Eszterházy Károly Főiskola
Matematikai és Informatikai Intézet

Developing Dynamic Web pages

Király Roland
Eger, 2011

1. Introduction

1.1. Content management systems

Deployable content management systems. With the spread of content management systems, the view that in order to create web pages it is sufficient to download the CMS system, developed for this purpose by a third party, then to copy it to the location provided by your hosting service, is becoming more and more general. After some configuration you can launch the web page and users are ready to go.

In case you do not want to spend long days nailed in front of your computer, spending your time with typing program codes to visualize one or two page long texts, or to write a blog for the world, then prefabricated ”Instant” web pages offer a convenient solution.

At least that is the case when CMS provides the service you want to use it for. If it does not, then it might be better to write the web page or module you need. If it is not enough, then there is yet another important reason for which it is important to learn how to create dynamic web pages.

We should raise the question how CMS systems come into being. How are instant web pages created? Certainly not by downloading a prefabricated CMS builder from the Internet. If it were so, then we would still have the dilemma of how the downloaded CMS builder was created.

No matter how much we ruminate this, we must know programming and in many cases when CMS is not sufficient, we have to create the programs that satisfy the demands of users. Because no matter how fiercely the big family of programmers oppose this, it is true enough that the demands, set up by the users, sustain and promote this branch of learning.

What is Content Management System, CMS for short in English? It is a partially built web page which we can launch within minutes

after some configuration like password settings and other smaller tricks.

A general CMS comprises of the following components:

- Login system
- Administration interface for managing content
- News menu and blog
- Image gallery
- Favorites and RSS news service
- Forum (this is generally an integrated external module)
- Calendar and its services
- Event handling system
- Adjustments of the appearance of the page
- Other special functions peculiar to the particular CMS

While building web pages, we must be aware of the attributes of all the functions above. It is important to see these functions running in practice, as that is the only way we can build similar modules.

In this lecture note we are going to install WordPress system. I have chosen this system because on the one hand it is pretty wide spread and on the other hand it is famous for its five minute installer.

1.2. Installing CMS

First you need to download the current version of Word Press. You do not necessarily need to choose this CMS. In case my amiable reader insists on another system, you can try that out, or you can try out several other systems as well.

In case of Word Press you can start download on the http://wordPress.com web page, by clicking on the download button.

How to install? In order to install you need a location provided by a hosting service, or you need to install a CMS system that provides a web server, PHP runtime environment and database storage services. For database management system you should choose a system based on SQL, if possible one based on MYSQL. This is important as in this lecture note we use MYSQL language queries.

Currently XAMP and WAMP servers are widespread among programmers so we use a version of XAMP. Please, notice that for building the server side we are also using an ”Instant” Program, taking the risk of never learning how to install a server. So, if you do not find a free hosting service, which is highly unlikely, then download a version of XAMP from the http://xamp.org web page and install it. (In Windows after some clicking the web service is ready.)

In case of a Linux operating system always read the file about installing before you start work. By the way, setting up the Linux version takes no longer than a couple of minutes either.

After launching the web server, your only task is to copy the Word Press files into the htdocs folder. This is the default folder for storing our web pages in the XAMP system, but it can change during configuration.

Launch the installer of the program in your favorite browser. It asks for the username and password of the database. You must also give the name of the administrator of the page and his password and a couple of other information which the program can not figure out by itself. (if it could figure these out, then there would be no need for users..).

As soon as you finish installation, the program informs us about it and you are ready to begin ”fine-tuning” the program. You can set the appearance attributes and the language.

You can add users who can use our would-be page, you can reorganize the modules and do anything allowed for us by the page – in case of more serious systems you can even use drag and drop technology for this purpose.

That is all. If you follow the steps enlisted in the program's README file, you can set up an operable web page, steeled with all the modules that can be commonly found on the Internet, without programming.

Unfortunately this web page is not really suitable for providing special functions, namely it only knows functions its programmers have built in its system.

Developing the system further.

Using and configuring the freshly installed web page is very comfortable and simple, but you can run into problems if you want something special.

In case you have created the page for only one user - he has paid for the program - who realizes that he would need peculiar functions in the program not contained by the CMS you have installed, you find yourself in a tough situation.

An experienced programmer, in this situation, looks into the source code and if he is lucky he can alter it in a way that makes it possible for the new function to be integrated.

Several CMS includes improvement options, but it is never as simple as altering our own programs.

You have to understand the source codes of other programmers and, you have to alter them without ruining the original functions, while integrating the new ones.

It is a difficult task even for experienced programmers. Of course this does not make CMS bad and you can use them comfortably.

These software are very good, but only for the purpose they have been built for.

Keeping this in mind you should think over whether the particular system is appropriate for us on the long run as well. If it is, then use it bravely, however, if you are not sure, then you should start developing.

This lecture note helps you with it, as it introduces its readers into the science of programming dynamic web pages, displaying the world of CMS systems, only known by developers, step by step.

1.3. Review questions

1. What does the word dynamic refer to in the world of web pages?
2. What components does a web server contain if you run database based PHP code web pages on it?
3. What tasks does a web server do?
4. Can you modify the configuration of a web server?
5. What does CSM stand for?
6. What is the advantage of module technology?
7. What does modular build up mean in case of web pages?
8. What protocols do web pages use?
9. What is the role of a web server while running PHP codes?
10. How can you get a public domain address?
11. What are main functions of a CSM based system?

animaciok/TAMOPweb0006.avi
animaciok/TAMOPweb0007.avi

2. Using HTML and PHP

2.1. Preparing programs of web pages

In order to edit web pages you must know the basics of HTML language. You must be able to shape the output of programs, namely the content, displayed in the browser.

HTML language can be used to display texts, tables and images aesthetically. With its help, you can define character encoding, special characters and we can also command the client side browser program not to use the default format settings.

It offers you the chance to create web pages with a unique appearance.

Unfortunately, the HTML language is not capable of handling interactions. You can not use selection or iteration in HTML source texts.

You can not connect to databases, prepare functions or methods.

The disadvantage of web pages, which are built using only HTML, is that their operation is static, implying that the texts, images and any other information displayed on the web page do not change unless the developer of the page rewrites the HTML files with the help of a text editor.

Consider what this would mean in case of a web page that describes currency rates and their changes in diagrams.

Solving this problem is almost impossible. For handling changing content and interactions it is by all means necessary to have a programming language and a program. Furthermore you need databases –for storing data-switches, loops, functions and methods.

To run the programs you need to install a runtime environment, which is able to debug, interpret and run the source texts written in a text editor.
Protection of the programs and source codes is also an important task as users download the content of the web page onto their computers. The downloaded text is interpreted and run by the browser program installed on the operating system, the stored and described information is displayed.

The browser program handles sound and image files with the help of plug-ins, it is able to run Flash and JavaSript based programs.

However, it is not able to run PHP codes or codes written in other programming languages, but there is no need for that at all.

No one expects the user to download the interpreter of the language, to configure the runtime environment and to test it, then to correct the mistakes that come up.
In case of using the PHP language, the programs of web pages run on servers.

The operator of the server configures the running environment and the database management system. In many ways it is better than using client side programs.
Interpreting and running the code on the server side calls forth serious sparing of resources on the client side computers. Data storage takes place on the server, too, which is also an issue that should not be neglected regarding the sparing of resources, not to mention security.

The source code of programs is not visible on client computer as only the result of a program, the generated HTML content is downloaded.

Running programs is a safer task since the source code and the logic of the source code is not visible for harmful users. Picture 2.1 shows the operation of the client and the server

Embedding PHP programs. PHP programs run on servers, unlike HTML texts which are interpreted by the client computer. Data is stored on the server, but it is displayed on client computers.

It seems like a contradiction at first. Various parts of the source code run in different places using various applications, but the source code itself can be stored in one file.

Most interpreters and runtime environments are all capable of detecting the source texts which they are able to interpret, so you can freely embed PHP programs into the HTML source.

Picture 2.1. Client server connection

```
<BODY>
<?
  echo
   "Currently registered user:".$regUser;
?>
</BODY>
```

Program 2.1. PHP embedded in HTML source text

It can happen the other way round as well. You can generate HTML output into PHP programs for the computer browser.

```
for ( $ i =0; $i <10; $ i++)
{
    echo "<TABLE><TR><TD>$i .sor</TD>
                                                    <
/TR></TABLE>" ;
}
```

Program 2.2. HTML code generated from PHP

Or you can completely separate the programs from the HTML code. No matter which option you choose, the point is to organize your programs in an appropriate way into the appropriate sized modules because it is easier to correct and alter codes that are well constructed.

Program texts are divided into parts either based on function or type in order to keep the source code manageable.

It is worth placing the pages, appearing in the client computer's browser, in different modules (most of the time it means a separate file)

It is also important, from the aspect of easy management, to organize the codes, which implement database management, login or generate various lists and all the code segments that belong together, into functions, the functions into modules or classes.

2.2. Testing the runtime environment

Before creating the first PHP based web program, you should check whether the runtime environment installed on the server is operable.

In order to test you must log in to the server. There are two ways two do this. Supposedly the computer on which we are developing and the one that runs as the server are not the same, so the first thing you need to do is to establish a connection.

To move files, use a file manager program which can use FTP protocol, but to establish terminal connection, install SSH or Telnet client program. The latter is not suggested as while running Telnet protocol our data travels through the web without encryption.

Create an index.php file in the server computer's web folder. Be careful because the web server handles files with different extensions in different ranking. If you have a file in the folder called index.html then that will not run the one with php extension.

The order they run, depends largely on the settings of the web server.

Picture: Folders of the web server.

In case of Linux operation system the folder you need to place the file in is one of the subfolders of var/www.

In case of user folders created in a Linux system this is either the public_html folder of the user's home folder's root or one of its subfolders.

The user can run as many web pages as many subfolders he has in the public_html folder.

The logic of operation is the following: The client computer's browser program establishes a connection to the server.

The connection is established on virtual port 80 of the server computer by default [1]. Due to an incoming call the server looks for the folder referred by the browser, then the index.php file in the subfolder. It interprets and executes the source code in the file. If it finds PHP code in the file it executes it with the assistance of PHP's interpreter.

The final HTML code which was either found in the file or generated by the PHP program is sent to the client computer, which interprets and displays it in the client side browser.

Picture 2.2. The URL of the web browser

In order for all this to take effect you have to type the following URL into the client's browser: http://www.nameofwebpage.hu (Picture 2.2.)

Note 2.1. The address would be valid in case of a registered domain in the .hu domain. . .

In case of the users' web folders the page is identified by the name of the server, the user and the route to the subfolder where the index file can be found.

```
http:\\nameoftheserver.hu\~username\subfolder
```

In this case it is compulsory to leave out the www part. The ~ character in the address comes before the name of the user and implies that the index file must be found in a user folder.

After creating the index file, open it with our favorite text editor program and type the following few lines:

```
<?
phpinfo ( ) ;
?>
```

Program 2.3. Checking the web server

Phpinfo is an embedded function of the language which displays the settings of the server, the variables, which characterize the

runtime environment and any other information that can be necessary regarding programming and operating, in a table.

The table besides supplying us with information also shows whether the server operates properly. We should see something similar in the browser to the output in picture 2.3

Before getting into the rules of PHP language, let us prepare a simple program that besides motivating us, serves as an example in case of constructing more complicated programs.

