Chapter 22.

SECTION V: ADVANCED TECHNIQUES

Using SimpleXML

An Introduction To XML

XML, (e**X**tensible **M**arkup **L**anguage), has been defined as a Meta language (i**.**e**.** a language that describes other languages) by the **W**orld **W**ide **W**eb **C**onsortium (**W3C**). XML is part of a large family of markup languages.

XML is **text** that follows certain rules. The rules are very similar to those for creating HTML, but stricter and with some enhancements. XML is a markup language for documents containing structured information. Structured information contains **markup** and **content**.

Structured information consists of words, pictures, numbers and a host of other data (i**.**e**.** content), hence should therefore contain some indication of what role its content plays.

For example**:** Content in a *section heading* has a different meaning from content in a *footnote*, which again is different from content in a *figure caption* or content in a *database table* and so on.

Almost all documents have some structure. A markup language is a mechanism to identify the structure in a document. XML defines a standard way to add markup to documents that both exposes and standardizes the document's structure.

It is important to know that XML is designed to store, carry and exchange data. It is **not designed** to display data.

XML is a **tagged language**. The actual data (i**.**e**.** document content) is contained within structured, **tagged elements** of the document. The XML document is parsed to extract its content.

Why XML?

In order to appreciate XML, it is important to understand why it is created. XML is created so that **richly structured** (*user defined structures*) documents could be used over the web. The only viable alternatives, **HTML** and **SGML**, are not practical for this purpose.

HTML comes bound with a set of semantics and does not provide for arbitrary structure.

SGML provides arbitrary structure, but is too difficult to implement just for a web browser. Full SGML systems solve large, complex problems that justify their expense. Viewing structured documents sent over the web rarely carries such justification.

XML is a simple, very flexible, text format derived (extracted) from complex SGML - **S**tandard **G**eneralized **M**arkup **L**anguage (ISO 8879). XML is originally designed to meet the challenges of large-scale electronic publishing. Today however, XML plays an ever expanding and important role in the exchange of a wide variety of data via the Internet.

What XML Does

XML Is Used To Exchange Data

With XML, data can be exchanged between **incompatible** systems. In the real world, computer systems and databases contain data in incompatible formats. One of the most time consuming challenges for developers has been to exchange data between such systems over the Internet.

Converting to XML greatly reduces this complexity. The data thus created can be easily transported via the Internet and read by many different types of applications.

XML Can Be Used To Share Data

With XML, plain text files can be used to share data. Since XML data is stored in plain text format, XML provides both a software and hardware independent way of sharing data.

This makes it very simple to create data that different applications can work with. It also makes it easier to expand or upgrade a system to new operating systems, servers, applications and browsers.

XML Can Be Used To Store Data

With XML, plain text files can be used to store data. XML, when used with third party application programming environments, can also be used to store data in files or in databases. Applications can be written to store and retrieve information from these files and databases. Generic forms based applications can then be used to display the data retrieved.

XML Can Make Data More Useful

With XML, data is available to more users. XML is independent of hardware, software and application. Data can be made available to more than HTML browsers alone.

Clients and applications can access XML files as data sources, as though they were accessing databases. Data can be made available to all kinds of reading machines (agents), it is simple to make data available for blind people or people with other disabilities.

XML Can Be Used To Create New Languages

XML is the mother of WAP and WML. The **W**ireless **M**arkup **L**anguage (**WML**), used to markup Internet applications for handheld devices like mobile phones, is written in XML.

XML Can Be Used To Reduce Incompatibilities

Often web pages on the Internet are not compatible across different devices such as Browsers, Mobile Phones, PDAs and so on. One of the major difficulties that web designers have today is that people are now accessing web pages using a variety of different devices like PCs, Macs, Mobile phones, Palmtops and even televisions.

Because of this, web designers must now either produce their pages in several different formats to cope up with this or they must cut back on the design in order to make the page compatible across all the possible download formats.

However, if XML is used to define what data means **and** not how it is to be displayed, it makes it very easy to use the same data on several different platforms.

Difference Between XML And HTML

XML is designed to **carry data**. XML is **not a replacement** for HTML. XML actually complements HTML.

HTML is designed to accentuate the look and feel of a web page i**.**e**.** HTML's focus is the **placement and display of data** (i**.**e**.** how data looks) whereas XML is designed to **describe data** and is focused on **what data is**.

When using HTML to create web pages, style sheets are often used. Known formally as **C**ascading **S**tyle **S**heets, they add styling elements to a HTML webpage. Web pages can also be written in XML. The XML **equivalent** of a CSS is **XSL** (e**X**tensible **S**tylesheet **L**anguage), implemented in almost the same way.

Since XML based pages provides more flexibility than HTML pages, XML can be expected to replace HTML as the language of choice in the foreseeable future.

