Developing web pages

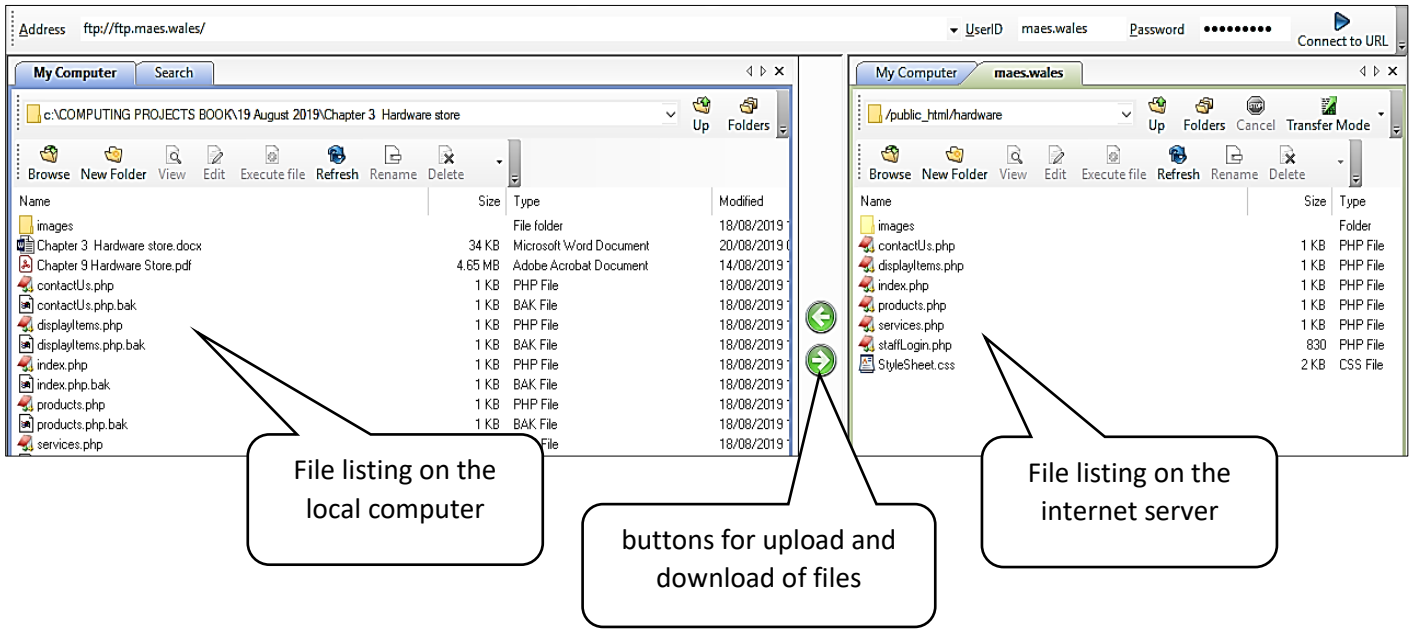
1

Internet programming environment

To undertake the example projects in this book, the first requirement is the use of an internet server. This may be arranged by the IT services department in a school or college, or you may subscribe at reasonable cost to a commercial Internet Service Provider. In addition to running standard web pages, the use of an SQL database will be required. Database facilities are usually a standard feature of ISP web site accounts.

Internet pages will be accessed in a web browser by means of a domain name. The Internet Service Provider may allow customers to select a domain name when setting up an account, or a domain name may be purchased separately from a registration agency and then linked to the server account.

Web pages are constructed from a variety of files, some containing program code and others providing media content such as photographs. It is important to develop a well organised folder structure on both the local computer used for web development, and on the server. Files are transferred between the local computer and the server by means of an FTP client application. Suitable programs may be downloaded from the internet, either free of charge or for a small subscription.



After transfer to the server, the current version of the web page can be run on-line in a web browser. Corrections or additions may then be made to the files on the local computer, and the revised version uploaded and tested. This process is repeated until the web page design and functionality meet the project requirements.

Care should be taken to keep regular backups on the local computer of all web page files and the SQL database. In the unlikely event of the server failing and the on-line content being lost, it will then be possible to easily reconstruct the web site on an alternative server.

SQL database

Most web sites will require the use of a database. This may be to hold page content which is uploaded and displayed when the page runs, or to receive and store customer orders or bookings which are made from the web site.

For the example projects in this book, an on-line SQL database will be required. An Internet Service Provider will generally make a suitable database program available on the server, and provide the access codes necessary to update the database content.

The SQL database may be accessed using a standard web browser. A management program such as **phpMyAdmin** provides similar functions to a stand-alone database such as Microsoft Access:

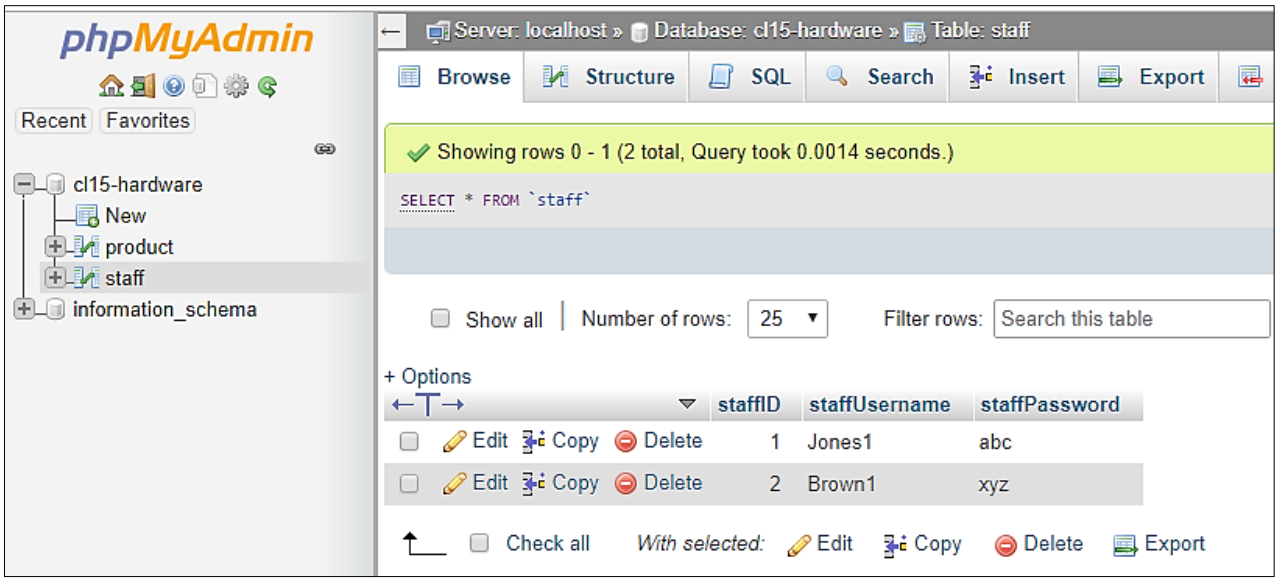
- New tables may be set up, with the fields and data types specified. Field types particularly useful for web site development include:

  Number: **integer** which represents whole numbers; **real** giving a variable number of decimal places as required; and **decimal** giving a fixed number of decimal places, such as two when representing currency.

  String: **varchar** which is a string of defined length such as 30 characters; **text** and **longtext** which have unspecified lengths up to fixed maximum sizes.

  Date specifying year, month and day; **time** specifying hour, minute, second and hundredths of a second; and **datetime** which combines the previous two.

- A primary key may be selected, and there is an option to set the key field values automatically using an auto-number function.
- Records may be added, edited or deleted in the database tables.



- As an alternative to manually updating records on screen as shown above, the phpMyAdmin software can run commands written in Structured Query Language to update records. When this is done, any errors in the SQL code will be identified and an error message displayed. This can be a

useful way of testing and debugging SQL commands which are intended to run directly from your own web pages but are not functioning as expected.

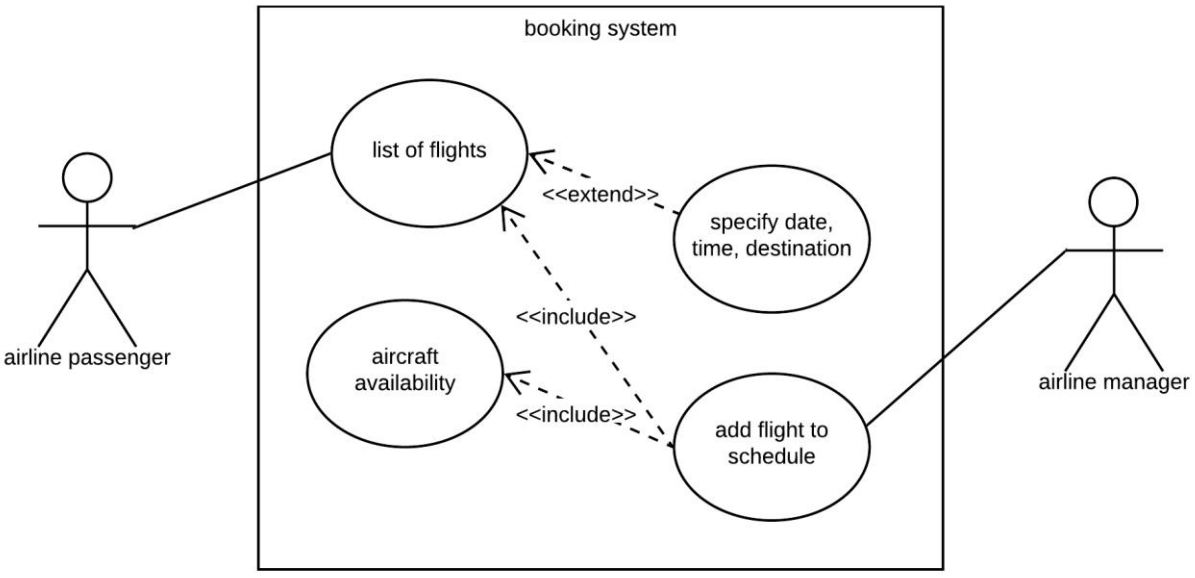
- Backups may be made of the whole database or individual tables. The files may be used to restore the data if corruption occurs.
- Tables may be copied using a different name. This is a useful starting point if a similar table, such as a list of products or a list of customers, is needed in more than one project.

Design documents

Conventions have been developed for a standardised range of computer software design documents in a system known as **Unified Modelling Language**. UML diagrams will be used where appropriate in this book.

Design often begins by developing a **use case** diagram. This specifies the objectives of the application and identifies the persons or groups who will be involved in its use. At this stage, we are not focussing on how a program might actually achieve these objectives. As an example, consider an on-line airline booking system. The computer system is represented by a rectanglular outline, whilst the users of the system (known as actors) are represented by stick figures outside the rectangle. The primary user for whom the system is designed is the airline customer, so they are shown to the left of the rectangle. The secondary user who will maintain and update the system is the airline manager, shown to the right.