It is true that PHP is an OOP [2] language. It is also true that you can completely separate the HTML text from the source code and that is what the developers of CMS do, however, let us embed the source text into the HTML code.

Picture 2.3. Information panel of the web server

This solution can be used in the long run as well if you do not want to write OOP based programs, or if you make minor developments, but do not forget that you can recycle OOP codes and they are easier to maintain and develop further.
The first step is to create a program that displays a short text message in the browser. Let our displayed text be the classic Hello World. The source code using HTML language only is the following:

```
<!DOCTYPE HTML PUBLIC "//DTD HTML4.01//EN">
  <HTML>
     <head>
        <TITLE>testprogram</TITLE>
          <meta http-eqquiv content-type
              content="text /HTML;charset=UTF-8">
         <meta http-equiv=" content-style-type
                                              "conte
nt="t ext/css">
   </head>
  <body>
      Hello World
  </body>
</HTML>
```

2.4. program. HTML header

Since this program only contains HTML tags and the text to be displayed, let us develop it further.

How could we display the text several times consecutively one under the other, numbering the lines? In case of a low number of lines the copy paste technology can help us, but what can you do in case of 1000 lines, or if the text you have to display on the client side screen is not constant?

You can write a program, using a loop and you can get the data from databases.

The strength of dynamic web pages lies exactly in this technique.

Let us create a short, but expressive PHP program. You can store data in a database and you can display it based on HTML. This way the text meant to be displayed by the browser can be generated by the PHP program based on the data coming from the database.

Since we are not able to use databases yet, let us store the text in an array (about arrays you can read later). Then with the help of a simple for loop you can generate the proper HTML tags.

```
<HTML>
<head>
<title>testprogram</title>
<meta http-equiv="content-type=
          "text/HTML;charset=UTF-8">
</head>
  <body>

 <?
  $szoveg = array("Europe", "Africa",
         "Australia", "America", "Asia");
  $db = count($szoveg);
  for ($i=0; $i<$DB; $i++)
```

```
  {
    echo "Hello $szoveg[$i]!<br>";
  }
 ?>

  </body>
</HTML>
```

Program 2.5. Generating HTML

The source code above is much more universal than the static HTML version, as it gets data from an array and the output (Picture 2.4) is generated intelligently from the data you work with and the prefix of the sentence to be displayed.

Picture 2.4. Output of the continents program

Do not be afraid of the special elements of the source code. The code seems complicated at first, but it is very similar to C, C++ languages, or the language used in Linux shell scripts.

The $ signs represent variable and the HTML tags in the I/O statement can be familiar to us by now.

What you have to get used to is that while creating PHP programs you have to use various programming or markup languages alternatively. Moreover, you must be able to integrate them. The output of your program can be seen on picture 2.4.

2.4. Review questions

1. Where does the output of the PHP program appear?
2. What are the steps of running a PHP based web page in a browser?
3. What does the abbreviation URL stand for?
4. In what programming languages can you address the MYSQL database management system?
5. How can you embed PHP source texts into HTML code?

6. What is the name of a web page's main file (by default)?
7. What extension do the files containing PHP programs have?
8. What is the order the web server uses the following extensions? .htm, .html, .php
9. Does PHP interpret object code?
10. Where is the program of a web page stored by default?

animaciok/TAMOPweb0001.avi
animaciok/TAMOPweb0002.avi
animaciok/TAMOPweb0008.avi
animaciok/TAMOPweb0009.avi
animaciok/TAMOPweb0010.avi

3. PHP basics

3.1. Handling the output

As the writing of PHP programs, the use of various server side applications while programming do not take place in the conventional "one development tool-one program" style, let us begin by getting acquainted with the language in an unconventional way, too.

We are not looking at the control structure or the instruction set one after the other.

Instead we observe how the output of our programs can be generated for the client computer's browser. How we can display the value of variables or expressions embedded into HTML texts.

You have to type the PHP program sections between the <? ?> or the <?php ?> tags,
Since that is how the web server can deduct that a PHP language source text is coming up and where it ends.

In PHP language to display strings you can use echo or print functions. Echo makes the source codes of programs easier to read and a bit faster as well (due to less brackets).

```
<?

  echo "Hello";
  echo "$i.sor";
  echo $i.".sor";

?>
```

Program 3.1. Displaying texts

The content to be displayed must be typed between " " and ' ' characters. Both forms have a different meaning and have a different effect on text type data, on variables ($a) and on expressions ($a + 2).

Variables typed between ' ' characters are not substituted, instead of their value, their identifier within the code is displayed.

```
<?

  $a = 2;
  echo '$a';

?>
```

Program 3.2. Use of single quotes

The result of the source code is not 2 but the $a text. This happens because the characters written between the ’’ appear on the screen, namely on the interface of the client computer’s browser, as they had been typed by the programmer, meaning that every typed sign is forced to a character.

In case we would like the content of the variable to appear on screen, we have to use the “ ”quotation marks. To tell the truth if you do not want to display any other text besides the content of the variable, then the double “ can be omitted.

```
<?

  $a = 2;
```

```
  echo "$a<BR>";
  echo $b;

?>
```

Program 3.3. Use of quotation marks

As it is apparent in the source code, all the commands end with ;. Its omission leads to syntax error, which is not too fortunate in case of web pages, where programs are interpreted by the server each time they are executed and the error appears on the screen every single time.

This operational principle, as we will see it later, involves various possibilities but at the same time besides its potential it can lead to serious errors which are displayed to the user by the system at once.

Displaying complex data and expressions.

It is much more complicated to display and get the right format in case of complex data, arrays or expressions.

```
<?

  $t = array(1,2,3,4);
  $db = count($t);
      //a $t number of array elements
  for ($i = 0; $I < $db; $i++)
  {
    echo "$i + 1<BR>";
  }

?>
```

Program 3.4. Displaying data

In the source code above (3.4) the loop does not display the value of the $i + 1 expression but the value if the $i variable and after every value the +1 text.

```
  0 + 1
  1 + 1
  2 + 1
  3 + 1
```

Program 3.5. Result of the display

This happens because the $i+1 in the display command is not evaluated.
First you must evaluate expressions before displaying them.

The right solution is to separate the expression from the text with another echo. In this case the display command displays the value, namely the result.

```
<?

  for ($i = 0; $i < $db; $i++)
  {
    echo $i + 1;
    echo "<BR>";
  }

?>
```

Program 3.6. Displaying passages

We can also assemble the text to be displayed from fragments. For this purpose we do not use the + sign in PHP, common in C, C++ or in C#, but the “.” operator. Unfortunately, this operator makes the code hard to read but with some practice you can quickly get accustomed to this syntax.

```
?>
```

```
  $t = array(1, 2, 3, 4);
  $db = count($t);
  $e = ".array element=" ;
  for ($i = 0; $i < $db; $i++)
  {
    echo $i.$e.$t[$i]."<BR>";
  }

?>
```

Program 3.7. Assembling text fragments

The result of running the code:

23

```
1.array element = 1
2.array element = 2
3.array element = 3
4.array element = 4
```

Program 3.8. List of array elements

As you can see it in the source code it is a bit complicated to display data but do not be surprised. We have embedded multiple languages together using multiple systems.

After acquiring the use of PHP programs we can use this property to our advantage to solve problems like generating source codes or preparing dynamic database queries.

3.2. Variables and Data

PHP is a weakly typed language. You do not have to declare simple scalar variables (integers, strings, fractions or characters) used in programs. However besides the omission of declaration

there are restrictions regarding the use of variables but they are not too complicated.

Before using a variable you must assign its value. If you do not do this the runtime environment stops running the program and displays an error message on the screen (or on the location set in the server configuration).

```
<?

  $prefix = ".value of array element:";
  for ($i = 0; $i < 10; $i++)
  {
    echo $a + $i;
    $a++;
  }

?>
```

Program 3.9. Declaration of value

However in case of complex data structures like arrays, records, classes or files you must introduce their type.

```
<?

  $t = array(1, 2, 3, 4, 5, 6);
  $text = array("Asia", "Africa") ;
  classtest class
  {
    . . . the inside of the class . . .
  }

?>
```

Program 3.10. Defining classes

Being weakly typed does not mean that the system does not deal with or handle types. Rather the handling of variables is more lenient, which holds a variety of possibilities when programming dynamic web pages.

Do not forget that displaying number and various string type data takes place in form of a text, as the browser gets simple HTML code from the server side program.

The data and expressions are converted to text all the time when they are displayed, this way forming and assembling is not a problem even if you omit direct data conversion. This option is the result of being weakly typed. You can use a variable for multiple purposes, storing various types of data.

The $a = 1 declaration can be followed by the $a = "text", then after the $a = 5 value change the $a variable can be used in an expression as well.

The type system of the language is permissive but be careful, more possibilities mean more possibilities of error. You can make mistakes which you do not recognize while testing the program but the user will generate them for the first time he uses the program.

Full-scale testing is very complicated even in the case of very simple programs.

Source codes containing a couple of branches can result several millions of execution paths. Not to mention that testing is a very expensive thing. Be well-considered and prudent with types and variables in order to avoid spending long hours correcting the source code.

3.3. Complex data

In order to understand the use of complex data structures better, let us group the types of the PHP language. There are three types. The most primitive one is scalar, namely numbers characters and all other primitive types.

The other large group is the group of complex data. Arrays, lists, strings and classes belong here.

The language can handle arrays in two different ways. We can refer to arrays as indexed vectors, namely as hash tables. This type is also called associative array.
In case of associative arrays you can access a certain element through its key. In practice this means that you can not access the particular element with the $array[$index], or the $array[1] formula but with the $array["key"] formula.

When you handle arrays as vectors we can refer to its elements simply by their index. To access any of the elements you use its index so if you know the location of an element within an array (its index) you can refer to it with the $arrayvariable[$index] formula.

Naturally, PHP also knows the concept of multi dimensional arrays.

```
$t[0][0] = "0.0";
$t[0][1] = "0.1";
```

Program 3.11. Multi dimensional array

You can refer to elements in all dimensions as you wish, so you can use indexes in one dimension and the associative way in another one.

```
$t [1]["first"] = 1;
echo $t[1]["first"];
```

Program 3.12. Using array elements

You can create array type variables with the help of the array and list functions or in a direct way by specifying the elements.

```
$t1 = array(1, 2, 3);
$t2[0] = 1;
$t2[1] = 2;
$t2[2] = 3;
```

Program 3.13. Specifying elements

You can also easily create associative arrays by defining element index pairs or with the array function.

```
$a1["first"] = 1;
$a1["second"] = 2;
$a2 = array("first" => 1, "second" => 2);
```

Program 3.14. The array keyword

Arrays a1 and a2 contain the same elements, only their names differ.

Let us have a look at a more difficult example that vividly shows us the use of arrays. In this example program you fill an n x m sized matrix with random numbers.

You get the elements of the matrix with the help of two embedded loops.