XML tags are not predefined. They are programmer-defined tags. Tags used to mark up HTML documents and structure them are predefined. The author of HTML documents can only use tags that are defined in the W2C HTML standard (like <p>, <h1> and so on). The author of XML pages can define their own tags and their own document structure. As of now however, in Web development, it is most likely that XML will be used to describe the data, while HTML will be used to format and display the same data.

SimpleXML In PHP 5.1

PHP 5.1 has been delivered with a slew of new features aimed at simplifying application development. Such as the introduction of exception handling, the **S**tandard **P**HP **L**ibrary (**SPL**), **enhanced support for XML**, reflection and quite a few enhancements to the object oriented features of PHP.

PHP 5.1 by default installs XML support and offers a new extension, **SimpleXML**. All XML functions are now standardized on the **libxml2** library and are fully **W3C standards compliant**.

SimpleXML provides a traversable structure to work on XML documents, allowing the programmer to simply change values in the structure and write the file out with only a few lines of code.

Early implementation of XML in PHP has been quite rudimentary and requires a fair amount of programming, so it's quite possible to see PHP 4 applications using XML without ever touching its built-in xml functions.

PHP 5.1 offers a replacement extension for **DOMXL** (with the DOM extension). This extension allows working on the XML files using the DOM object model and is superior to SimpleXML particularly when a programmer is uncertain of what document format to expect with the application.

While the DOM is more powerful, SimpleXML is much **quicker** to implement and easier to get hands on for inexperienced programmers. Both of the extensions are robust and well thought out and any can be used depending upon programming needs and taste.

SimpleXML turns an XML document into a data structure, which can be iterated through like a collection of arrays and objects. This is excellent when one is only interested in an element's attributes and text and one knows the document's layout ahead of time.

SimpleXML is easy to use because it handles only the most common XML tasks, leaving the rest for other extensions.

The SimpleXML extension provides a very simple and easy to use toolset to convert XML to an object that can be processed with normal property selectors and array iterators.

Getting Started

SimpleXML is best suited to operations such as reading data from an XML file, Updating data in an already existing file.

SimpleXML works by reading in the entire XML file at once and converting it into a PHP object containing all the elements of that XML file chained together in the same way. Once the file has been loaded, data can be simply pulled out by traversing the object tree.

Once the XML file has been read in the memory**:**

- Properties denote element iterators
- Numeric indices denote elements
- Non-numeric indices denote attributes
- String conversion allows access to TEXT data

Examples in this sub topic require an XML document to work with.

An XML document can either be stored in a PHP memory variable of type string inside a **.**php file **OR** can be an individual file with **.**xml file extension. Both approaches are demonstrated in this material.

Since the examples to follow, will require an XML document to work with, the XML document, held inside a memory variable is created and saved in a PHP file named books**.**php. This file can then be included in the examples that require it.

Creating A .php File Holding XML Document

Using an ASCII editor, create a file named **books.php** with the following content**:**

```
<?php
$xmlstr = <<<XML
<?xml version='1.0' standalone='yes'?>
<books>
    <book>
        <title>Database Concepts And Systems B.Sc. (I.T.)</title>
        <isbn>81-7366-795-0</isbn>
        <category>Technical</category>
        <publisher code="SPD">Shroff Publishers Distributors Pvt.
                                Ltd</publisher>
        <authors>
            <name>Ivan Bayross</name>
        </authors>
        <cost>
            <rupees>200</rupees>
            <dollar>4</dollar>
        </cost>
        <rating type="poor">1</rating>
        <covers>
            <topic>Oracle</topic>
        </covers>
    </book>
    <book>
        <title>Application Development Using PHP With Oracle On Linux</title>
        <isbn>81-7656-282-3</isbn>
        <category>Technical</category>
        <publisher code="SPD">Shroff Publishers Distributors Pvt.
                                Ltd</publisher>
        <authors>
            <name>Sharanam Shah</name>
            <name>Ivan Bayross</name>
        </authors>
        <cost>
            <rupees>550</rupees>
            <dollar>11</dollar>
        </cost>
        <rating type="excellent">4</rating>
        <covers>
            <topic>PHP</topic>
            <topic>Oracle</topic>
        </covers>
    </book>
</books>
XML;
?>
```

In the above code spec the technique

```
<<<XML
    ...
    ...
    ...
    XML;
```

is used to identify the scope of the XML document.

This document is stored in a memory variable named **$xmlstr**.

```
<?php
    $xmlstr = <<<XML
    ...
    ...
    ...
    XML;
?>
```

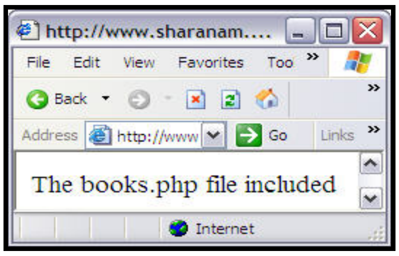
Once books.php is created and saved to HDD it can simply be put to use by including it within any PHP file that requires it to work with.