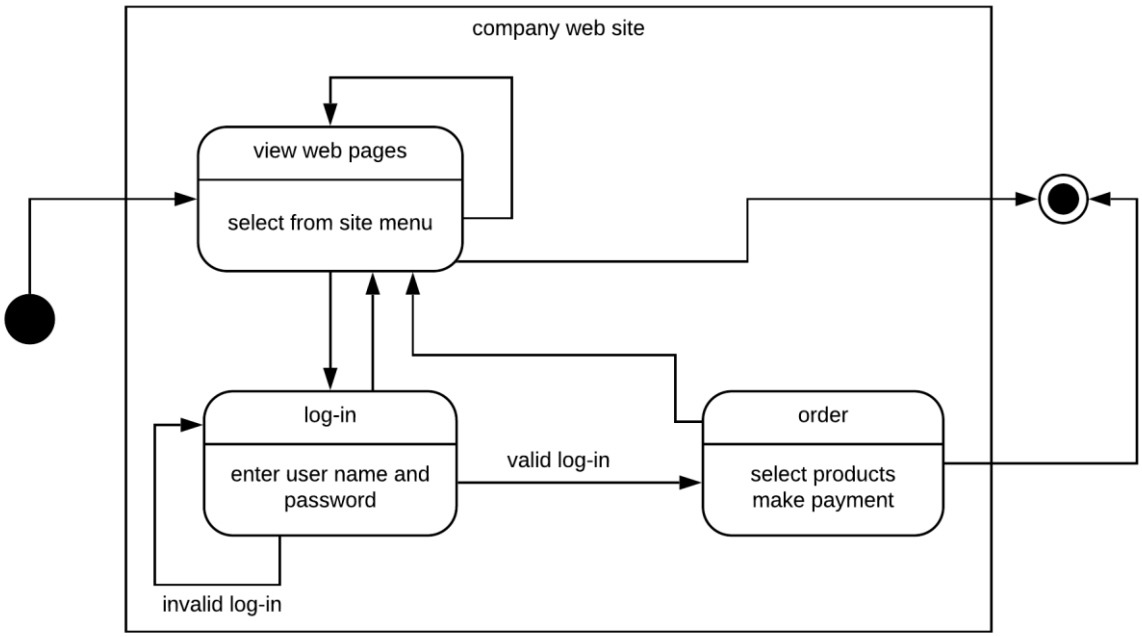
The passenger goes to the web site to obtain flight information. They may simply scroll through the list of available flights to obtain the information they require. However, they may prefer to make a search using such criteria as flight date, time or destination. Since the search function is optional, it is linked by an arrow pointing to the list of flights and marked with an <<extend>> label.



The airline manager will be responsible for adding flights to the schedule. Before a flight can be added, the manager must check that an aircraft is available. Since this is an essential operation, an arrow is drawn to the aircraft availability check and marked with the label <<include>>. Similarly, once a flight has been

scheduled then it is essential that this is added to the list of flights. An arrow is drawn to list of flights and again marked as <<include>>.

Once the general requirements of a project have been established, the detailed operation of individual sections of the system can then be considered. A useful tool for this work is a **state transition diagram**. As an example, consider part of an on-line shopping site.

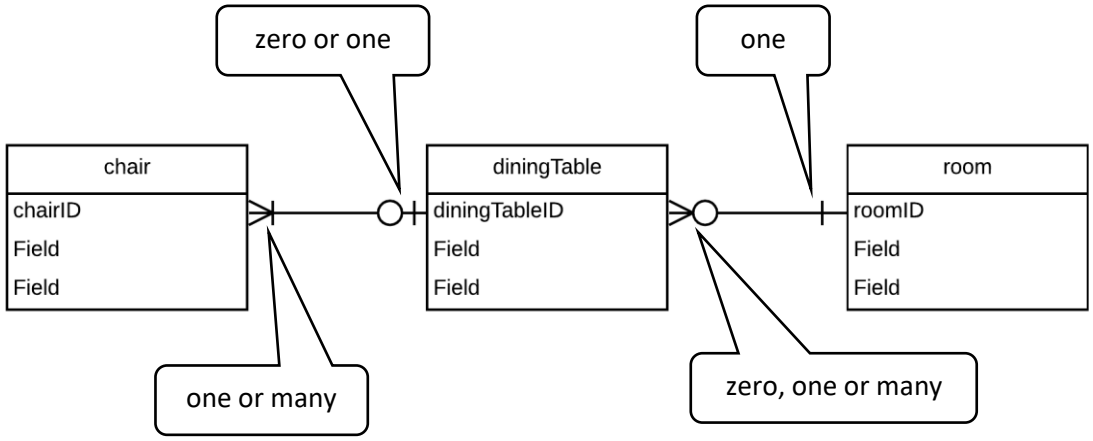


The computer system is outlined by a rectangle. The starting point for entering the web site is the black circle to the left of the diagram. Customers may then view the site, moving between pages by selecting from menu options. The upper section of the double box specifies the current situation or state, while the lower section indicates activities which may take place at this time.

If the customer wishes to make a purchase, they must go to a log-in page and enter their user name and password. If the log-in details are verified as correct, the customer is taken to an order page where they can select the goods required. Payment must then be made. After completion of the order, the customer may return to the main website, or may exit from the site at the outlined circle on the right of the diagram.

Before beginning detailed program design, it is often important to consider the database requirements of the project. An **entity-relationship diagram** is generally used to specify the structure for a relational database. In addition to listing the fields for each table, the possible relationships between tables are shown by linking symbols. As an example, imagine a database is set up to provide an inventory of furniture in a hotel. Three entities which might be included are **chairs**, **dining tables** and **rooms**. Relationships are shown:

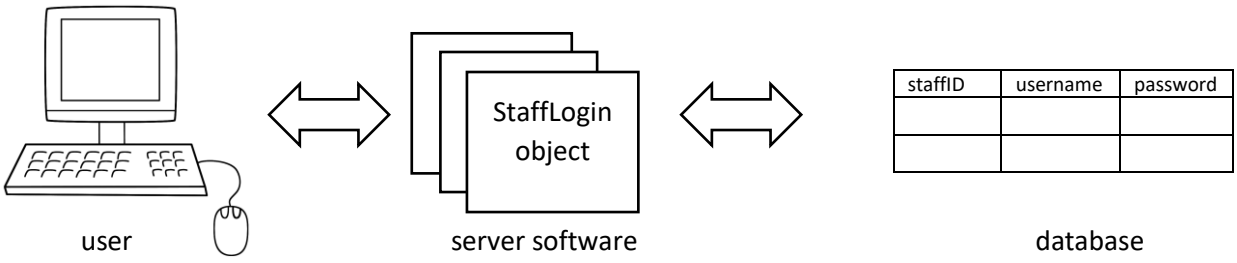
- A chair may be linked to zero or one dining table: it may be an individual arm chair in a lounge, or may be located at a table in the dining room. A dining table may have one or more chairs.
- A room may have zero, one or more dining tables: for example, it might be a lounge with no tables, a staff room with one table, or the guests' dining room with many tables. Each dining table will be allocated to one room.



