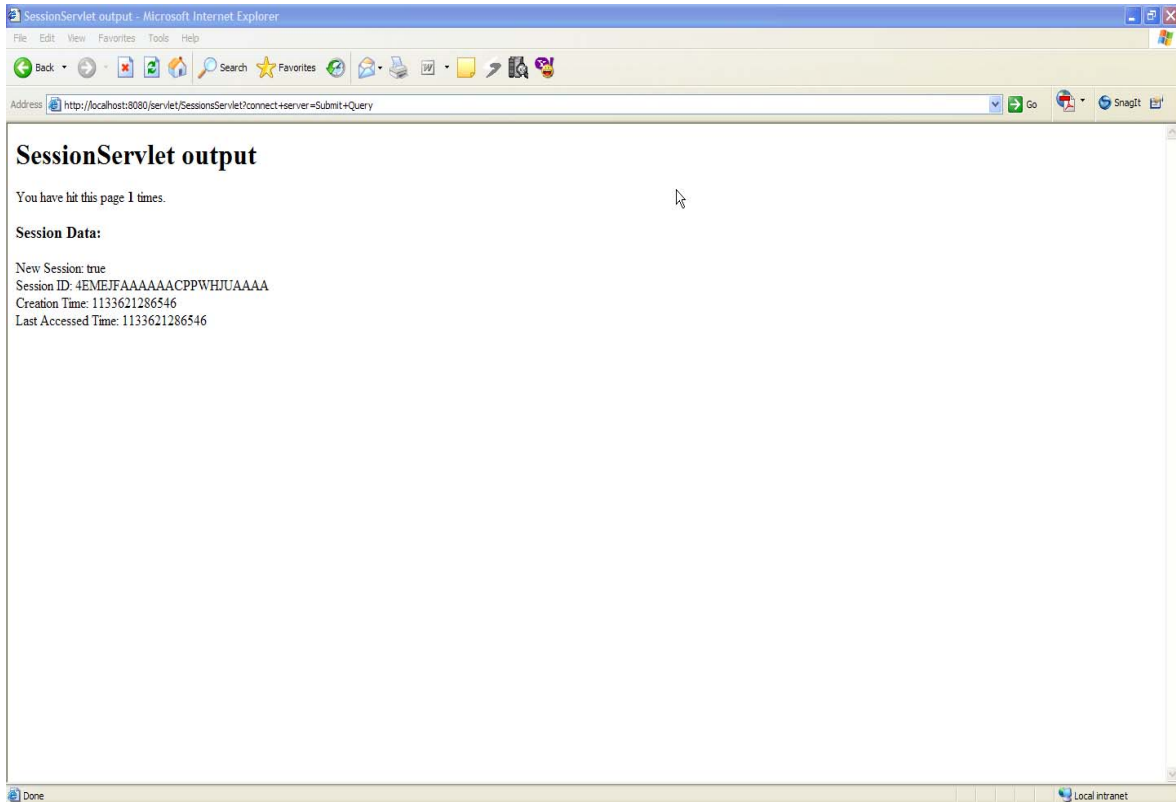
```

import java.io.*;
import java.util.Enumeration;
import javax.servlet.*;
import javax.servlet.http.*;

public class SessionsServlet extends HttpServlet {
    public void doGet (HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        HttpSession session = req.getSession(true);
        res.setContentType("text/html");
        PrintWriter pw = res.getWriter();
        pw.println("<HEAD><TITLE> " + "SessionServlet output " +
"</TITLE></HEAD><BODY>");
        pw.println("<h1> SessionServlet output </h1>");
        Integer ival = (Integer) session.getValue("sessiontest.counter");
        if (ival==null) ival = new Integer(1);
            else ival = new Integer(ival.intValue() + 1);
        session.putValue("sessiontest.counter", ival);
        pw.println("You have hit this page <b>" + ival + "</b> times.<p>");
        pw.println("<p>");
        pw.println("<h3>Session Data:</h3>");
        pw.println("New Session: " + session.isNew());
        pw.println("<br>Session ID: " + session.getId());
        pw.println("<br>Creation Time: " + session.getCreationTime());
    }
}

```

```
        pw.println("<br>Last           Accessed           Time:           "           +  
session.getLastAccessedTime());  
        pw.println("</BODY>");  
        pw.close();  
    }  
}
```