Including A .php File That Holds A XML Document

Any file that needs to work with books.php (that holds the XML Document in a memory variable) simply includes this file within itself using the following code:

```
<?php
/* Including books.php file */
    include 'books.php';
/* Intimating on successful file inclusion */
    echo "The books.php file included";
?>
```

Output:



The books.php file included

Diagram 22.1: books.php file included

Extracting Data From An XML Document

The simplicity of SimpleXML appears most clearly when one extracts a string or a number from within a XML document.

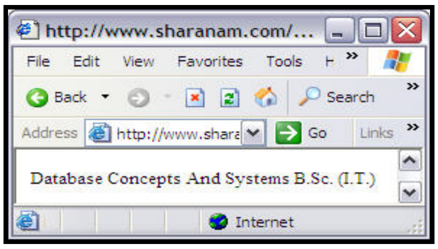
Using any ASCII editor, type in the following code in a file named view.php:

```
<?php
/* Including books.php file */
    include 'books.php';
```

```
/* Defining string variable xmlstr which is declared in
books.php file and holds the XML Document */
    $xml = simplexml_load_string($xmlstr);

/* Extracting field Title from books.php file */
    echo $xml->book[0]->title;
```

```
?>
```

Output:


http://www.sharanam.com/...
File Edit View Favorites Tools H
Back Search
Address http://www.shara Go Links
Internet

Database Concepts And Systems B.Sc. (I.T.)

Diagram 22.2.1: Viewing title of books

In the above codespec,

The XML Document is **loaded into MEMORY** from within the string type, memory variable **$xmlstr**.

This is done using the **simplexml_load_string()** function.

Syntax:

```
object simplexml_load_string(string <Data> [, string <Classname>
                                            [, int <Options>]] )
```

This function accepts a well-formed **xml** document in the form of a string and returns an object of class **SimpleXMLElement** with properties bound to the data held within the xml document. If errors are encountered while loading the XML document, the function returns FALSE.

The optional parameter **Classname** can be used to make **simplexml_load_string()** return an object of the specified class.

PHP 5.1.0 and **Libxml 2.6.0** onwards, the **Options** parameter can be used to specify additional **Libxml** parameters.

In the above code spec**:**

```
$xml = simplexml_load_string($xmlstr);
```

will **read** the contents held in the string named **$xmlstr** into an object named **$xml**.

Once the object is loaded with the XML document's data, it can traverse through to extract any data using its built in properties. To extract the first book's title, the object **$xml** is used, its **book** array is traversed to its first (i**.**e**.** zeroth) **book** element and the **data contents** of the child element **title** is extracted and displayed. This follows the hierarchy in which data is stored in the XML Document.

```
echo $xml->book[0]->title;
```

Here, the **book**[0] denotes the **first element** of an array. This is required as the XML document holds data in following hierarchy.

BOOKS - {The Opening Tag}
    BOOK - {The Opening Tag Of the First Book} - The Array Index therefore will be **0**
        TITLE
    BOOK - {The Closing Tag Of the First Book}

    BOOK - {The Opening Tag Of the Second Book} - The Array Index therefore will be **1**
        TITLE
    BOOK - {The Closing Tag Of the Second Book}
BOOKS - {The Closing Tag}

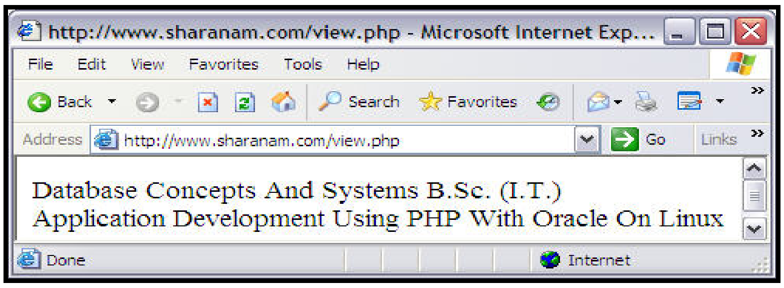
HINT

When multiple instances of an element exist as children of a single parent element, normal iteration techniques apply.

Example: (Using A FOREACH Loop)
The earlier code spec is modified to apply iteration via a **for** loop as:

```
<?php
/* Including books.php file */
    include 'books.php';
/* Defining string variable xmlstr which is declared in
books.php file and holds the XML Document */
    $xml = simplexml_load_string($xmlstr);
/* Extracting multiple Titles from books.php file */
    foreach ($xml->book as $book)
    {
        echo $book->title . '<BR />';
    }
?>
```

Output:


http://www.sharanam.com/view.php - Microsoft Internet Exp...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.sharanam.com/view.php Go Links
Done Internet

```
Database Concepts And Systems B.Sc. (I.T.)
Application Development Using PHP With Oracle On Linux
```

Diagram 22.2.2: Viewing multiple title of books

In the above code spec, the iteration technique applied is:

```
foreach ($xml->book as $book)
{
    ...
}
```

Here, an alias named **$book** is created as:

```
$xml->book as $book
```

The above code spec will simply create an alias named **$book** which will point to every book (CHILD NODE) while iterating through a list of BOOKS (i.e. the ROOT NODE of the XML document).

```
foreach ($xml->book as $book)
{
    echo $book->title . '<BR />';
}
```

Now since there are two entries (Refer to the books.php file which holds the XML document) for the BOOK (CHILD NODE) under the BOOKS (ROOT NODE) in the XML Document, the **for** loop will execute twice. Through each iteration the title of a book is displayed as:

```
echo $book->title . '<BR />';
```

Extracting Data From Attributes

SimpleXML can also access element attributes. Accessing attributes of an element is as simple as accessing the elements of an array.