It is important that records have primary key fields, so that a record can be uniquely identified without the risk of an incorrect selection. Some types of record automatically contain a unique key field, such as a National Insurance Number for a member of staff or a Vehicle Registration Number for a car. However, it is often necessary to allocate primary key values. The convention adopted in this book is to use the same name as the corresponding table, with the letters 'ID' added. This makes it easy to identify the primary key within the list of fields of any table. As far as possible the primary key values will be allocated automatically by the database software using an auto-number function. This ensures that no values are accidentally duplicated.

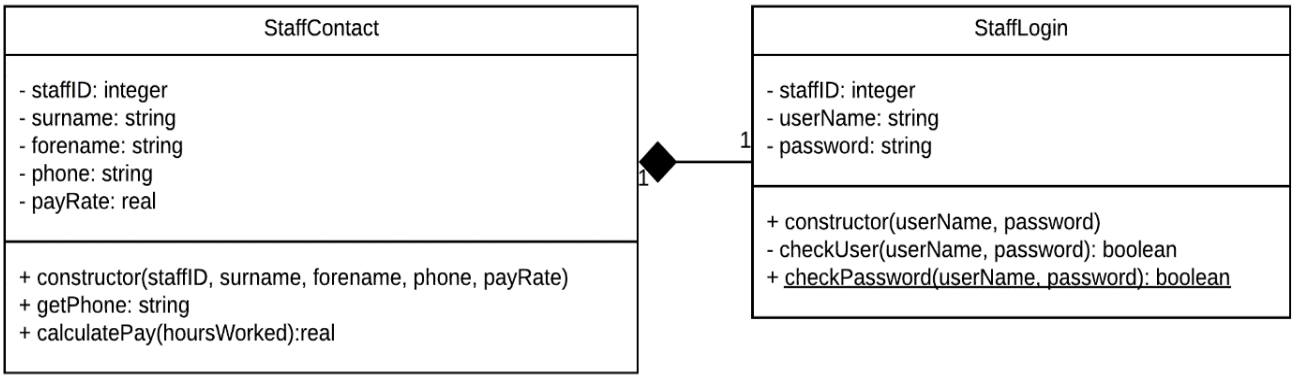
Object oriented programming

An object oriented approach to programming will be used in this book when appropriate. In particular, classes of objects will be used as an interface between the user and the on-line database. For example, several programs require a log-in user name and password to be verified against entries in a **staffLogin** table. The approach used is to create a set of **StaffLogin objects**, which contain usernames and passwords as attributes. A method in the **StaffLogin class** can then check for an object with attributes matching the data entered.



Class diagrams are used to specify the attributes and methods of objects within each class, and the relationships between different classes. The login function described above may form part of a personnel records system. In this example, two classes have been defined: **StaffContact** and **StaffLogin**.

Each **StaffContact** object contains the staffID, name, phone number and hourly pay rate for a member of staff. These are listed in the upper part of the box below the class name. Data types, such as **string** for the telephone number, are specified. Notice that each attribute is preceded by a minus (-) symbol to indicate that it is **private** and can only be accessed by a method within the same class.



In the lower part of the box are listed the methods which can carry out actions involving **StaffContact** objects. The first is a **constructor** method, which creates new objects within the class and sets the values for the attributes. For example, if a new member of staff is appointed then a **StaffContact** object would be created containing their name, phone number and rate of pay.

The other two methods shown in the diagram will obtain the phone number for a particular member of staff, and calculate their pay based on the number of hours worked. The data type returned is specified: string in the case of the telephone number, and a real number for the amount of pay. Notice that all the methods are preceded by a plus (+) symbol to indicate that they are **public**. These methods can be called by any other part of the program as required.

Another class **StaffLogin** is shown which contains the computer usernames and passwords for each member of staff. The classes are linked by a line carrying a black diamond symbol. This indicates a dependency between objects in the two classes; if a **StaffContact** object ceases to exist because the member of staff has left the company, then the corresponding **StaffLogin** object would also cease to exist. Notice the small numbers at the ends of the link line, indicating that each **StaffLogin** object is related to one **StaffContact** object, and vice versa. The **StaffLogin** class contains the private attributes staffID, username and password, and again provides a public **constructor** method.

The principal function of the **StaffLogin** class is to check whether log-in details entered by a user are correct. This task is carried out by the **checkPassword( )** method which is made public. Only one instance of the **checkPassword( )** method is necessary for the whole class, so this is known as a **static method** and is shown by underlining in the diagram. The **checkPassword( )** method then calls each of the **checkUser( )** methods belonging to each individual staff object to determine whether correct log-in details can be found. The **checkUser( )** method is therefore only accessed within the class, and can be designated as private. When the checks are completed on all staff objects, the **checkPassword( )** method will return a true or false Boolean value to indicate whether the log-in was valid.