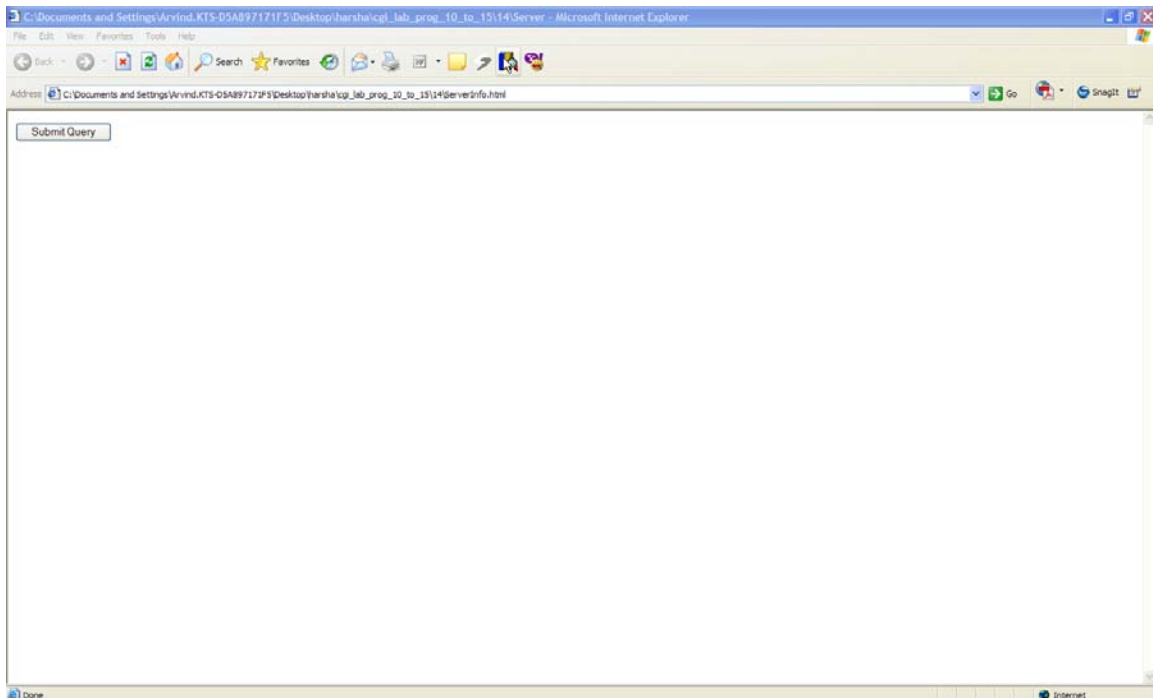
14) PROGRAM TO REQUEST SERVER INFORMATION VIZ REQUEST METHOD,URL,PROTOCOL AND REMOTE ADDRESS

Steps:”p14.java”

- We make use of the built-in methods to display information regarding the Server. The names of these methods are self-explanatory .
- But one point to be noted is that URI is different from URL. For example if the URL typed is: - www.heythere.com/content/chapter1, the URI here is stuff after www.example.com/

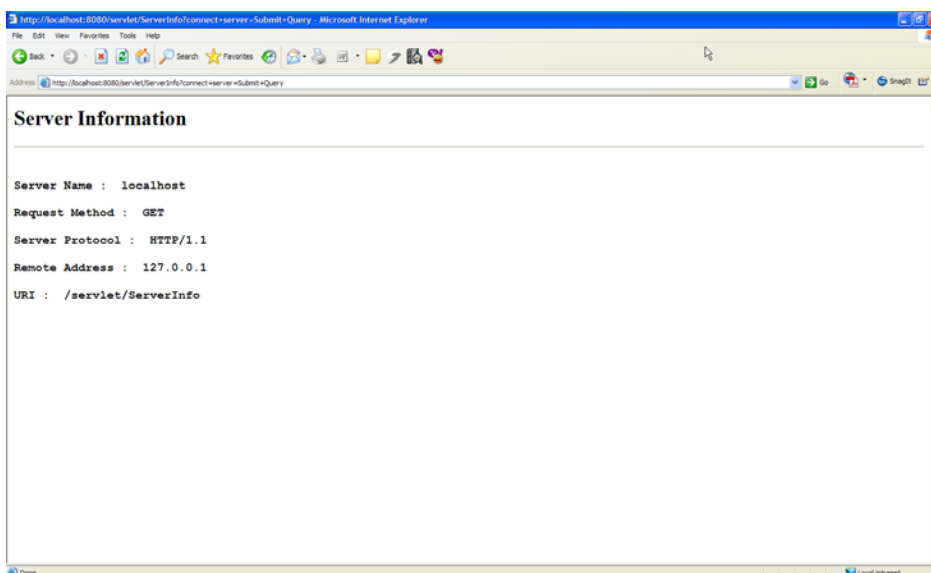
HTML FILE

```
<html>
  <body>
    <form action="http://localhost:8080/servlet/ServerInfo" method="get">
      <input type="submit" name="connect server">
    </body>
</html>
```