The code spec below uses an HTML Table to hold the data extracted from elements and their attributes. This is done to make the output look elegant.

The following code spec demonstrates extracting data from an attribute called ***Type*** belonging to the **Rating** element (Refer to the books.php file which holds the XML document):

```
<?php
/* Including books.php file */
    include 'books.php';

/*  Defining  string  variable  xmlstr  which  is  declared  in
     books.php file and holds the XML Document */
    $xml = simplexml_load_string($xmlstr);
```

```
?>
<TABLE BORDER=1 Align="center" BORDERCOLOR="FF6EC7">
    <TR>
        <TH>Title</TH>
        <TH>Ratings</TH>
    <TR>
<?php
```

```
    foreach ($xml->book as $book)
    {
```

```
?>
    <TR>
        <TD VALIGN="top">
            <?php
```

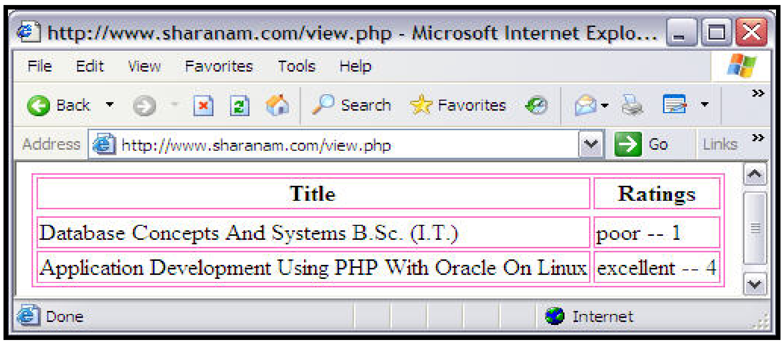
```
            /* Extracting Titles from books.php file */
                echo $book->title . '<BR />';
```

```
            ?>
        </TD>
        <TD VALIGN="top">
            <?php
            / Accessing attributes of ratings from books.php file
            */
                echo $book->rating['type'] . " -- ". $book->rating;
            ?>
        </TD>
    </TR>
<?php
```

```
    }
?>
```

```
</TABLE>
```

Output:



Title	Ratings
Database Concepts And Systems B.Sc. (I.T.)	poor -- 1
Application Development Using PHP With Oracle On Linux	excellent -- 4

Diagram 22.3: Viewing ratings for a title of books

In the above code spec, the FOREACH loop now has an additional line of code which extracts data held by the ***Type*** attribute of the element **Rating**. The extracted data is held in a table to make the output look elegant.

The **FOREACH** loop now looks like**:**

```
<?php
   foreach ($xml->book as $book)
   {
?>
   <TR>
      <TD VALIGN="top">
         <?php
         /* Extracting Titles from books.php file */
            echo $book->title . '<BR />';
         ?>
      </TD>

      <TD VALIGN="top">
         <?php
         /* Accessing attributes of field rating from books.php
         file */
            echo $book->rating['type'] . " -- ". $book->rating;
         ?>
      </TD>
   </TR>
```

In the books**.**php each ratings has an attribute named ***type*:**

```
<rating type="poor">1</rating>
```

The statement,

```
echo $book->rating['type'] . " -- ". $book->rating;
```

extracts the attribute (rating**[**'type'**]**) data as well as the element ($book->rating) data.

Since the *Type* attribute belongs to the Rating element it is accessed just like accessing an array element i**.**e**.** $book->rating**[**'type'**]**. Here ***type*** is the **array element** and **rating** is the **array**.

Further to this, the above statement also extracts the data held by the Rating element and concatenates the same (**.** " -- "**.** $book->rating) to the data extracted from the attribute.

Comparing Elements And / Or Attributes With The Criteria

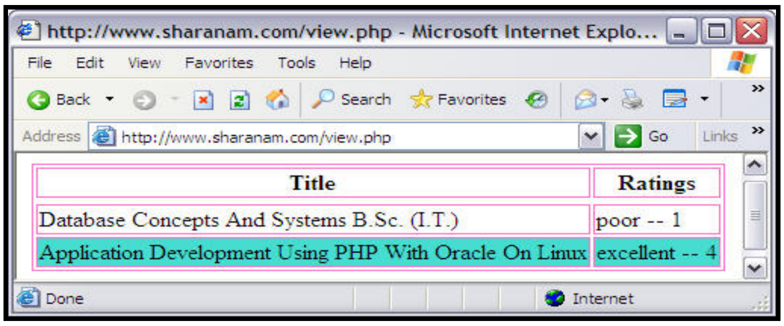
SimpleXML also allows comparing an (element or attribute)'s data with a string.