HTML and CSS

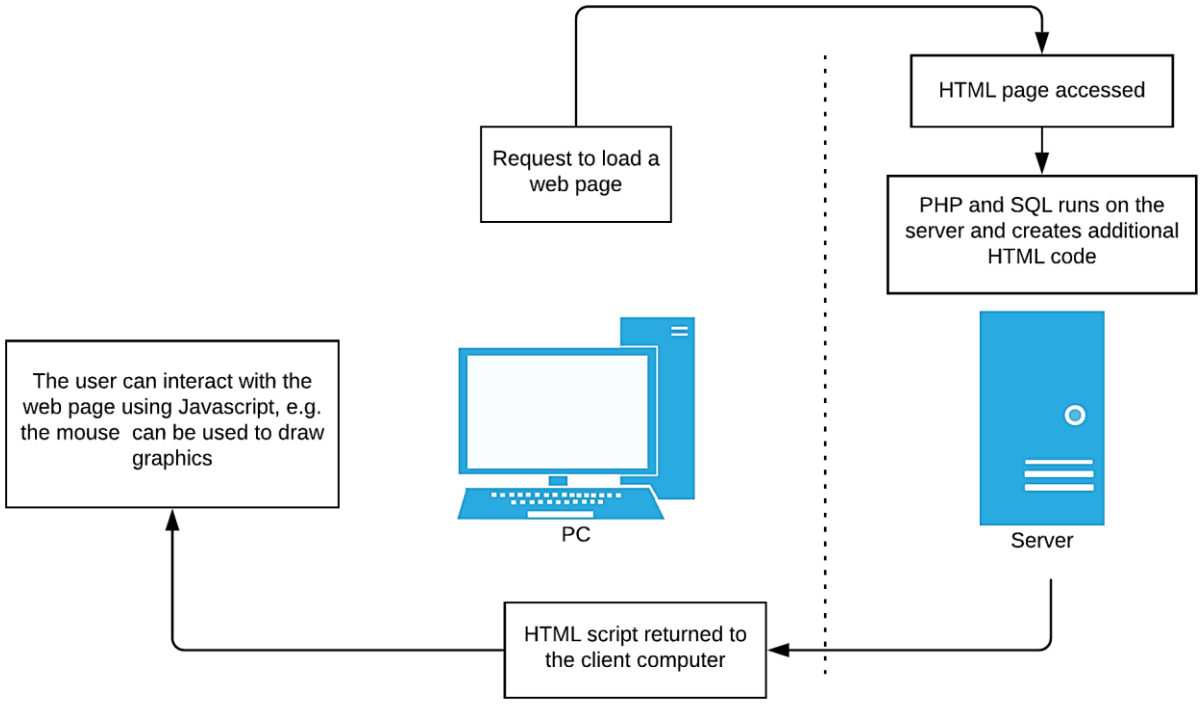
Websites are generally written in **HTML** (HyperText Markup Language) which provides the main display elements: text including headings, lists and tables, photographs and other media, and interactive components such as buttons, text input boxes, tick boxes and drop-down lists.

Cascading style sheets (**CSS**) provide a way of modifying the appearance of the basic HTML components, such as: producing coloured backgrounds for the rows or columns of a table; or changing the position, size and appearance of buttons to create a set of menu options on a web page. CSS styles may be applied to all HTML components of a particular type, or may be restricted to only a section of a web page by defining **classes** or **divisions** within the page.

HTML and CSS code may be written in a plain text editing program, with Microsoft Notepad being the simplest. However, it can be helpful to use a text editor such as **EditPlus** which is designed specifically for web site programming. These applications use colour coding to identify different components of the program code, such as variables, text strings, or commands. This makes it easier to understand the structure of the program code, and can help in locating errors such as misplaced speech marks.

HTML can provide simple input of data and allows the user to navigate between the different pages of a web site. However, it is not designed to carry out calculations, draw graphics by means of geometry, or transfer data to or from a database. To carry out these more complex operations on a web page, a programming language is needed.

Web programming languages can be divided into two groups: those that run on the server, such as **PHP** and **SQL**; and those that run on the local computer, such as **JavaScript** and **p5.js**. Imagine a case in which a user wishes to load a web page to display details of different garden plants including photograph, text description of growing conditions, and purchasing information. The page will then allow the user to set up a plan of their garden and add selected plants to the design. The page will operate in several stages, as shown below:



- The request to load the page is passed to the server. PHP program code within the web page script will run an SQL query to obtain the garden plant information from a database table. This information will then be formatted into a screen display by writing HTML code which will be inserted into the web page script.
- The web page script is now downloaded to the local computer. Additional resources requested by the HTML code, such as the photographs of the plants, are downloaded from the server and added to the screen display.
- The user may then interact with the screen display to input the garden design. They will be working with JavaScript code on their local computer in a similar way to using a stand-alone software application.

PHP language

PHP (Pre-Hypertext Processing) can create new lines of HTML code which become part of the web page which is downloaded. PHP code is included in the web page script between <? bracket symbols:

```
<p> This is HTML code
<?
        echo "<p>This is PHP code";
?>
<p> This is HTML code again
```

When the web page is accessed from the server, the **echo** command creates a new line of HTML code and the PHP program line is removed. The script which is downloaded to the local computer now becomes:

```
<p> This is HTML code
<p> This is PHP code
<p> This is HTML code again
```

Loops within PHP can be used to generate multiple lines of HTML code for display on the web page. For example, the **for** loop in this program will operate four times. Variables in PHP are indicated by a $ symbol.

```
<p> Members taking part:
<?
        for ($i=1; $i<=4; $i++)
        {
                echo "<br> Group ".$i;
        }
?>
   <p> The event starts at 10.30am.
```

This program would generate the HTML script:

```
<p> Members taking part:
<br> Group 1
<br> Group 2
<br> Group 3
<br> Group 4
<p> The event starts at 10.30am.
```