You directly specify the stored values and create them with a random number generator.

```
<?

  $M = array();
  $n = 10;

  for ($i = 0; $i < $n ; $i++)
  {
    for ($j = 0; $j < $n; $j++)
    {
      $M[$i][$j] = rand(1, 9);
    }
  }
```

```
  for ($i = 0; $i < $n ; $i++)
  {
     for ($j = 0; $j < $n ; $j++)
     {
        echo $M[$i][$j]." ";
     }
     echo "<BR>" ;
  }

?>
```

Program 3.15. Filling an array

Displaying of arrays is thoroughly discussed in the loops subsection of the control structures section but we can show you the result of program 3.15.

```
8 2 4 6 4 5 8 3 4 7
8 1 2 6 3 1 7 4 8 4
3 5 7 8 4 6 4 7 6 7
5 5 9 8 1 3 3 9 6 7
6 4 7 8 9 1 8 7 4 7
2 6 2 8 5 6 4 8 4 1
6 8 5 5 6 6 7 9 5 4
6 2 7 3 9 7 3 8 5 7
5 7 3 7 5 8 4 9 6 7
1 3 6 5 7 3 2 5 2 6
```

Program 3.16. Displaying a matrix

Filling and displaying the matrix have been carried out in separate iterations, as it is a basic programming error to fill and display arrays in the same loop.

A much better way of list processing is to use the foreach control structure.

3.4. Control Structures

The PHP language contains the control structures, branches, loops and the foreach list handler which are regular in C, C++ and C# programs.

Let us have a look at the PHP version of control structures.

Branches. The language contains the primitive and complex forms of conditional branches. You can write if {} else {}, or switch type branches.

```
<?

  $a = 2 ;
  if ($a % 2 == 0)
  {
    echo "$a even";
  }
  else
  {
    echo "$a odd";
  }

?>
```

Program 3.17. Conditional branch

Of course in case of a single command you can omit the { } characters but their omission is not favored as if you add new commands to the existing program it is very complicated to add the { } characters afterwards.

Naturally, you can write branches with multiple paths introducing elif branches or with the use of the switch conditional branch. Using elif paths is clumsy, so if possible chose the switch type branch, which is illustrated in the following example:

```
<?
```

```
  $a = 2 ;
  switch ($a)
  {
    case 1  : echo " Monday " ; break ;
    case 2  : echo "Tuesday" ; break ;
    case 3  : echo " Wednesday " ; break ;
    case 4  : echo " Thursday " ; break ;
    case 5  : echo " Friday " ; break ;
    case 6  : echo "Saturday" ; break ;
    case 7  : echo " Sunday " ; break ;
    default :
       echo " wrong index" ; break ;
   }

?>
```

Program 3.18. The switch control structure

The particular paths in the switch conditional branch are introduced with the case keyword.
At the end of each path there is the break word and a ;. If you omit it, then after the particular path all other paths that follow are executed until the next break;.

The word default introduces the else branch and it can be omitted. (It is obvious that this program snippet can be executed more effectively with the help of an array and a loop command.)

The switch command can be used exquisitely to organize the menu handling of web pages. We will return to this option later.

Loop commands. There are three types of loops in the PHP language, completed with the foreach command, which can be used for processing arrays or lists.

```
<?

  for ($i = 0; $i < 10 ; $i++)
  {
    echo "$i <BR>";
  }
```

```
  echo "<BR>";
  $i = 0;

  while ($i < 10)
  {
    echo "$i <BR>";
    $i++;
  }

  echo "<BR>";
  $i = 0;

  do
  {
    echo "$i <BR>";
    $i++;
  } while ($i < 10)

?>
```

Program 3.19. Iterations

The three loop commands above execute the same task, namely they display the numbers, each one in a new line, from 0 to 9 on the screen. In while and do-while loops you must make sure that you can exit them. Do-while, as in C, C++ and in C# programs, contains the condition for remaining in the loop and not the one for exiting it.

Foreach slightly differs from the regular C format. In PHP we do not use the foreach (scalar variable in list expression), but the

```
foreach (list as listexpression) …
```

format.

```
 $t = array(1, 2, 3, 4, 5, 6);
```

```
foreach ($t as &$i)
{
  echo $ i ;
}
```

Program 3.20. Processing Foreach Lists

The &$t is necessary because in a foreach command we do not refer to elements of an array or a list but to the array itself , so we must not handle it as a scalar type variable but as an array type.

3.5. Subroutines

The basics of the language are not complicated, especially not for one who has already learnt another programming language.

However, to create web pages that operate with data from a database you must acquire the use of functions and methods at all cost.

Note 3.1. From here on, subroutines will be uniformly called functions. Where necessary, we will explain the handling of types and return values as well…

In order to create functions we must be aware of the concepts of formal and actual parameters and the different ways of calling functions.

Using functions. Subroutines are introduced with the function keyword. Then comes the list of parameters between (), followed by the function body enclosed in {} curly brackets.

The () curly brackets must never be omitted not even in the case of functions without parameters.

The function body contains the commands and after the return keyword it contains the expression holding the return value of the function.

You can omit the return keyword and the expression that follows if you wish to write a function without type and return value.
Let us examine the following program list that contains highly expressive functions which can be useful in our future work.

Observe that you do not have to give the type of a function's parameters

Passing parameters in case of functions is very important and its omission can lead to very serious problems, since in the function body only those variables are visible that had been registered as global.

```
<?

  function hello()
  {
    echo "Hello";
  }

  func t i on hi ($BODY)
  {
    echo "Hi".$BODY."!";
  }

  function amount($A, $B)
  {
    return $A + $B;
  }

  func t i on sum( $LIST)
  {
    $SUM = 0 ;

    for ($i = 0; $i < $db; $i++)
    {
      $SUM += $LIST[$i];
```

```
    }

    return $SUM;
  }

  function trim($TEXT, $LIST)
  {
    return str_replace($LIST, " " ,$TEXT);
  }

  function createID($NAME, $BADList,
                           $GOODList)
  {
    $NAME = str_replace($BADList
                        $GOODList,
                        $NAME) ;
    return rand().$NAME;
  }

?>
```

Program 3.21. Using Functions

The first function is very simple and its usefulness can also be debated. (The same could be said about the second one as well.). It has no parameter and if you call, it displays Hello, namely it greets us.

The second function is somewhat more learned. Its input is a name (or an arbitrary text). When you call it puts the word Hi in front of the input parameter and adds an exclamation mark at the end.

The amount function adds the two numbers it gets as parameters and returns the value. Sum gets a list as its parameter and returns the sum of list elements.

Note 3.2. In case of textual data both functions return the concatenation of the passages. . .

Trim is a little bit more useful, since it erases, namely exchanges, all the appearances of the word fragment, it gets as the second parameter, from the text it gets as the first parameter.

The procedures in the function are accounted as routine tasks in writing web pages if we process data from input fields or forms.

CreatId works similar to trim but its parameter is an identifier or name. It exchanges unwanted elements from ($BADList) with their pairs from the list containing the good elements ($GOODList), then it attaches a fairly large integer number to the beginning of the result, thus ensuring its uniqueness.

This function is practically essential in case of generating unique identifiers or filenames but it is very useful in handling images or text files.

Note 3.3. Digital cameras use a generated name for storing images. The naming convention usually happens with a very simple algorithm. The name of the image begins with the word Image, then it is followed by an ordinal number in the order the images are taken.

The ordinal number is often forerun by a space and is between () brackets.

Finely comes the extension which is capitalized by the user's computer or the camera itself.

This is a real nightmare for a web server running on a Unix-Linux operating system. The space, the brackets and the capital extensions (.JPG, .PNG) in the name make copying of images and referring to images on a server impossible…

As you are approaching the end of the list, you can see that writing functions in the PHP language is a strainless and useful activity.

Besides all these, creating classes, code recycling, writing library modules and thus creating effective programs can hardly be imagined without using subroutines.

3.6. Review questions

1. How do you declare variables in a PHP program?
2. What kind of arrays does PHP language have?
3. How can you process arrays effectively in PHP programs?
4. How can you generate HTML code from a PHP source text?
5. How do you define functions in PHP programs?
6. How do you access global variables in PHP functions?
7. What kind of complex data structures can be defined in PHP?
8. How do you refer to scalar variables in PHP?
9. How does the foreach command work in PHP?
10. What kind of control structures does PHP include?
11. What kind of loop commands are there in PHP?
12. How do you mark variables in PHP programs?
13. What options does PHP offer to display and write data on screen?
14. What does the concept of associative array mean?
15. What kind of types can we define in PHP?
16. How do you refer to elements of an associative array?

4. Accessing Databases in PHP Programs

4.1. DBMS systems

Most of the data, menu items and all dynamic content that occur on the pages of dynamic web engines are stored in databases.

This is a very useful attribute, as with the help of database management systems it is much easier to input, display data and to handle data security (DBMS -
Database Management System)

Besides common tasks database management systems also solve problems which must be taken into consideration in case of data handling applications working on a network.

- Data can be accessed simultaneously by multiple users, so you must grant parallel access.
- In case of data input you must grant exclusive access to avoid various anomalies. [4]
- Data storage does not necessarily happen on one computer; that is why you must grant distributed data access.
- You must handle indexes of database tables
- You must control access levels
- You must grant the potential of writing stored processes [3] and also grant access through various network devices

Accordingly, there are several tasks from which a good database manager can spare the programmer.

There are multiple database managers. There are better and worse among them. Choosing the appropriate one can be the key to create a good web program. If speed is the objective and you store little information in data storage, then it is sufficient to choose a smaller and less effective system like MySQL. On the other hand, if you store or move huge amount of data, then you should choose more powerful systems like PSQL or MSSQL which know more.

Never make the mistake of trying to do the data handling system yourself, unless it is your task.

4.2. Accessing Database Managers

No matter which database manager you choose, your first task will be to examine the database management routines of the language you use for programming and the library modules which are responsible for controlling interactions with the database manager.

Remember how web pages operate. Client computers connect to port number 80 of the web server. The server, based on the incoming query, namely the URL, looks up the library and the indexed file within the library that belongs to the query.

If the file contains PHP source text, then the system interprets it with the help of the PHP's interpreter and executes the commands in the program.

Database management can be implemented at this point with the help of the database management routines stored in programs.
In practice this means that we embed the functions, supported by the language, with proper parameters into the source text.

For the sake of simplicity we will use the MySQL system because it is an open source, free, easy to configure and widely spread application. [5]

4.3. The First Connection

To establish a database connection you must possess your own database management system or you must get access to a DBMS version operated by someone else.

It does not matter if you have personally created the web server and use its database manager, demonstrated in the first section, or you choose to subscribe to access; in both cases you must have a username and a password. You need these data during writing the program.

Most service providers keep a sharp eye on protecting their own and users' data; that is why they restrict access to the server as strictly as they can.

Due to all this, it can happen that you can not access the particular server through SSH or Telnet, only through FTP.

Unfortunately, in that case it is impossible to use the MySQL system directly, namely from command prompt, to create tables.

The solution to the problem is to create tables with the help of programs but before doing that, plan the table in which you are going to store your data.
For practice let us prepare a simple application, which stores user data for a primitive MiniUserBook application.

The table in the picture is suitable for data storage:

Picture 4.1. The Table of MiniUserBook

Let us prepare the SQL query that creates our table. The following list contains the SQL command that later we send to the DBMS with our PHP program.

```
CREATE TABLE users
(
  id int primary key auto_increment,
  name varchar(255),
  picture varchar(255),
  cv text
)
```

Program 4.1. Table of Users

It shall do as a start. If you need additional data, you can supplement the table with the alter table command. Now, you had better examine the fields that constitute the SQL command and what they mean.