To do this, the (element or attribute)'s data that is extracted, has to be typecast to a string using **(string)**. This is because by default, PHP treats an element/attribute as an object, which will not allow comparisons.

The following code spec demonstrates comparisons**:**

```
<?php
/* Including books.php file */
    include 'books.php';
/* Defining string variable xmlstr which is declared in
books.php file and holds the XML Document */
    $xml = simplexml_load_string($xmlstr);
?>
<TABLE BORDER=1 Align="center" BORDERCOLOR="FF6EC7">
    <TR>
        <TH>Title</TH>
        <TH>Ratings</TH>
    <TR>
<?php
    foreach ($xml->book as $book)
    {
    /* Comparing the element name and giving the border color */
        if ((string)$book->authors->name[0] == "Sharanam Shah")
        {
            $bgcolor = "Turquoise";
        }
?>
    <TR BGCOLOR="<?php echo $bgcolor; ?>">
        <TD VALIGN="top">
            <?php
            /* Extracting Titles from books.php file */
                echo $book->title . '<BR />';
            ?>
        </TD>
        <TD VALIGN="top">
            <?php
            /* Accessing attributes of field rating from books.php
            file */
                echo $book->rating['type'] . " -- ". $book->rating;
            ?>
        </TD>
    </TR>
<?php
    }
?>
</TABLE>
```

Output:



http://www.sharanam.com/view.php - Microsoft Internet Explo...

Title	Ratings
Database Concepts And Systems B.Sc. (I.T.)	poor -- 1
Application Development Using PHP With Oracle On Linux	excellent -- 4

Diagram 22.4: Viewing multiple title of books

In the above code spec, the color **turquoise** is assigned to the string **$bgcolor**, if the first element **NAME** under the **AUTHORS** element holds the string **'Sharanam Shah':**

```
if ((string)$book->authors->name[0] == "Sharanam Shah")
{
    $bgcolor = "Turquoise";
}
```

Here, the value extracted by the $book->authors->name**[**0**]** statement is typecast to **STRING** (**(string)**$book->authors->name**[**0**]**). It is only after the data that is extracted is typecast, will the comparison be successful.

The ***BGCOLOR*** attribute of the **<TR>** HTML tag is assigned the value held by the memory variable **$bgcolor** as follows**:**

```
<TR BGCOLOR="<?php echo $bgcolor; ?>">
```

Using Xpath

There are many common operations on XML documents such as**:**

- Accessing all descendants of a given node
- Searching a document and finding a tag and a value that both match a given condition

These operations are a pain to write by hand and *desperately need simplification*.

As a solution to this problem, SimpleXML does not reinvent the wheel, but instead provides the **xpath()** method, which allows performing **W3C standard xpath queries** on an XML document. A problem like getting all descendants of a given node turns into a highly optimized xpath query.

REMINDER

It is recommended that anyone serious about processing XML should learn to use **Xpath** syntax, which is as important to XML as Regular Expressions are to plain text.

SimpleXML includes a built in support for Xpath support

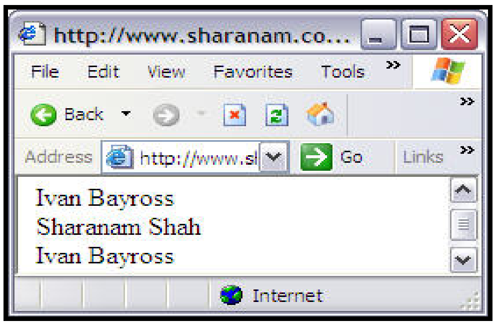
Syntax:

```
<VariableName>->xpath('//<Element>') as <VariableName>
```

Create a file using any ASCII editor with the following code spec**:**

```
<?php
    include 'books.php';
    $xml = simplexml_load_string($xmlstr);

    foreach ($xml->xpath('//book/authors') as $AuthorsNode)
    {
        foreach ($AuthorsNode as $AuthorNames)
        {
            echo $AuthorNames . '<BR />';
        }
    }
?>
```

Output:



```
Ivan Bayross
Sharanam Shah
Ivan Bayross
```

Diagram 22.5.1: Viewing Authors For Each book title

In the above code spec, the following command**:**

```
foreach ($xml->xpath('//book/authors') as $AuthorsNode)
```

indicate that the traversing will begin from inside the Authors Node.