In addition to carrying out loops and conditional IF.. statements, PHP can make calculations. For example, we might calculate VAT on an order and output this value on the web page.

```
<?
    $cost = $price * $quantityOrdered;
    $VAT = $cost * 17.5/100;
    $total = $cost + $VAT;
    echo "<p>Total price: £".$total;
?>
```

When run, the amount would be calculated, then the result inserted into a line of HTML code for display on the web page, for example:

```
<p>Total price:  £67.85
```

File operations

PHP can be used in combination with Structured Query Language to carry out database functions. Connection is made to the database and a query is run. A database operation typically has the structure:

```
$conn = new mysqli(localhost, $username, $password, $database);
if (!$conn) {die("Connection failed: " . mysqli_connect_error()); }

$query="     ";         //the SQL query to carry out the required operation

$result=mysqli_query($conn, $query);
$num=mysqli_num_rows($result);
mysqli_close($conn);
```

The command outlined above will depend on the operation required:

- To add a new record to an existing table, the INSERT command is used. For example:

```
$query="INSERT INTO product VALUES ('$stockcode','$title',
                                    '$description', '$price')";
```

  This instruction would add a record to a **product** table using field values specified by the variables: $stockcode, $title, $description and $price.

- Records can be retrieved from a table using a SELECT command. For example:

```
$query="SELECT * FROM product";
```

  This instruction will obtain all records from the **product** table. The * symbol indicated that all fields should be downloaded. If only some fields are required, the field names can be listed in place of the * character.

- Existing records can be modified using the UPDATE command. For example:

```
$query = "UPDATE product SET description='" .$description."', price='".$price."'
                WHERE productID='".$productIDwanted."'";
```

  This instruction will set the values of the **description** and **price** fields of a **product** record. The record to be changed is identified by its **productID** value.

- A record can be removed from a table using the DELETE command. For example:

```
$query = "DELETE FROM product WHERE productID='".$productIDwanted."'";
```

  This instruction will delete a record from the product table, identified by its **productID** value.

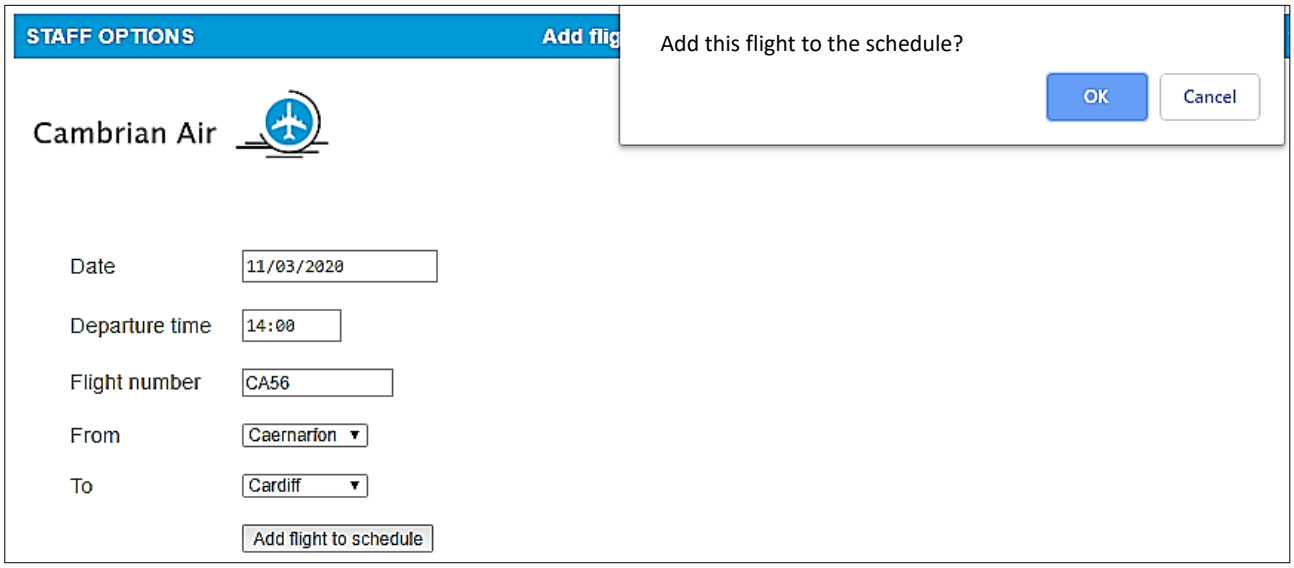
After running a query to obtain records from a table, the **mysqli_num_rows( )** function will indicate how many records have been downloaded. This value can then be used as a loop counter if a loop is used to display the records on a web page.

JavaScript

JavaScript operates on the local computer without referring back to the server. JavaScript can, for example, respond to mouse movements or button clicks, then take appropriate action such as drawing graphics on screen. For example, a function could be written to determine the horizontal (x) and vertical (y) screen position of the mouse pointer when the mouse is clicked:

```
function mouseDown()
{
    var x = event.clientX;
    var y = event.clientY;
}
```

Important uses of JavaScript are in checking data entries for errors before uploading to the server, and obtaining responses from the user before actions are carried out. For example, confirmation may be requested before carrying out a database operation.



Another function of JavaScript is to produce run-time graphics by means of geometry commands. The graphical display may include interaction and animation, for example in an on-line computer aided design application or artist's drawing program. The program extract below creates a drawing area known as a **canvas**, then adds a white rectangle with a red circle similar to the Japanese national flag.