- The id field is the primary key of the table. You need this in case of all the tables, as this field identifies records. The type of the field is int, namely a signed integer number.

- The auto_increment modification means that we entrust the maintenance of this field to the database manager. The system grants the specification of the proper subsequent value after deletes and modifications. Using this

modification you do not have to deal with the field even in case of inserts or deletes.

- The second field is supposed to store the names of individuals in the record. Its type is a 255 characters long varchar, which is the equivalent of the STRING type in MySQL.

- The picture field will handle the images which belong to people.

- The images are not stored in the database but on the hard drive next to the files containing your programs. You only insert the names of the images and their access path into the table; that is why the varchar type is sufficient.

- You can input people's CVs into the last field. For this purpose a 255 characters long variable is not enough; that is why we have chosen the text type.

In order to really create the table you must access the database manager and send it the query, you have created, for processing.

The following program list contains the source code that completes this task.

You must write the source into a file, then put it in the root directory of your web page. To run it in a browser you must refer to it in the way you have learnt.

```
<?

  $USERNAME = 'username';
  $PASSWD = 'password';
  $DBNAME = 'webbok';

  $conID = mysql_connect('localhost',
                         $USERNAME,
                         $PASSWD);
```

```
  if (!isset($conID))
  {
    echo "Connection failure
                    Error message : "
                    .mysql_error ( ) ;
  }

  mysql_select_db($DBNAME);

  $querySTR = "
     CREATE TABLE users(
       id int primary key auto_increment,
       name varchar(255),
       picture varchar(255),
       cv text)";

  mysql_query($querySTR)
      or die("Failed to create...<BR>".
              $querySTR );

  mysql_close($conID);

  echo "Table created successfully<br>";
  echo $querySTR ;

?>
```

Program 4.2.Creating a Table from a PHP Program

By no means shall you refresh the program in the browser because after creating the table it cannot be created again and you would end up with just another error message. If the program is bad, then it will not work due to refreshing.

After running the program, the message" Table created successfully." should be displayed in the browser. In case of an error, a message coming from the database manager or the one displayed by us should appear.

If you see ” Connection failure. ”, you should look for the mistake somewhere around the mysql_connect function. ”Failed to create…” appears when you make a mistake in the query.
When you finish looking for the error, examine the source text (in this program we have not paid attention to proper character coding or to the quality of the HTML output, so do not be surprised if, instead of accented letters, strange characters appear on screen.)

The first line of the code contains the call of the mysql_connect function. This function has different parameterizations.

In our program the first parameter of the function is the word localhost, since the database manager and the web server are on the same machine.

Localhost means that you have to connect to the local loop, namely to the 127.0.0.1 IP address [1]. This parameter can be supplemented with the number of the virtual port used for accessing the database.

The second and third parameters contain the username and password needed to access the databases. These data are worth the trouble of being protected and stored in the same place. On the one hand to avoid hacking of our system, on the other hand to avoid being forced to rewrite them at several places in the program if they change.

You should store the return value of the function in a variable $conID in order to be able to verify successful connection in the next conditional branch. In case of failure we display an error message in the browser window.

The parameter of the mysql_select_db function is the variable containing the name of the database. Besides the name and the password you must possess this information as well.

The identifier of the database is either provided by the service provider or created by you on your own system. This database stores your tables just like the users tables that are currently used.

You place the command, specifying the creation of the table, in the $querySTR variable.

You do not necessarily have to store the SQL commands in a variable, but is worth it, as in case of an error you can have them displayed on the screen if you want to correct and check mistakes.

The mysql_query function sends the SQL command to the DBMS. If the function fails due to any reason, the exception handler, directly written after it, takes control and displays the possible cause of error.

The task of the mysql_close function is to terminate connection It must be done without exception because, that is true that open connections end sooner or later, but the open connection makes easily attackable web applications even more vulnerable.

When you write applications working with databases, always pay attention to close connections as soon as possible.

Conversely, do not expect the connection to be open in the next minute. Think over how programs are used by users acting on the web. Sometimes they click, take an older address, then they refresh the browser window. They can possibly wait for long before clicking on something again.

To suppose that the connection is open in the time between the two clicks is nonsense. If you write the program badly, it can easily happen that the connection is terminated automatically before the user clicks on your following query. In the best case your program throws in an error message on the screen.

Our program is much cleverer than that. When it finishes work, it terminates connection and informs us about the success.

4.4. Private Functions for Connecting

Communication with the database consists of the following easily separable steps:

- Connecting to the system

- Choosing database
- Creating query
- Sending query to server
- Processing result
- Terminating connection
- Handling errors in case they occur

These routines comprise of the same few steps, or sequence of commands, which may only alter in their parameters (username, password, or the SQL command you send to the server).

Obviously you should group the various routines and write functions, whose input parameters describe the changing information.

4.5. Creating Database Management Module

It is worth to place the database management functions into a separate module, since from then on you can place the module next to any web pages' program that you create and you are free to call its functions.

This is the concept of code recycling, which you can improve further by organizing functions into classes and using an instance of the class in your programs.

Do not forget that PHP is an object oriented language. This solution is very useful if you later replace the database management system that belongs to the web page.

Database management functions are at the same location, thus they are easier to find.

Low-level database managing procedures can be replaced in a way, that you do not change the interface parts of your functions only the commands in the function block.

From this point on, if a particular function does not connect to a MySQL system, the other parts of the program will not even notice.

Preparing your programs to be platform independent is a basic programming technology. It is implemented with middlewares also known as layers.

We will follow the principles of simplicity and traceability and choose the solution that uses functions and leave the creation of classes for later on.

The only question left is what groups can you enumerate the steps of the database manager into.

You should carry out the grouping based on the various functions. You need one group to establish connection, one to send queries and one, in order to be platform independent, to terminate connection. (This will be a very short but very useful function)

Let us prepare our database management module and place it into the proper library next to the index.php file.
The formerly written routine, creating a table, should not be integrated into this module because all of its functions will be replaced by the functions of the module.

```
<?

#====DBmodule=============================
#
# Author : Author
# Date : 2010.01.20
# TODO : functions for PSQL system
# Change log:
#  2010.01.20:Comments of Mysql functions
#
#==========================================

  #TYPE::connect():int
  function connect()
  {
    $ID =
```

```
      mysql_connect($HOST, $USER, $PWD);

    if (!isset($ID))
    {
      return "error:".mysql_error();
    }
    return $ID;
  }


  #TYPE::select_db(DBNAME:string):void
  function select_db($DBNAME)
  {
    mysql_select_db($DBNAME);
  }

  #TYPE::query1(querySTR:string):void
  function query1($querySTR)
  {
    mysql_query($querySTR)
        or die("Error ".mysql_error());
  }

  #TYPE::query2(querySTR:string):array
  function query2($querySTR)
  {
    return = mysql_query($querystr)
         or die("Error ".mysql_error());
  }

  #TYPE::close(ID:int):void
  functionclose( $ID)
  {
    mysql_close($ID);
  }

?>
```

Program 4.4. Database management module

Let us consider the elements our library module contains. Right at the beginning there is all the information you need regarding the module.

This part is a comment and does not contain anything that might influence running of the program but it is still very important. Developers can get all the important information regarding the source code's life cycle from here.

The parts before functions, beginning with TYPE, inform the programmer about functions and the type of their parameters. Running of the program is not affected by this part either (it can be omitted), but it is suitable for generating documentation and writing down all the important attributes of a function making further development of the program easier.

For this purpose you can either use existing documentation tools or make up your own. This latter one is much more useful as you can learn a lot with writing the analyzer program that belongs to it.

The connect, select_db and the close functions provide the functions reviewed earlier.

The task of the two query functions is different though, that is way there are two of them.

The query1 executes insert, modify and delete type queries, while the query2 executes ones which provide data for your program for further procession.

Note 4.1. Notice that changing the blocks of functions, or rewriting them with the commands of another database manager do not change the way functions can be applied at all.

Since PHP does not generate object code, it is enough to copy the accomplished module next to the other files of the application, then to make it visible at all the places where you want to use it.

In practice this means that in the programs in which you want to use the module's functions you must place an include or require function parameterized with the name of the module.

```
<?

  #at the place of use
  require "dbmodule.php";

  #or
  requi re_once "dbmodule.php";

  #thus linking is not duplicated,
  #in case you are careless

?>
```

Program 4.5. Linking modules

To create platform independent database manager the above described two query functions are not sufficient. Unfortunately, at the place of call you need mysql specific functions to process data coming from the DBMS.

```
<?

  ...
  while (list($v1, $v2, ..., $vn) =
      mysql_fetch_row($result))
  {
   ...
  }
  ...

?>
```

Program 4.6. Processing data

The problem lies at the mysql_fetch_row function as you must call it at all the regions of the main program where you would like to process data.

One solution is to change the query2 function in a way that it will carry out the functions of fetch_row, namely it will decompose the recordlist into lines and put the result into an associative array.

```
  #TYPE::query2(querySTR::string)::array
  function query2($querySTR)
  {
    $lista = array();
    $result = mysql_query($querySTR)
          or die("Error ".mysql_error());

    while($line =
        mysql_fetch_assoc($result))
    {
       $lista[] = $line;
    }

    return $lista;
}
```

Program 4.7. Modified query

The first line of the modified query2 function defines an array by calling the array() function. It places the result of the SQL query line by line into the $line array in a while loop.

The obtained array can be processed by the main program in the following way: By embedding two foreach loops all the useful information can be extracted from the result list. The outer loop provides the lines, the inner one selects the field names and the field content from the particular line with the help of pattern matching.

```
  ...
  $lista =
      query2('select * from tablename ');

  foreach ($lista as $line)
```

```
{
    foreach ($line as $k => $v)
   {
     echo "$v is the key,
           $k is the data<br>";
   }
}
...
```

Program 4.8. Data processing with pattern matching

This method of data processing is universal enough so its database manager module can be used in all regions of the program or even in other programs as well. There is only one part of the source text that has not been discussed yet, the pattern matching of the inner foreach.

```
foreach ($line as $k => $v )...
```

Program 4.9. Linking modules

The content of the $line variable, which is by the way the element of an associative array, is a key => value form data. Its first segment is the key of the particular array element (its name), the second is a data that belongs to the key (value).

In the course of pattern matching the key gets into the $k variable, the value that belongs to the key gets into the $v variable; that is why you must use the => selector.

4.2. Note. If you still do not understand the code, read the section on handling arrays carefully.. . .

With this slight modification you have modified the code to be universal, as you paid attention to be able to use the functions in as many programs as possible regardless of parameterization.

You have made database management transparent by creating the platform independent interface, so if you replace the database management system you do not have to modify the main program.

Any modification, which concerns the database management, can be carried out by modifying the module's functions or by keeping the interface and replacing the whole module.

The invested effort and the plenty of source code might seem to be needless but it is definitely worth it on the long run.