Currently the ROOT NODE i.e. BOOKS holds BOOK as the CHILD NODE which in turn holds AUTHORS as a CHILD NODE. The Authors CHILD NODE holds the NAME element for which there can be multiple entries.

```
BOOKS - {The Opening Tag}
    BOOK  - {The Opening Tag Of the First Book} - The Array Index therefore will be 0
        AUTHORS - {The Opening Tag of the Child Node}
                NAME - {The First ELEMENT}
                NAME - {The Second ELEMENT}
                ...
        AUTHORS - {The Closing Tag of the Child Node}
    BOOK  - {The Closing Tag Of the First Book}

    BOOK  - {The Opening Tag Of the Second Book} - The Array Index therefore will be 1
        AUTHORS - {The Opening Tag of the Child Node}
                NAME - {The First ELEMENT}
                NAME - {The Second ELEMENT}
                ...
        AUTHORS - {The Closing Tag of the Child Node}
    BOOK  - {The Closing Tag Of the Second Book}
BOOKS - {The Closing Tag}
```

Then the following command:

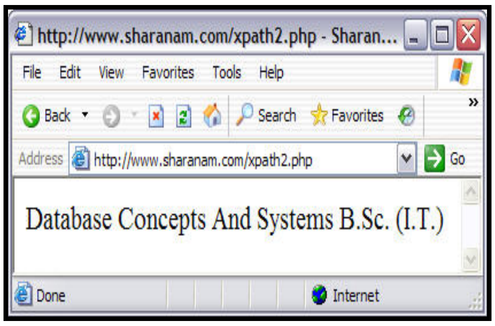
```
foreach ($AuthorsNode as $AuthorNames)
{
    echo $AuthorsName . '<BR />';
}
```

displays every individual author's name (i.e. Books→Book→Authors→Name) for every individual book.

Example:
Using any ASCII editor, type in the following code spec:

```
<?php
    include 'books.php';
    $xml = simplexml_load_string($xmlstr);

    $FilteredData = $xml->xpath('//book[contains(covers/topic, "Oracle")]');
    foreach ($FilteredData as $MyData)
    {
        echo $MyData->title . '<BR />';
    }
?>
```

Output:


http://www.sharanam.com/xpath2.php - Sharan...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.sharanam.com/xpath2.php Go
Done Internet

Database Concepts And Systems B.Sc. (I.T.)

Diagram 22.5.2: Viewing book titles that covers Oracle as a topic

In the above code spec, the following command**:**

```
$FilteredData = $xml->xpath('//book[contains(covers/topic, "Oracle")]');
```

indicate that

- Traversing will begin from the book node itself
- Array **$FilteredData** will hold only the data that matches the following criteria**:**

  Book->Covers->Topic contains a string **Oracle**.

This is done using the following search pattern**:**

```
[contains(covers/topic, "Oracle")]
```

Here, the **contains()** Xpath function is used to filter the data held under covers/topic that contains the string Oracle.

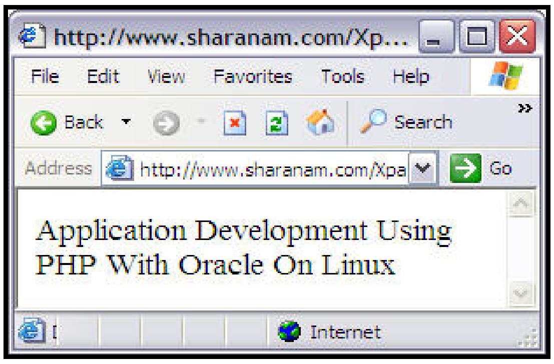
Finally, a **foreach** loop is used to extract and display data from the **$FilteredData** array as**:**

```
foreach ($FilteredData as $MyData)
{
    echo $MyData->title . '<BR />';
}
```

Example:
Using any ASCII editor, type in the following code spec**:**

```
<?php
    include 'books.php';
    $xml = simplexml_load_string($xmlstr);

    $FilteredData = $xml->xpath('//book/title[starts-with(., "Application")]');
    foreach ($FilteredData as $MyData)
    {
        echo $MyData . '<BR />';
    }
?>
```

Output:


http://www.sharanam.com/Xp...
File Edit View Favorites Tools Help
Back Search
Address http://www.sharanam.com/Xpa Go
Internet

Application Development Using
PHP With Oracle On Linux

Diagram 22.5.3: Viewing book titles that begin with the word **Application**

In the above code spec, the following command**:**

```
$FilteredData = $xml->xpath('//book/title[starts-with(., "Application")]');
```

indicate that

- Traversing will begin from inside the title node
- Array **$FilteredData** will hold only the data that matches the following criteria**:**

  Book->Title begins with the string **Application**.