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
</canvas>

<script>
    var canvas = document.getElementById("myCanvas");
    var ctx = canvas.getContext("2d");
    ctx.fillStyle = "#FF0000";
    ctx.strokeStyle = "#000000";
    ctx.beginPath();
    ctx.arc(100, 50, 50, 0, Math.PI*2, true);
    ctx.closePath();
    ctx.fill();
    ctxt.stroke();
</script>
```

p5.js programming language

Whilst JavaScript is a very effective language for many web page operations, we can see from the previous examples that it is not particularly user-friendly or easy to learn. To improve the ease of use, the language p5.js has been developed as a high level extension of JavaScript to simplify the programming code. Exactly the same Japanese flag program shown above in JavaScript is represented in p5.js by:

```
<script>
    function setup()
    {
        createCanvas(200, 100);
    }
    function draw()
    {
         stroke(0);
         fill(255);
         rect(0, 0, 200, 100);
         fill(255, 0, 0);
         ellipse(100, 500, 50, 50);
    }
</script>
```

In this book, **p5.js** will be used where this makes the programming quicker and easier.

Web site navigation

A key feature of internet programming is the ease with which a user can move between the different pages of a web site. Page requests can be made in HTML code by including hyperlinks selected by mouse click. Suppose that a page 'customers.html' is to be loaded. We may write:

```
<a href='customers.html'> View list of customers </a>
```

The text ' View list of customers' will appear underlined and highlighted in colour, indicating that it is an active link. An alternative approach is to create an HTML **form** which will load the page when a **submit** button is clicked:

```
<form method='post' action='customers.html'>
     <input type='submit' value='View list of customers'>
</form>
```

The caption 'View list of customers' will be displayed on the button.

Other programming languages can also create links between web pages. To load another page within a section of PHP program code, we include a **header** command:

```
if ($displayCustomers==true)
{
   header('Location: customers.html');
}
```

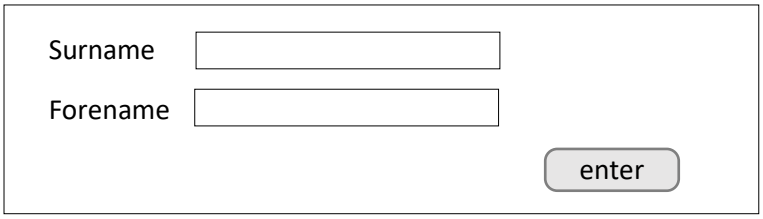
Pages can be loaded in a similar way in a JavaScript program using a **window.location** command:

```
if (displayCustomers==true)
{
   window.location = "customers.html";
}
```

Transferring variables to another page

When another page is loaded, it is likely that data input by the user or created in PHP or Javascript will be needed on the new page.

User data entered in an HTML **<form>** block is automatically carried to the server where it may be transferred to PHP variables. We might, for example, display a form to input a customer's name:



Text boxes are created by HTML **<input>** components and are given appropriate variable names:

```
<form method='post' action='newcustomer.php'>
    Surname
    <input type='text' name='surname'>
    Forename
    <input type='text' name='forename'>
    <input type='submit' value='enter'>
 </form>
```

When the submit button is clicked, the data entered into the text boxes will be carried to the server, and can then be transferred to variables using PHP **request** commands:

```
<?
    $surname=$_REQUEST['surname'];
    $forename=$_REQUEST['forename'];
?>
```

PHP variables can be passed between web pages by including them in the URL address of the page to be loaded. For example, we might link to another web page and wish to transfer a customer name:

```
<?
    echo"<a href='newcustomer.php?surname=".$surname."&forename=".$forename."'>";
    echo"Add new customer";
    echo"</a>";
?>
```

The variables will arrive at the server, and can then be accessed by means of PHP **request** commands in the same way as in the previous example.

Transferring variables between programming languages

Web pages may combine PHP and Javascript code in whatever ways are convenient. As a consequence, programmers often have to transfer variables between PHP and Javascript languages.

Transferring a variable from PHP to Javascript is relatively straightforward. The PHP variable will have been assigned a value by the server before the page is downloaded to the local computer, so can also be made available to JavaScript for use at any time while the page is running. To transfer the value, we simply copy

the PHP value to the Javascript variable using an **echo** command. The following script block will use the PHP variable **$total** to create a JavaScript variable **total**.

```
<script>
	var total = <? echo $total ?>;
</script>
```

Transferring a variable from JavaScript to PHP is slightly more difficult, as this process can only take place on the server. The Javascript variable must first be uploaded. This can be done by including the variable within the page address. For example, the following script will attach the JavaScript variables **surname** and **forename** to the page address **customer.php**.

```
<script>
	window.location = "newCustomer.php?surname="+surname+"&forename="+forename;
</script>
```

We then include PHP **request** commands within the **customer.php** web page. These commands will run on the server before the page is downloaded to retrieve the data values and reassign them to the PHP variables **$surname** and **$forename**.

```
<?
    $surname=$_REQUEST['surname'];
    $forename=$_REQUEST['forename'];
?>
```

Using these various techniques, it is possible to work flexibly in HTML, PHP and JavaScript, transferring data between pages and languages. This allows programmers to make the best use of the special features of each of these technologies as required within any particular web based project.

Program debugging

Checking a web based program for errors can present special difficulties:

Testing and debugging a stand-alone program written in a language such as Java is generally fairly straightforward. The compiler or interpreter software will stop if a program error is encountered and an error message will be displayed. The line where the error occurs will be indicated. However, debugging a web page may not be so easy. HTML is designed to be fail-safe, so lines of code containing errors are simply ignored and the browser continues to display as much as possible of the page. The only indication of an error is that the output is not as expected. JavaScript often simply stops running if an error is encountered. The page freezes and becomes unresponsive, with no error message displayed.

A simple strategy for web page debugging involves two techniques: displaying the values of variables to determine whether these are correct; and isolating sections of code temporarily to stop them from running, as a means of locating program lines which are causing errors.

In PHP, the value for a variable can be displayed by inserting a temporary '**echo**' command. For example, the section of program outlined will display the value of the variable **$invoiceTotal**. This block can be removed when testing is completed.