4.6. Review questions

1. What are the steps of establishing database connection?
2. How can you check if the database connection has been successfully established?
3. How can you create SQL tables in PHP programs?
4. How do you process the multiple record result of an SQL query?
5. What are the advantages of grouping database functions into modules?
6. How can you create SQL tables?
7. How can you give commands to the DBSM?
8. What tools does PHP supply to manage the database with?
9. How do you terminate database connections?
10. How can you avoid multiple file linking in case of using multiple include or require functions?

animaciok/TAMOPweb0003.avi
animaciok/TAMOPweb0005.avi

5. Module Structure

5.1. Modular Architecture

After creating the table, necessary for storing data, and having written the library module containing the database management routines, you should not forget that you are creating an application that runs in a browser and can be accessed by anyone through the

net. You will probably need to gradually develop the program further due to emerging demands.

If you redesign your program keeping all that in mind, you will soon realize that you need a system shell in which you can describe menu handling, menu items and the files storing their content.

Picture 5.1. Modular architecture

It is worth it to group the source texts of the various pages. Remember the CMS modules or their menu structure you got acquainted with in the first section.

So, before beginning writing the database manager, create the system shell which will be stored in the index.php file.

In this file you will write the HTML code of the page, namely its appearance. The program parts that manage menus, the code for building menu items and the loading of pages appearing due to this code are also linked here.

If you succeed then later you will only use the index.php file to integrate new menus or programs of new functions. Thus the structure of your page will somewhat resemble the structure of CMS

This happens consequently because the programmers of CMS also build up the structure of their programs logically. In picture 5.1 you can see the sketch of modular architecture. The picture helps you to better see through and understand module structure.

```
<html>
 <head>
  <title>testprogram</title>
  <meta http-equiv=" content-type "
       content="text/HTML;charset=UTF-8">
  <meta http-equiv="content-type"
       content="application/xhtml+xml;
                          charset=UTF-8">
  <meta http-equiv="content-style-type"
```

```
                        content="text/css">
 </head>
 <body>

   <? require "dbmodule.php"; ?>

 <table class=header>
   <tr>
     <td>

      <? include "header.php"; ?>

    </td>
   </tr>
   <tr>
     <td class=menu>

         <? include "menu.php"; ?>

     </td>
     <td class=content>

        <? include "content.php"; ?>

     </td>
   </tr>
  </table>

 </body>
</html>
```

Program 5.1. Module Structure

List 5.1, showing the program of the module structure, besides the HTML code includes the reference to the style sheet and the PHP blocks. The blocks are compulsorily placed between the <? ?> tags in order for the web server to recognize the commands written in the source texts.

5.1. Note Files with css extension describing the style sheet are to be placed next to the index or else you must provide access…

The parameter of the include functions placed at some regions of the source code is the name of the file that is supposed to be loaded at the particular phase (or its path and name).
Now let us prepare the HTML code of the header, with which you load a picture and place a short text that functions as the address of the web page. This program must be saved into a header.php file.

```
<table class=header width=760>
 <tr>
   <td>
     <img src=images/header.jpg>
   </td>
   <td>
     Header of the web page
   </td>
 </tr>
<table>
```

Program 5.2. Structure of the header

The program works as if the source code, written in the file it gets as a parameter during running, was copied to its place and run during execution.

The include function holds a lot of potential for the programmer regarding dynamic code generating and code conversion. This potential is to be utilized by you to display only those contents belonging to the menu items in the browser which are attached to the specific menu.

The code snippet containing menu items can either be simple HTML texts or menu items can be generated from lists as well.

5.2.Note. Create the files that belong to the branch but are still missing, then copy them to the directory of the web page, if you do not want to receive an error message...

To operate the web page similar to CMS you can store menus in a database. In the next part we show examples of both, static and dynamic, solutions.

5.2. Menu Management

Creating menu items. To store menu items and the links that belong to them you must create a proper table. The table contains the fields you can see in picture 5.2 and can be created in the way you have learnt earlier.

```
CREATE TABLE menutable
(
 id int primary key auto_increment,
 name varchar(255),
 link varchar(255)
)
```

Program 5.3. Storing menu items

5.3. Note. Notice, that the field for storing the links is pretty short; only 255 characters long.

Fortunately, this is not a problem, as you do not store the long URLs used on the net, only the name of the index.php file and the content of a couple of variables (see later). In case of longer links, use fields with bigger capacity…

Storing menu items in a table make it possible for you to later provide an administrative interface for creating menus where your users can add, delete and modify items of the menu.
Program list 5.4 contains the source code of the menu manager. The module uses functions, previously written in the database management module, to access data.

```
<?

  $conID = connect();
  select_db("webbok");

  $q="select name,link from menutable";

  $res = query2($q);

  close($conID);

  echo "<table class=menutable>";
  while (list($menutitle,$link)
              = mysql_fetch_row($res))
  {
    echo "<tr><td>
           <aclass=menu
             href=index.php?d=".$link.">"
             .$menutitle."</a>
          </td></tr>";
  }

  echo "</table>";

?>
```

Program 5.4. Storing menu items

The first few lines of the source text contain the source code necessary for building up database connection; the further regions contain query of menu items.

The loop building up the HTML source for the browser, from names and links of the menu, is a new feature considering the antecedents.

The while loop runs until the query returns records, so if the output is 4 records it runs four times.

Every time the loop runs, the values of the fields form the record returned by the mysql_fetch_row function are loaded into the variable in the list function. The field of the first record is loaded into the first variable, the second field into the second variable and so on.

In your case the value of the name field is loaded into the $menutitle variable, the value of the link field is loaded into the $link variable.

The link field only contains the “number” of menu items and not the name of the file specifying the content of the menu item (URL).

Unfortunately, this does not allow handling of external references. The order of fields and variables and their number is important as well. If you want to load the values of four fields into three variables, then you lose the last field.

However, the name of variables is not important, but the program is more comprehensible and more legible if you name them based on their functions. The name should be similar to the name of the field, where possible, but it should indicate the function.

The block of the loop in the source code creates the menu items based on the data it gets, puts their names between the <a…> tags and it places the link referring to the page to the href= part

5.3. Routines of Menu Management

```
<?

  if (!isset($_GET["d"]))
  {
    $_GET["d"] = 0;
  }

  switch ($_GET["d"]))
  {
   case 1: include "menu1.php"; break;
```

```
   case 2: include "menu2.php"; break;
   case 3: include "menu3.php"; break;
   case 4: include "menu4.php"; break;
   default: include "startpage.php";
                                  break;
  }

?>
```

Program 5.5. Menu Management

Downloading content of menu items. The first conditional branch of the menu management code, which is either in the index file or the content management module, has a very important role. In case of its omission the program stops at its first run because in the PHP language the variable is created when it is first assigned.

If the user does not click on any of the menu items, which is highly probable in case of the first download of the page, your d variable will not be created, but you refer to it in the next line.

It's a serious mistake, so you must write the conditional branch to the beginning of the source code to evaluate this possibility.

The isset function tells about the variable it gets as its parameter if it exists or not. Since you negate with the ! operator in the conditional branch, the condition is fulfilled if there is no d variable.

In this case d gets 0 as its value, which means the default path for your program, namely the download of the start page.

In all other cases when d has already got a value, it should not be touched, as it gets its value when you click on the menu items.

Your page works in a way that the value of d is 0 by default when first downloaded, namely when the client computer refers to it. In this case the default path of the switch is run and the start page is downloaded.

If the user of the page clicks on one of the menu items, it changes the value of d, and the page defined in the path with the particular value will be downloaded.

You refer to the d variable with the $_GET["d"] expression because the use of global variables on Apache web servers has been restricted with setting the REGISTER_GLOBALS directive OFF.

This is not true on all servers. Naturally, the operator of the system can change the value of the directive. Fortunately your program works well in all cases as it stores values in the session that belongs to the page.

Note 5.4. This solution is used in case of Form and file management as well, only with different names, regarding arrays used for data transfer. ($_POST $_FILE). . .

There are other ways of menu management as well but you should never do the following. Do not transfer file names in URL:

```
http://...index.php?d=file.php
```

If you do so, then by renaming the file the content of any other files which can be accessed on the server can be displayed on the screen of the client browser.

Todo so, you only need to change the URL:

```
http://...index.php?d=/etc/passwd
```

The result will be catastrophic because the menu management code snippet reads the passwd file with the help of the include function and displays it in the browser window.

With the development of web servers this attack surface is becoming harder to exploit but still it is a vulnerability to pay attention to.

If you do not want dynamic menu management, you must simply write the source code of menus into the HTML code. This way adding new menu items can only be done with editing the source code.

Program list 5.6 contains the description of static menus:

```
<table class=menutable width=150>
  <tr>
   <td>
     <aclass=menu
       href=index.php?d=0>startpage</a>
  </td>
 </tr>
 <tr>
  <td>
   <aclass=menu
      href=index.php?d=1>Menu1</a>
  </td>
 </tr>
 <tr>
  <td>
   <aclass=menu
      href=index.php?d=2>Menu2</a>
  </td>
 </tr>
 ...
</table>
```

Program 5.6. Creating Static Menu

Of course the code that runs when you choose a menu item does not change as it only checks the value of the d variable that is created by menu items; otherwise it is totally independent.

5.4. Review Questions

1. How do you link modules and files to PHP programs?
2. What web server settings lead to the use of the $_GET, $_SET and the
3. $_FILES arrays?
4. How do you manage global data in case of dynamic web pages?
5. With which protocol can you access the library of the web server?
6. What are the consequences of setting the REGISTER_GLOBALS parameter OFF?
7. What do various parts of the URL identify?
8. Is it possible to transfer data between the server and the client?
9. How do you access modules containing pre-composed PHP functions?
10. Why is it worth placing functions of various web pages to separate files?
11. How do you inform the web server that a PHP script is coming up in the HTML code?
12. Can you access associative array elements with indexes?

6. Creating Data Management Applications

6.1. Data Input

Try to make data input as simple as possible just like we did it with breaking up web pages into modules.

Web interface does not make direct data input possible. As you can not use simple data input commands, you need to think your options over.

How do dynamic web pages work? The web browser sends its queries to the number 80 port of the server. The web server processes the URL in the query and looks up the referred page in the root directory, then running the PHP source text in the file it assembles the output it sends to the browser.

Picture 6.1. Data Flow on the Web

As you can see, it is much more complicated to acquire data from the users than in case of programs with user interfaces running under operating systems.

To manage the input in case of web applications it is evident to use forms

Form is a unit used in the HTML language, which can contain controls.

A control, which has many types, is a unit helping user interaction. List elements are often supplemented with special controls written in client side script language

Note 6.1. Most of the time we use Java based controls...

You can write forms embedded in HTML code or in PHP programs in the way illustrated in program list 6.1.

Note 6.2.: The use of forms is only detailed to an extent necessary to solve the task.

We try to focus solely on the attributes which are essential in data management.. . .

```
 <form action=use.php method=post
           enctype=mult ipart/form-data>

  <input type=t ext name=text>

  <input type=submit name=btn1 value="OK">

 </form>

<?

  echo "<form attributes>
             form  buildelement </form>";

?>
```

Program 6.1. Creating Forms

Examine the elements used for creating the form in program 6.1:

- The action=use.php part means that you process the form units with the program in the use.php file.