This is done using the following search pattern**:**

```
[starts-with(., "Application")]
```

Here, the **starts-with()** Xpath function is used to filter the data held under book/title that begins with the string Application. The **.** as the first parameter in the **starts-with** function indicate the current element which in this case is title **(xpath('//**book**/**title**).**

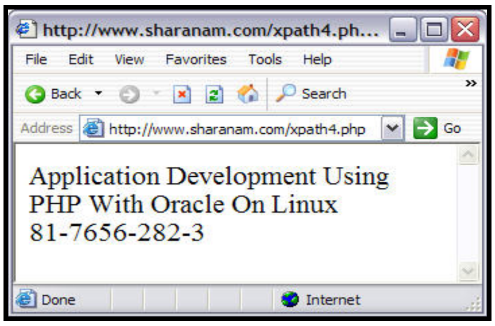
Finally, a **foreach** loop is used to extract and display data from the **$FilteredData** array as**:**

```
foreach ($FilteredData as $MyData)
{
    echo $MyData . '<BR />';
}
```

Example:
Using any ASCII editor, type in the following code spec:

```
<?php
    include 'books.php';
    $xml = simplexml_load_string($xmlstr);

    $FilteredData = $xml->xpath('//book[starts-with(title, "Application")]');
    foreach ($FilteredData as $MyData)
    {
        echo $MyData->title . '<BR />';
        echo $MyData->isbn . '<BR />';
    }
?>
```

Output:



http://www.sharanam.com/xpath4.ph...
File Edit View Favorites Tools Help
Back Search
Address http://www.sharanam.com/xpath4.php Go
Done Internet

```
Application Development Using
PHP With Oracle On Linux
81-7656-282-3
```

Diagram 22.5.4: Viewing book titles that begin with the word Application

In the above code spec, the following command:

```
$FilteredData = $xml->xpath('//book[starts-with(title, "Application")]');
```

indicate that

- Traversing will begin from inside the book node itself
- Array **$FilteredData** will hold only the data that matches the following criteria:

  Book->Title begins with the string **Application**.

This is done using the following search pattern:

```
[starts-with(title, "Application")]
```

Here, the **starts-with()** Xpath function is used to filter the data held under book/title that begins with the string Application. The **first parameter** here as against the previous example (where it was **. -** to indicate the current element) is **title**.

This method will give the same results as the previous example except that here the **traversing will start from the book node and not from inside the book/title**. This will thus give access to all element data.

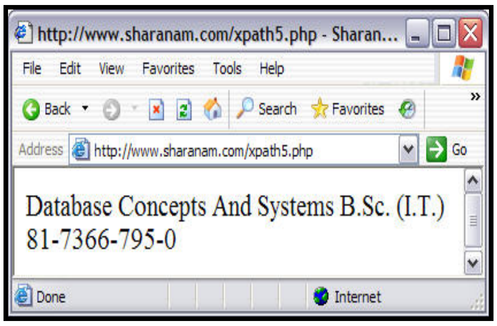
Finally, a **foreach** loop is used to extract and display data from the **$FilteredData** array as**:**

```
foreach ($FilteredData as $MyData)
{
    echo $MyData->title . '<BR />';
    echo $MyData->isbn . '<BR />';
}
```

Example:
Using any ASCII editor, type in the following code spec**:**

```
<?php
    include 'books.php';
    $xml = simplexml_load_string($xmlstr);

    $FilteredData = $xml->xpath('//book[contains(rating/@type, "poor")]');
    foreach ($FilteredData as $MyData)
    {
        echo $MyData->title . '<BR />';
        echo $MyData->isbn . '<BR />';
    }
?>
```

Output:


http://www.sharanam.com/xpath5.php - Sharan...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.sharanam.com/xpath5.php Go
Done Internet

```
Database Concepts And Systems B.Sc. (I.T.)
81-7366-795-0
```

Diagram 22.5.5: Viewing book titles and ISBN that have a poor rating

In the above code spec, the following command**:**

```
$FilteredData = $xml->xpath('//book[contains(rating/@type, "poor")]');
```

indicate that

- Traversing will begin from inside the book node itself

- Array **$FilteredData** will hold only the data that matches the following criteria**:**

  type attribute of **Book->Rating** holds **poor**.

This is done using the following search pattern**:**

```
[contains(rating/@type, "poor")]
```

Here, the **contains()** Xpath function is used to filter the data held in an attribute named ***type*** under book/rating that holds the string poor. The first parameter **rating/@type** specifies that rating has an attribute (indicated by **@**) named ***type***.

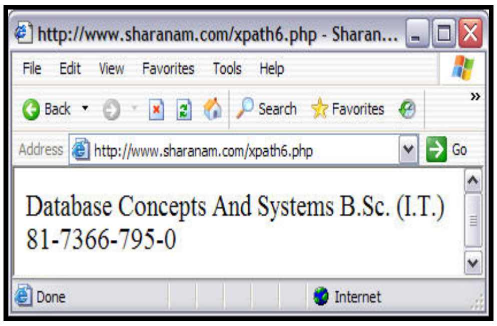
Finally, a **foreach** loop is used to extract and display data from the **$FilteredData** array as**:**

```
foreach ($FilteredData as $MyData)
{
    echo $MyData->title . '<BR />';
    echo $MyData->isbn . '<BR />';
}
```