```
<h2>Customer information</h2>
   <?
       echo "invoiceTotal = ".$invoiceTotal;
   ?>
<table>
   <tr>
```

In JavaScript, a convenient method for displaying the value of a variable is to add an **alert( )** command to produce a pop-up message box. In this example, the values of the variables x and y will be displayed, and the program will pause until the user clicks the '**OK**' button to proceed.

```
function mouseDown()
{
    var x = event.clientX;
    var y = event.clientY;
    alert('x = ' + x + '  y= ' + y);
    x = x + xoffset;
    y = y + yoffset;
```

The **alert( )** box can be removed when testing is completed.

The second technique is to isolate lines or blocks of code where an error is suspected. The code can then be progressively brought back into action until a point is reached where the error re-appears. In this way, the problematic line of code is identified.

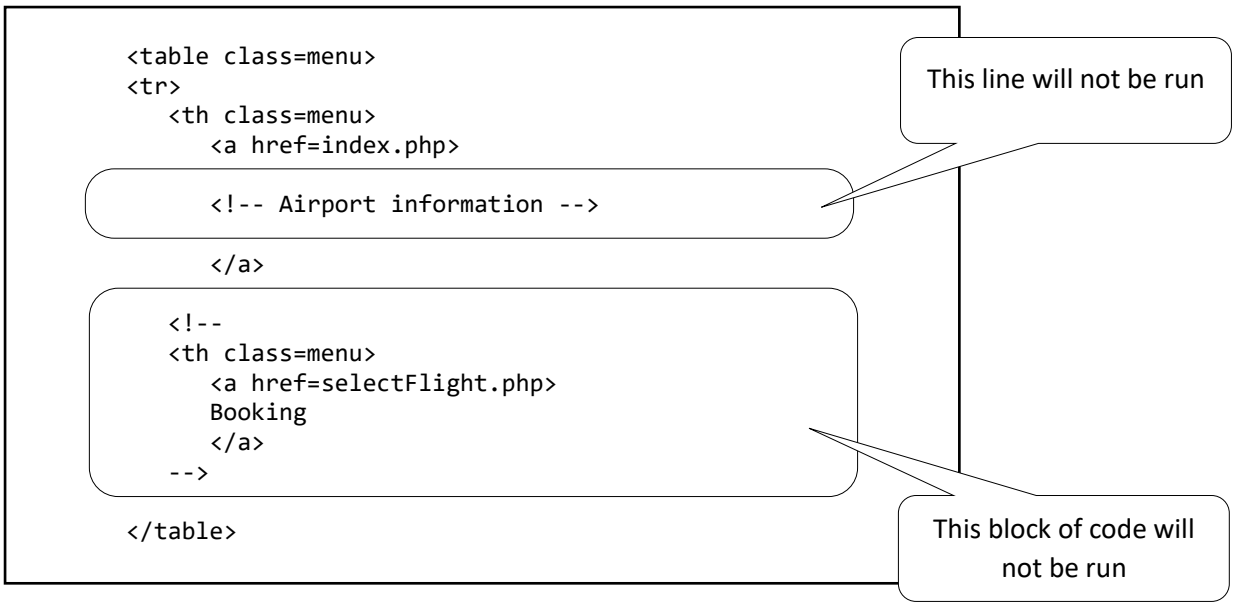
To isolate a single line of code in PHP or JavaScript, a double forward slash // symbol is used. To isolate a block of code, the block is enclosed by the symbols /* and */

```
var n = stockcode.length;
if (n<1)
{
   alert("A stockcode must be entered");
 // error=true;
}
/*
n = title.length;
if (n<1)
{
   alert("A product title must be entered");
   error=true;
}
*/
if (error == false)
     saveRecord();
```

This line will not be run

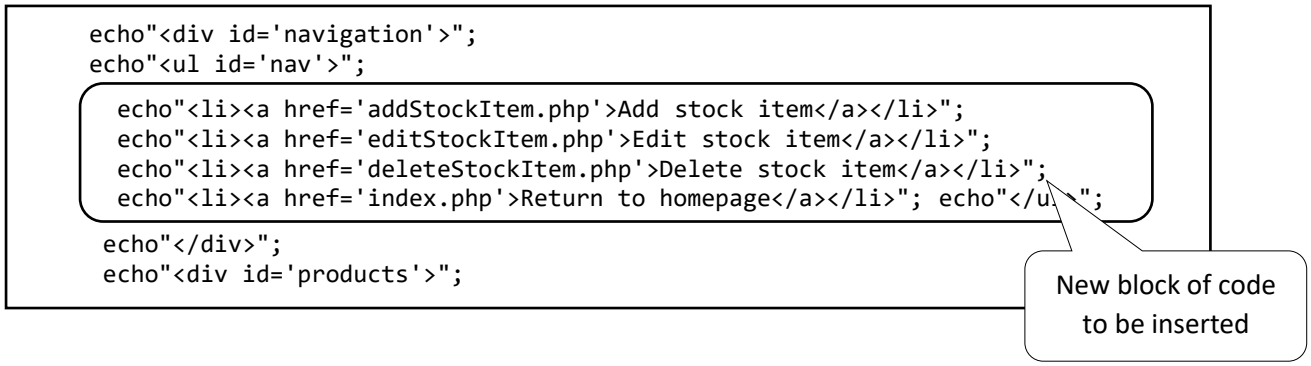
This block of code will not be run

In HTML, a line or block of code can be isolated using the symbols <!-- and -->



Conventions used in this book

Where lines of code need to be inserted into an existing section of program, the new lines of code will be enclosed in a round-cornered rectangle:



In some cases, a single long line of code has to be split onto several lines in the book due to the limited page width. This will be indicated by a curved arrow symbol. When this symbol is shown, it is important that the code is entered by continuous typing without any line breaks, to avoid errors occurring when the program is run.