- The method=post refers to the form being POST type, so it forwards data to the program region describing processing routines

- The multipart/form-data attribute is necessary to later put the data meant to be uploaded into the $_FILES variable; data that describe files, data that are necessary for file processing,

- The controls, which can be used in forms, give enough support for you to be able to transfer any data from the client side to the server and back, so it is worth it to examine the more important ones. the text field can be used to read short text data.

```
<input type=text name=text...>
```

the textarea is an even HTML language element, which creates a textbox capable of inputting texts of several lines

- You can write text between the <textarea> and the </textarea> tags and it appears on the interface of the control. With the rows and the cols attributes you can give the column and row number of the textbox, while the name attribute defines the name of the variable allocated to the control.

- With the help of checkbox, you can create several options for the users. You can group checkboxes and within a group several or even all of the items can be active when you submit the form.

Note 6.3. Of course it is possible for the user not to check any items from the group. You should keep that it mind and make sure that the variable has a default value to avoid problems caused by empty or non-existing variables while processing data.

As you already know, in PHP, variables are created when they are assigned a value and this can cause serious problems if none of the items in the checkbox are checked and the program region responsible for processing refers to the name of the group. . .

- the item named radio creates a traditional radio button on the form. These items can also be grouped just like checkboxes.

The names of items must be the same, however their value must differ. Only one item can be checked within the group and thus you can bring the user to answer yes or no questions.

```
<input type=radio name=r1 value=1>
```

There is a control capable of browsing files, which can be created using the following:

```
<input type=file name=file1>
```

As you can see, the type of the control is file, its name is file1. Do not use a name that is the same as the type because that can lead to various problems while processing variables. The button of this item starts a file browser wizard in which you can choose the file you want to upload. If you do not want to browse, you can type the path and the file name.

When browsing files, you can only choose one for processing. It is futile to look for a browser capable of browsing folders. Managing multiple files and folders would be a fundamental attack surface, since in this case server side scripts would have to allow access to

the local folders of users (in Linux this would mean the permission of X access).

Note 6.4. If you want to upload more than one file, you must select them one by one. We will return to file management, and show you upload techniques and security issues later. Now content yourself with the first step of being able to manage files. . .

This much is enough of the items on forms. With their help you can read any data that you need, however this much knowledge is not enough for processing.

You must process and store the information you get with the form and react to upcoming errors.

The following part of this section deals with these problems. Place the input program regions as modules on the web page you have already created, and with the help of the data management module store the information coming from the user.

6.2. Data Processing

From the form you need to transfer data to the program region that had been prepared for data processing. This is not a difficult task but you must understand its functions to be able to manage incoming information effectively.

If we examine how data flow happens, we need to mention the unique statelessness of the WEB. In contrast with the programs running on the clients, web pages exist in a unique so called “stateless” state.

In reality this seemingly contradictory situation ensures security on web interfaces, data flow among web pages and data flow among web pages and the web server.

If you type the address of a page into the browser then it downloads the content from the server included in the address.

The program regions in the web page's source create variables and assign values to them.

The set of variables and data, gained this way, is called the state space of the program.

The user of the web page clicks on the page or simply refreshes his browser and the majority of the variables used by the program are gone or their content is replaced; as by refreshing, the program runs again and executes all commands again.

Unfortunately by doing so it overwrites the variables that have formerly been assigned. Data transfer among the "currently" running web pages is only possible through special means. To store global values you can use session variables or external storage, a file or perhaps a database.

Note 6.5. Session management will be discussed later because it is essential to use it in logins and tracking. . .

In case of forms, data transfer is not that uncomfortable but you have to be cautious if you want to do everything nice and correct.

You can get data from the form by two means depending on the settings of the web server.

The wrong solution is to directly refer to variables, defined in the form, in the program ($textbox1, $file1).
As we have mentioned earlier this solution works on some servers, where REGISTER_GLOBALS directive is set default but there are less and less servers like that and you cannot be sure that the administrator of the server does not change settings.

The right solution is to use the $_FILES arrays to process data, in case of $_GET, $_POST and in case of files. The system places data that it gets from forms in them. In case of post type forms it places data into the $_POST associative array; in case of get type it places data into $_GET. The data of the file coming from the file browser is stored in the $_FILES array.

Each array element contains the value of the variable that has the same name as the name of the particular control, which can be referred to with the label that has the same name as the particular control $_POST['name of the control'].

To tell you the truth these arrays are the global variables of your web page with which you can store and transfer data while shifting to various pages and sub-pages. For all the web pages and to one instance of all the pages, a session is created in which you can freely store anything.

Note 6.6. It is important to protect data in sessions, too. If possible, protect it from harmful users. Try to do your best in you programs regarding data security. . .

Now, that you know the theory of data transfer, let us have a look at how information gets from the form to the processing module.

Remember the formerly created form that refers to a file. The task of this file is to process the data of the form.

Actually this means that the users activates the form by clicking on the submit button and his browser requests the web server to interpret the source code in the file.

This file does not display anything on the screen, it only executes the tasks specified for it. After finishing work, it gives control back to another program region that informs the user about what has happened.

Picture 6.2 illustrates this. As you can see in the picture, this works exactly the same as the main program when it calls a function, which executes its tasks, then returns the result to the main program.

Note 6.7.. In your case the function is a program region specified in a file, or if you like it better this way, it is a subprogram with its own tasks and state space. . .

After seeing how the program is called and how it returns, prepare the module that is used for data input. Let us continue the module that you have started in the third section.

To input data of people you need a form and a program which can store and display the data input

Picture 6.2. Calling Functions and Modules

Later on you can create queries and different filters to structure data

Note 6.8. Data management routines and the table for storing data are at hand, as you have already created them.

```
CREATE TABLE users
(
  id int primary key auto_increment,
  name varchar(255),
  picture varchar(255),
  cv text
)
```

Program 6.2. Table of Users
Program text 6.3 contains the source code of the form:

```
<form action=insert.php method=post
            enctype=multipart/form-data>

 <input type=text name=nev><BR>

 <input type=file name=kep><BR>

 <textarea rows=10 cols=50 name=cv>
 </textarea><BR>

 <input type=submit value="Add">
```

```
</form>
```

Program 6.3. Data Input Form

Note 6.9. It is worth formatting the form that helps the input with the use of a css file or at least placing it in a table to make it more transparent and easier to handle.. . .

Now, that the source code is ready, save it to the useradd.php file, then place a reference to the file into the menu management module.

```
...
switch ($_GET["d"])
{
  ...
  case 1: include "useradd.php"; break;
 case ...
}
...
```

Program 6.4. Menu of the New Function

Place the new menu item among the older ones. With dynamic menus this operation involves inputting the proper record in the table of menu items. If you have created a static menu, simply type the following link into the code.

```
<ahref=index.php?d=1>MiniUserBook</a>
```

Program 6.5. Placing New Menu Items

You can try the effect of your work. If you have done everything right, then by clicking on the menu item you should see the form shown in picture 6.3.

Of course input does not work yet, as you have not created the program that does the insert.

6.3. Sending and Inserting Data in Databases

If everything has been appropriate, let us go on. Create the program responsible for data storage.

```
<?

  require_once "dbmodule.php";

  $c = connect();

  select_db("webbook");

  $querystring =
   "insert into users (name, picture, cv)
      values ('".$_POST['nev']
                ."','"
                .$_FILES['pic']
                ."','"
                .$_POST['cv']
                ."')";

   query1($querystring);

   close($c);

   echo '<meta http-equiv="refresh"
           content="0;URL=index.phpd=1">';

?>
```

Program 6.6. Inserting Form Data into a Database

The source seems complicated for the first blink but if you analyze it further then you will realize that every single character has its role.

First of all, the require_once function loads the formerly created database management module.

Note 6.10. If it has been already loaded by another program region, the program will still run properly as the require_once function does not allow multiple load initiatives. . .

The connect, and the selectdb functions connect to the database manager in the formerly discussed way and they choose the database which the program requires.

The next few lines, where you create the variable called $querystring, are the most important and at the same time the most complicated parts of the program.

This string is sent to the database manager, which executes the insertion of the record assembled by us.
The record comprises of the data from the form. One after the other, the name, given, which comes from the $_POST['name'] formula. The $_POST array is an associative array created for the post type form, whose element, which is referred to by the label ‘name’, contains the $name variable of the form

Picture 6.3. Output of the Mini Application

Note 6.11. Actually, this array element only gets the content of the variable and not the variable itself, but it is easier to consider the variable to be in the array. . .

The same stands for the $_POST['cv'] element, however, here, instead of a name, a longer text is stored by the system. This data is also appended to the record to be inserted, namely to the sql query command that we create in the $querystring variable.

The $pic variable placed in the $_FILES array is somewhat more complicated than the previous ones, as all of its elements contain the data of a file chosen by the user for upload.

Its every element is a complex data structure, which besides containing the name, the path, the type and the size of the file, contains all the information that is necessary for processing the file

You must copy the file, with the help of its name and path, onto the web server, into a folder created for this purpose. You must create its new name, then by “sticking together” the new path and filename interlace it to the SQL command.

Note 6.12. Renaming the file is a very important step if you do not want to run into errors while uploading. On the one hand, this must be done because the names of uploaded pictures can contain characters which cannot be handled by the web server as paths. Such character is the space, which means the end of the path in an operating system, the rest is not taken into consideration.

On the other hand, you must ensure that the names of uploaded files are not identical. If they were, then either uploading would be unsuccessful or the previously uploaded files would be overwritten.

This problem will be solved later with a short function that removes unwanted characters from the name and attaches a relatively large random number to its front. This way the uniqueness of filenames can almost be guaranteed. . .

Uploading the file will be tackled in another section. This time rest satisfied with inserting the name and the path into the database and while displaying data only displaying the name of the file on the web page.

Returning to the source code you can see that the query1 function has also been called from the database management module. This function is responsible for sending data.

The SQL command, sent to the database, deserves some attention. Assembling its content is interesting itself. This technology is essential in the programming of dynamic web pages and is probably one of the most interesting things regarding them.

SQL commands, which are sent to the database manager, are assembled by the program in runtime, affected by user intervention.

Practically, queries are created from the variables of form data and the appropriate SQL elements, meaning that these supply the correct result under any circumstances. This dynamism is necessary because it would be impossible to program queries in advance.

The SQL command in source text 6.6 comprise of the following:

- The first section of the query contains the SQL command itself, that is to say it specifies the operation to be executed.

- This operation is an insert. Furthermore it contains the name of the table in which the record has to be inserted

- The next section specifies the fields, before the values key word, between brackets, in which data will be stored. The enumeration is required because the id field, namely the primary key of the table is assigned a value automatically. This field must be omitted from the list, and it is only possible if the fields which get a value during the insert are enumerated, while the others are not.

- The following list in brackets contains the values to be inserted. This part of the query is rather complicated because it is full of single quotes, quotation marks, commas and brackets.

- This unique and complicated specification is necessary as you use two languages in the same code, and this blend is embedded into a third one, HTML.

You need to append the content of the variables defined in PHP to the SQL sections; and in addition these are stored in another session variable. Single quotes and quotation marks come handy as with their help the interpreter and compiler programs can differentiate among the languages you used

To be able to interpret the ('".$_POST['name']."','. expression, examine how the list of elements to be inserted would look like in case of a static query.

```
insert into users (name, picture, cv)
             values('Name of Person',
                    'picture01','CV...')
```

Program 6.7. Static SQL Command

You can see that sql interprets the data to be inserted as strings if you write them between the ‘ ’ characters. The distinct items must be separated with comas.