JAVA FILE

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ServerInfo extends HttpServlet
{   public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        String method = req.getMethod();
        String uri = req.getRequestURI();
        String protocol = req.getProtocol();
        String remoteuser = req.getRemoteUser();
        String remoteaddr = req.getRemoteAddr();
        String servname = req.getServerName();
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        pw.println("<h1>Server Information</h1>");
        pw.println("<hr><pre><h3>");
        pw.println("Server Name : " + servname + "<br>");
        pw.println("Request Method : " + method + "<br>");
        pw.println("Server Protocol : " + protocol + "<br>");
        pw.println("Remote Address : " + remoteaddr + "<br>");
        pw.println("URI : " + uri + "<br>");
        pw.println("</h3></pre>");
        pw.close();
    }
}
```



15) PROGRAM TO ACCEPT USERNAME AND ADDRESS AND DISPLAY THEM IN A WEB PAGE BY PASSING PARAMETERS

Steps:“p15.java”

- The “Name” and the “Address” details are taken from the users by making use of the text box “t1” and Text area “t2” respectively. A text area is a bigger text box with a “scroll” feature present if the input goes beyond the size of the text area. The values entered by the user are then displayed back on the browser.

HTML FILE

```
<html>
  <body>
    <form action="http://localhost:8080/servlet/PassParameter" method=get>
    <center>
    <h1>
    Personal Details
    </h1></center>
    <hr>
    <table>

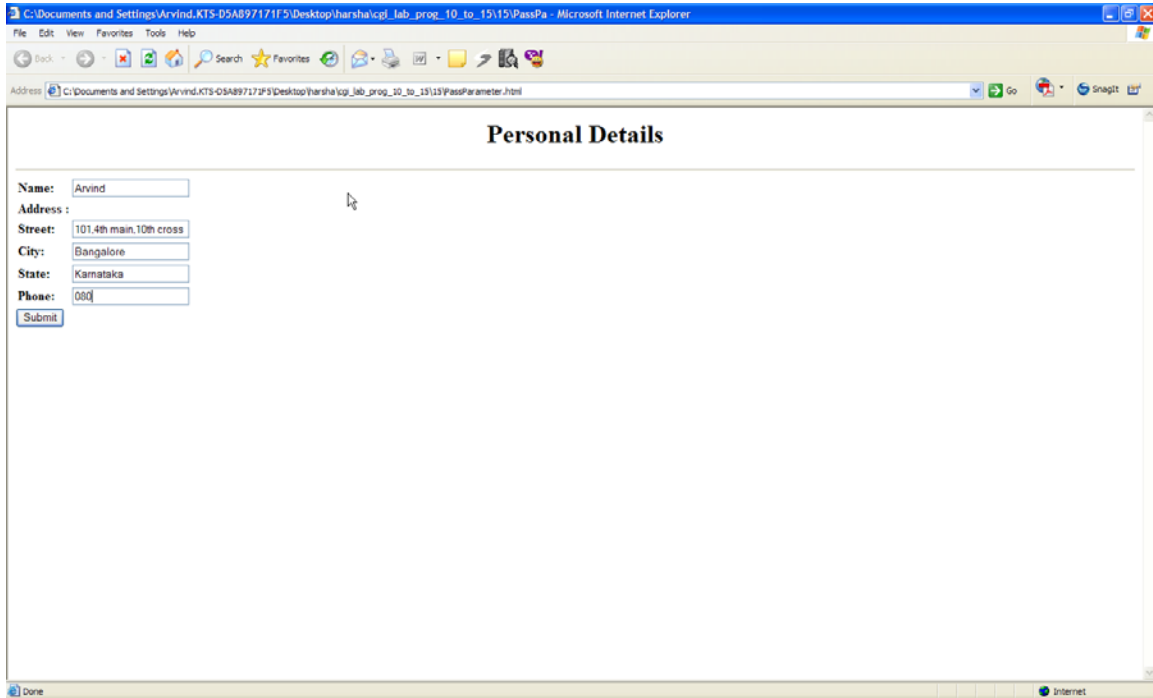
    <tr>
      <td><b>Name: </b></td>
      <td><input type=textBox name="user"></td>
    </tr>
    <tr>
      <td><b>Address : </b></td>
    </tr>
    <tr>
      <td><b>Street: </b></td>
      <td><input type=textBox name="street"></td>
    </tr>
    <tr>
      <td><b>City: </b></td>
      <td><input type=textBox name="city"></td>
    </tr>
    <tr>
      <td><b>State: </b></td>
      <td><input type=textBox name="state"></td>
    </tr>
    <tr>
      <td><b>Phone: </b></td>
```

```

        <td><input type=textBox name="phone"></td>
    </tr>

</table>
<input type=submit value="Submit">
</form>
</body>
</html>

```



JAVA FILE

```

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class PassParameter extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        Enumeration e = req.getParameterNames();
        pw.println("<h1>Personal Details</h1>");
    }
}

```

```
pw.print("<hr><pre>");
while(e.hasMoreElements())
{
    String pname = (String) e.nextElement();
    pw.print(pname + " = ");
    String pvalue = req.getParameter(pname);
    pw.println(pvalue + "<br>");
}
pw.print("</pre>");
pw.close();
}
}
```