That is why you need single quotes and comas in dynamic queries as well ','" . $_POST['name'] . "','.

Quotation marks "" in PHP mark the beginning and end of character strings. They fulfill the same task in the beginning and end of sql segments as well.

Note 6.13. Notice that queries are simple texts which only gain a meaning when they are assembled and sent to the database manager. They only become real commands when they are interpreted by the database manager. . .

Dots (.) between parts separated by the "" characters mean the concatenation of text data. Their role in the sql command is to concatenate the content of the PHP variables to the segments of the SQL command.

Obviously, assembling of the SQL command could have been done by other means as well.

You could think about simplifying the query but the solution you have used works fine in any server context and if you get used to using the characters in it, it will become transparent for you.

The last few lines of the source text are responsible for terminating the database connection and after the insert, for taking you back to the form where the insert was initiated.

The simplest way to carry out this backtrack is by a META command. In the META command you specify how long to wait after insert (in this case it is 0), and which page to load, where to jump back. REFRESH means that the browser must be refreshed and the URL defines the page to be loaded.

We choose the simple solution to write the same link to the URL part of the command that we have written next to the menu item used for input. This is the index.php?d=1 link.

By this you achieve that the browser is refreshed when the user clicks on submit after typing in the input data. Then the user immediately gets back the input from. Seemingly that is all that happens but meanwhile his data is stored in the database and if you create the module carrying out the display, then this new entry will appear for him.

6.4. Review Questions

1. How do you concatenate strings in PHP programs?
2. What is the source of error if the error message beginning with Forbidden appears in the browser?
3. What do you do when you see the error message that begins with Forbidden?
4. How do you read data in PHP programs?
5. How can you carry out transferring of parameters between web pages?
6. How do you look for errors in PHP programs?
7. What client side languages can you use for supplementing programs of web pages?
8. How do you refresh pages in the browser?
9. With which arrays can you access global variables?
10. How are dynamic queries made?

animaciok/TAMOPweb0004.avi

7. Generating Lists

7.1. Displaying Data in Lists

Unfortunately there is only one way to check the result of the input at this phase of the program. By giving the following commands in the command prompt of the DBMS 7.1.

```
mysql> use webbook;
...
mysql> select * from users;
mysql>
```

Program 7.1. The Command Prompt of the DBMS

Note 7.1.To access the command prompt of the database manager you must establish terminal connection with the server. For the connection you can use a client program with a graphical interface or you can initiate it from the command prompt of the operating system. In Linux it can be done with the following command: ssh server -l username. Using SSH protocol you give your username after the –l switch. Then the system asks your password and if that is correct, it allows you to access the server.

You must login in the MySql console, too. For this, use the following command: mysql -u username -p. If you login without trouble, run the sql query above. . .

If the last input data appears among the records of your table, your program works fine and you can start building up the list constructed from the data.

The database management module offers a way for this task as well. The query2 function expects an sql command as its parameter and returns the records in a list. You just have to process and display this list embedded into an HTML code.

You cannot handle the length of the result list as a constant because it depends on the number of records. Obviously, you need to operate with a loop command.

7.1. ábra. The Output of the SQL Query

The loop stops after you run out of records, namely when you have displayed all the data in the client computer's browser.

The question is how you know that you have reached the end of the list. The PHP language contains a mysql_fetch_row function which helps you to stop the loop.

```
<?

  require_once "dbmodule.php";

  $c = connect();

  select_db("webbook");

  $querystring = "select * from users";

  $result = query2($querystring);

  echo "<table width=100% border=1>";
  while (list($id, $name, $pic, $cv)
              = mysql_fetch_row($result))
  {
    echo "<tr>
            <td>$id</td>
            <td>$nev</td>
          </tr>
          <tr>
            <td col span=2>$pic</td>
          </tr>";
  }
  echo "</table>";

  close($c);
```

```
?>
```

Program 7.2. Displaying the Records of the Table in HTML

When displaying the records you use the same technique that was used earlier for displaying menus. The output of the PHP program was displayed in HTML format.

Note 7.2. You could have prepared the program with embedding the PHP program segments into the HTML code, or simply with marking the PHP sections and entrusting the system to load them.

Currently, the most widespread tool for that is Smarty, that can be used to distinguish between the PHP and HTML codes but there are numerous other solutions as well. We leave it to the dear reader to get acquainted with the tools. . .

The beginning of the program might be familiar. In the first few lines you load the database management module, connect to the database manager, assemble the query, generate the HTML output from the result, then you terminate connection.

After saving it (for saving you can use the output.php filename),

You have to integrate this program region into your MiniUserBook application too.

You can do it by placing the following line into the source code of the input form: include "output.php";.

If you place the reference before the region specifying the form, the list appears before it, if you place it after the region specifying the form the list appears after it.

Let us have a look at the result. If you have worked with more than one record, then they appear under each other in the order they had been input. This format is not too practical in case of long lists, as with the increase of the length of the list, your page is becoming less and less clear-cut.

To deal with this problem it is enough to change the displaying a little bit.

The screen will be much more transparent if you place one or two elements in one row based on the size of the pictures.

Note 7.3. Obviously, you do not have pictures yet, only paths but that does not change the essence. . .

To create line-wrapping after every third element you must use modulo division to evaluate when the third element is coming up. If you find it, you must insert a </tr><tr> tag pair, which closes the current line of the table and opens up a new one for the next three data.

To wrap the table properly, you begin a line before the loop, then you create three cells in each line, which contain the data of a record in an embedded sheet.

You need this many cells and sheets to be able to format each and every part of the list separately, or to be able to assign an individual style to them.

```
<?

  ...
  $i = 0;

  echo "<table width=100% border=1><TR>";
  while (list($id, $name, $pic, $cv)
     = mysql_fetch_row($result))
  {
    if ($i % 3 == 0)
    {
      echo "</TR><TR>";
    }

    echo "<TD>
          <table>
           <tr>
             <td>$id</td>
```

```
            <td>$nev</td>
          </tr>
          <tr>
           <td col span=2>$kep</td>
          </tr>
         </table>
        </TD>";
    $i++;
  }

  echo "</TR></table>";
  ...

?>
```

Program 7.3. Wrapping the Table

Note 7.4. In the modified program we have written the new HTML elements with capital letters to be able to differentiate them from the ones typed earlier. . .

In picture7.2 you can see that the list generating routine is placed after the input form in the program and the database has been filled with some test data.

For a more aesthetic appearance you can use a css file again, or you can format the HTML code directly.

Picture 7.2. Creating a List

7.2. Review Questions

1. With which control structure can you process lists?
2. How do you stop data processing loops?
3. With which commands can you access the console of the DBMS?
4. With which commands can you modify SQL tables?
5. How do you connect to a MySql server?
6. How do you wrap lists when displaying them?
7. How do you return to another URL?

8. What do you use META commands in the program of web pages?
9. How do you read fields from the result of queries?
10. How can you utilize long lists?

8. File Management

8.1. Uploading Files to the Server

Now, the form and the program that belongs to it only lack one thing. Uploading of pictures has not been programmed yet. When uploading, the paths which belong to the pictures are inserted into the database but the pictures are not uploaded to the server so they cannot be displayed.

Unfortunately, uploading files is much more complicated than uploading simple texts, as for uploading pictures requires the services of FTP (File Transfer Protocol). The PHP language has a solution for this problem as well.

For uploading files you must properly set the permissions of a folder of the web server selected for this purpose. In Linux it is possible with the chmod command.

There are three different permissions for all the files and folders of a Linux system. R is the permission to read, W is to write, X is to execute. Besides these they differentiate three types of users, namely the owner, the group and others. The owner is the one who created the file or got it to own. The people who belong to the group have collective permissions but do not have the individual permissions of the owner. The third type of user is everyone else (other) who is not the owner or a group member.

Note 8.1. Obviously in this context the word “owner” can mean a program or even a virtual device. . .

Based on the set permissions all three user types can have the permissions to read, write and execute the particular file or folder, which permissions are to be set by the chmod command.

To be able to copy files into a folder and to be able to display them on a web interface you must give the read and write permissions to everyone else.

If you want to give everyone read and write permissions, all the permissions to the owner and none to the group, then you must give the following command: chmod 706 images

You can give the various permissions with numbers. The value of permission R is 4, the value of W is 2 and the value of X is 1. The word images marks the name of the folder, which is to be created with the mkdir images command before setting the permissions.

On effect of the chmod command the owner gets all three permissions but the group, due to 0, gets none. Everyone else gets write and read permissions (4 + 2 = 6). It is unfortunate to grant permission to execute, especially for everyone because that would lead to a security risk.

Note 8.2. If you cannot establish terminal connection or you do not want to bother with complicated commands, get in touch with the operator of the server and ask him to do the settings. . .

Now you can begin completion of your former program. If you want to write a nice program it is worth it to create a function for the uploading of the file, which can be called from other modules. For the sake of simplicity we extend the data input program with the appropriate parts.

```
<?

  require_once "dbmodule.php";

  $safe_filename = preg_replace(
     array ( "/\ s+/" , " /[^-\.\w]+/" ),
    array ( "_" , "" ) ,
```

```
    trim($_FILES['pic']['name']));

  $safe_filename = rand().$safe_filename;

  move_uploaded_file(
        $_FILES['kep'][' tmp_name ' ],
        "images/".$safe_filename);

  $c = connect();

  select_db("webbook");

  $querystring =
  "insert into users (name, picture, cv)
   values('".$_POST['name']
            ."','"
            ."images/"
            .$safe_filename."','"
            .$_POST['cv']
            ."')";

  query1($querystring);

  close($c);

  echo '<meta http-equiv="refresh"
     content="0;URL=index.php?d=1">';

?>
```

Program 8.1. File Management

As you can see it in the source code of the extended program, the file upload came before the database management transactions. The reason for this is that you have to insert the file name into the users table after the copying transaction to make it possible for the displaying program to load the uploaded picture into the client's browser.

It is good to test the size and type of the picture (we have not done that yet...) and change its name to answer all criteria. With the help

of a regular expression you remove the unwanted characters, spaces and with the help of random number generator you make the filename unique.

Note 8.3. Regular expressions are type 3 Chomsky 3 grammars which are suitable for making general expressions with which you can search and replace text passages based on patterns Type 3 grammars are recognized by deterministic finite automatons [6] [7]. . .

Having created the proper file name, you copy the file into the images folder of the server with the move_uploaded_file function. You have already set the appropriate permissions on this folder.

8.2. Picture Management and Resizing

Immediate file uploading is not the most ideal solution in case of pictures if you do not reduce the size of pictures to the possible minimum. This can become a problem if the number of records is growing. It is unnecessary to bother the server with a huge number and size of pictures. In addition your page will load slowly. If you wish to help the problem caused by the size of the pictures then you have to resize them at uploading, which is possible by calling the proper function. Source text 8.2 shows an example of this.

Note 8.4. In the sample program, while uploading, you create a smaller version of the original picture which will have the width of 130 pixels. You need the smaller picture to be display in the list.

By clicking on this picture the user gets the original one. Unfortunately with this solution instead of reducing disk space requirements you increase it, but the loading of the page becomes faster by showing only the small pictures in the list. . . .

```
<?

  require_once "dbmodule.php";

  $safe_filename = preg_replace(
      array("/\s+/","/[^-\.\w]+/"),
      array("_",""),
```

```
      trim($_FILES['pic']['name']));

  $safe_filename = rand().$safe_filename;

  move_uploaded_file(
        $_FILES['pic']['tmp_name'],
        "images/".$safe_filename);

/*===Resizing the picture===============*/

  $bigpic = $_FILES['pic']['tmp_name'];

  $smallpic = "small_".$safe_filename;

  system("convert -resize 130 "
              ."$bigpic $smallpic ");

/*===Uploading small size picture=======*/

  move_uploaded_file(
          $_FILES['pic']['tmp_name'],
          "images/".$smallpic);

  $c = connect();

  select_db("webbook");

  $querystring = "insert into users
     (name, picture, cv) values
     ('".$_POST['name']
        ."','"
        ."images/"
        .$safe_filename
        ."','"
        .$_POST['cv']."')";

   query1($querystring);

   close($c);

   echo '<meta http-equiv="refresh"
```

```
      content="0;URL=index.php?d=1">';

?>
```

Program 8.2. Resizing Pictures

Having modified uploading, you must also change displaying. Open the output.php file and change it following program list 8.3.

```
<?

  require_once "dbmodule.php";

  $c = connect();

  select_db("webbook");

  $querystring = "select * from users";

  $result = query2($querystring);

  echo "<table width=100% border=1>";
  while (list($id, $name, $pic, $cv)
    = mysql_fetch_row($result))
  {
    echo "<TD>
          <table>
           <tr>
             <td>$id</td>
             <td>$name</td>
          </tr>
         <tr>
          <td colspan=2>
         <ahref=".$kep." target=_blank>
          <imgsrc=small_"
            .$pic."width=120 border=0>
         </td>
       </tr>
      <tr>
        <td colspan=2>$cv</td>
      </tr>
```

```
      </table>
    </TD>";
  }
  echo "</table>";

  close($c);

?>
```

Program 8.3. Modified Displaying with Pictures

Complemented with the new parts, the database management application is ready. Obviously some modifications are required and if you thoroughly examine the source, you can surely optimize it.

As we have not paid attention to security, do not publicize this program.

Unfortunately the input form would fail at the simplest hacking attempt. The input fields which store data in a database are most vulnerable to the sql injection hacking method.

Note 8.5. To be able to comprehend this hacking method, imagine an input field where users type in their passwords. The program starts the query the following way:

```
select * from table where
      login='$logname' and pwd='$passwd'
```

It is interpreted as the following after substituting variables:

```
select * from table where login='name'
                        and pwd='password'
```

Seemingly, this part is flawless but in reality this is a vulnerability in the program if you fail to protect the input field by disabling special characters and by other tricks.

Why do special characters hold a threat? Imagine the situation when you know the name of one of the users but you do not know his password and you enter logname='username' '—into the input field.
The query will be changed due to the use of dashes in the following way:

```
select * from table where
     login='name' -' andpwd='anything'.
```

The part after the -' will be ignored by the system so all the data of the user will be displayed on the screen
Another way of hacking the input fields is to place new items in the query.

```
select * from table where
          login='username' pwd='' or 1=1
```

You can achieve that by typing the following into the input field of the password.

```
' or 1=1 –
```

The expression in the where clause is always true since after the or operator the 1 = 1 always returns true, so the system will verify any password.

Obviously this is a very simplified example but it is enough to help us comprehend the importance of protecting the input fields [9].

There are other cracking methods as well, like placing scripts in input fields or using various phishing programs.

You can use the htmspecialchars and the strpslashes functions, which filter the input characters based on parameters set by you, to establish minimal security.

You must place these two function embedded together at the insert transaction, before the data get into the table, this way harmful text passages will cause less problem while processing and displaying...

8.3. Review Questions

1. Describe the permission system of UNIX systems.
2. How do you set file permissions to execute them on a web server?
3. What permission system is used in Windows systems?
4. How do you protect the input fields of web pages?
5. What does SQL injection mean and how do you set up a defense against it?
6. What attack forms are used for cracking web pages?
7. What protocols do you use to copy files to web servers?
8. What protocols do you use to give remote commands to the server computer?
9. Which PHP function do you use to copy files to the web server?
10. What permissions do you have to set for the folder of the server if you want to copy files into it.?

9. Administration of Web Pages

If you try out the newly created data input and the picture uploading that belongs to it, you will soon realize that they must be extended.

The list, created by users, is becoming longer and longer and can cause various problems.

On the one hand it is necessary to wrap by pages due to the length of the list. On the other hand you must secure the possibility for a certain person or the owner of the particular picture to delete Solving the first problem is simple, solving the second one is a bit more complicated, but not impossible.

9.1. Paging Lists

You can implement paging on the database level in a way that you limit the SQL command in displaying, namely you define how many records and from which record would you like to get the result.

```
select  * from users order by name l
                              imit 0 ,10
```

In the query in the example you order the list of records by name in ascending order, and finally place the limit clause in the end, which will always return 10 records beginning with number 0.

(Obviously, if there are less than 10, it cannot return 10).
The query is not working yet as your aim is not to display the first 10 records but always the current 10 ones.

To implement that you must place a link or button with the captions ”forward” and ”last one” before or after displaying, which will shift the list. You must also keep a record of where you are in counting.

```
<?

  if ($_GET['db'] + 10 < $max)
  /* the $max variable comes from the
     total number of records in the
     database */
  {
    $db = $_GET['db'] + 10;
  }
```

```
  $d = $_GET['d'];
  echo "
  <ahref=index.php?d=".$d."&db=".$db.">
    Next"
  </a>;
  ...

?>
```

Program 9.1. Wrapping the List - Paging

The creation of the links with the captions previous, first, and last must be based on program list 9.1.

Paging to the first and last pages is not difficult because by doing so you need to use the 0,10 and the max-10,10 limits.

This code snippet is complicated enough to be placed in a function, then you use the function with proper parameters to shift the different lists. Input parameters can be the maximum number of elements or the number of records you want to see. The return value is the serial number of the current record or a list that contains the links necessary for shifting.
You can also write paging into a separate file and activate it with the include or the require functions where appropriate.

With the introduction of paging the limited query is modified in a way that after the limit clause you place the current number instead of 0, so the query will always return the next 10 records. (this version can be seen in program list 9.2)

```
select * from users order by name
            limit ".$_GET['db'].",10"
```

Program 9.2. Limited Select

Which version you choose, is up to you, but always pay attention not to leave elements out of the list or not to display an empty list on screen. Most problems are due to miscalculations,

miscalculation of elements whose serial number is the same as the number of records on the page at the same time. (in the example serial number 10 can be a problem). When counting, beware that indexing begins from 0 and not from 1.

Note 9.1. Indexing from 0 is often a problem in languages where array and list indexes do not begin with 1. It is difficult because in everyday life, when you have to count something you do not start counting from 0. Get used to having the n element list indexed from 0 to n. . .

9.2. Deleting Records from Lists

After paging try to solve the problem of list length, namely that a user assigned to this role should be able to delete data from the list.
This program need not be written from the basics because your list helping displaying of data is ready.

It is enough to change that source code in a way that you create a form around every displayed record, a form that contains the value of the unique identifier, the value of the id field, of the record and a submit button to initiate the deletion of the record.

Note 9.2. You can also create a form and identify the record to be deleted with the name of the submit buttons but this solution is much more complicated and harder to develop. . .

So the task is to open the output.php file and save it as admin.php.

Note 9.3. Do not integrate this file, or its content, in your main homepage because only a few users see the admin page and thus you can avoid a lot of cracking attempts. . .

Later you will protect this file with a password to protect it from the access of unauthorized people who could delete data from the database.

After saving the program you need to modify the displaying loop based on program list 9.3.

Note 9.4. If you have extended your program with parts necessary for paging earlier, then you are lucky, as that makes it possible to page the list of delete. . .

```
<?

  require_once "dbmodule.php";

  $c = connect();

  select_db("webbook");

  $querystring = "select * from users1
                            order by name";

  $result = query2($querystring);

  $i = 0;

  echo "<table width=100% border=1><TR>";
  while (list($id, $name, $pic, $cv)
    = mysql_fetch_row($result))
  {
    if ($i % 2 == 0)
    {
      echo "</TR><TR>";
    }

    echo "<form action=torles.php
                            method=post>
   <input type=hidden name=id value=$id>";

   echo "<TD>
          <table>
           <tr><td>$id</td><td>$nev</td>
           <td>
             <input type=submit value=X>
           </td>
           </tr><tr>
             <td colspan=3>
```

```
              <img src=".$pic." width=60>
             </td>
            </tr>
              <td colspan=3>$cv</td>
            </tr>
          </table>
        </TD>";
    echo "</form>";
    $i++;
   }
   echo "</TR></table>";

   close($c);

?>
```

Program 9.3. Deleting Records

As you can see, by modifying the former program you have achieved that it is possible to delete from the list, however, you must still create the program region that deletes the particular record from the database.

Writing that program is not a big deal, as you can modify the former insert routine to make it capable of deleting records.

If you examine the code you can realize the necessary steps of modification. The program will only differ in the parameters of the SQL command and you will have to return after executing the transaction.

Of course, management of the pictures requires some extra work here as well. Now you do not have to insert files, you need to find and delete them from the database based on their name, using the unlink function.

```
<?

  require_once "dbmodule.php";

  $c = connect();
```

```
  select_db("webbook");

  $search =
     "select picture from users1 where
                        id=".$_POST['id'];

  $fresult = query2($search);

  list($file) =
      mysql_fetch_row($fresult);

  if (is_file($file))
  {
    unlink($file);
  }

  $querystring = "delete from users1
                 where id=".$_POST['id'];

  query1($querystring);

  close($c);

  echo '<meta http-equiv="refresh"
        content="0;URL=index.php?d=2">';

?>
```

Program 9.4. File Management

Note 9.5. Notice that your code is more and more redundant, program parts only differ slightly from each other. The cause of the difference is the different parameterization of the program regions.

This means that it would be worth it to consider the structure of the source text and structure the program regions into functions. The best solution would be to think over the introduction of classes. . .

To run the program, enter the path and the name of the file into the URL field of the browser program.

```
http://serverhost.hu/~username/delete.php
```

Note 9.6. If you find this method uncomfortable, place a menu item among the others that loads the list of delete into the browser. . .

Delete works similar to insert, they only differ in sql level transactions. You store the identifiers of records in the hidden field of a list generated with the loop, which identifier the program, prepared for executing the delete, will get from the form after clicking on submit.

The first query finds the name of the file that belongs to the record and enables the unlink function to delete it, parameterized with the name.

Of course, before the program begins delete, it tests if a file with the name in the record exists or not.
Having completed the delete transaction, your only task left is to delete the record from the database, based on the identifier stored in the id variable.

The META command leads you back to the list which of course, has been refreshed.

The above illustrated transactions are perceived by the user as if the record to be deleted had simply disappeared from the list.

9.3. Simple Password Protection

The final step to create the administration interface is to write a password login. For login we have chosen a very simple solution, which is not perfectly safe but will certainly be appropriate for us.