Example:
Using any ASCII editor, type in the following code spec**:**

```
<?php
    include 'books.php';
    $xml = simplexml_load_string($xmlstr);

    $FilteredData = $xml->xpath('//book[rating/@type = "poor"]');
    foreach ($FilteredData as $MyData)
    {
        echo $MyData->title . '<BR />';
        echo $MyData->isbn . '<BR />';
    }
?>
```

Output:


http://www.sharanam.com/xpath6.php - Sharan...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.sharanam.com/xpath6.php Go
Done
Internet

```
Database Concepts And Systems B.Sc. (I.T.)
81-7366-795-0
```

Diagram 22.5.6: Viewing book titles and ISBN that have a poor rating

In the above code spec, the following command:

```
$FilteredData = $xml->xpath('//book[rating/@type = "poor"]');
```

indicate that

- Traversing will begin from inside the book node itself
- Array **$FilteredData** will hold only the data that matches the following criteria:

  type attribute of **Book->Rating** holds **poor**.

This is done using the following search pattern:

```
[rating/@type = "poor"]
```

Here, the **operator =** is used to filter the data held in an attribute named ***type*** under book/rating that holds the string poor. The first parameter **rating/@type** specifies that rating has an attribute (indicated by @) named ***type***.

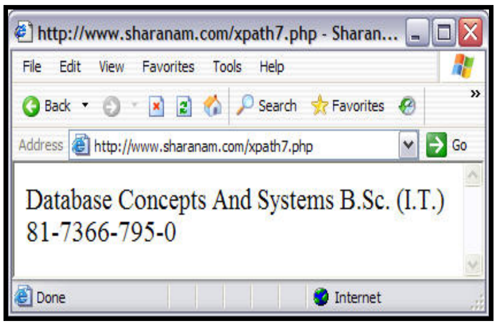
Finally, a **foreach** loop is used to extract and display data from the **$FilteredData** array as:

```
foreach ($FilteredData as $MyData)
{
    echo $MyData->title . '<BR />';
    echo $MyData->isbn . '<BR />';
}
```

Example:
Using any ASCII editor, type in the following code spec:

```
<?php
    include 'books.php';
    $xml = simplexml_load_string($xmlstr);

    $FilteredData = $xml->xpath('//book[rating/@type = "poor" and
                                cost/rupees > 185]');
    foreach ($FilteredData as $MyData)
    {
        echo $MyData->title . '<BR />';
        echo $MyData->isbn . '<BR />';
    }
?>
```

Output:


http://www.sharanam.com/xpath7.php - Sharan...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.sharanam.com/xpath7.php Go
Done
Internet

```
Database Concepts And Systems B.Sc. (I.T.)
81-7366-795-0
```

Diagram 22.5.7: Viewing book titles and ISBN that have a poor rating and cost more than Rs. 185

In the above code spec, the following command**:**

```
$FilteredData = $xml->xpath('//book[rating/@type = "poor" and cost/rupees > 185]');
```

indicate that

- Traversing will begin from inside the book node itself
- Array **$FilteredData** will hold only the data that matches the following criteria**:**

  type attribute of **Book->Rating** holds **poor**

  rupees element under the **cost** node holds a value more than **185**

This is done using the following search pattern**:**

```
[rating/@type = "poor" and cost/rupees > 185]
```

Here,

The **operator =** is used to filter the data held in an attribute named ***type*** under book/rating that holds the string **poor**

The **operator >** is used to filter the data held in an element named **rupees** under **cost** node is more than **185**

The **operator and** is used to make both the above condition mandatory

Finally, a **foreach** loop is used to extract and display data from the **$FilteredData** array as**:**

```
foreach ($FilteredData as $MyData)
{
    echo $MyData->title . '<BR />';
    echo $MyData->isbn . '<BR />';
}
```

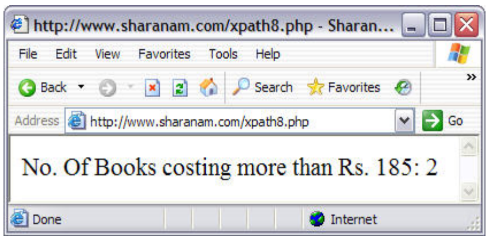
Similarly, following other operators can be used and applied to an xml file**:**

<	Less Than
>	Greater Than
>=	Greater than or Equal to
<=	Less than or Equal to
=	Equals
!=	Not Equal

Example:
Using any ASCII editor, type in the following code spec**:**

```
<?php
    include 'books.php';
    $xml = simplexml_load_string($xmlstr);

    $NoOfBooks = count($xml->xpath('//book[cost/rupees > 185]'));
    echo "No. Of Books costing more than Rs. 185: " . $NoOfBooks;
?>
```

Output:


http://www.sharanam.com/xpath8.php - Sharan...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.sharanam.com/xpath8.php Go
Done Internet

```
No. Of Books costing more than Rs. 185: 2
```

Diagram 22.5.8: Counting the number of books that cost more than 185

In the above code spec, the following command**:**

```
$NoOfBooks = count($xml->xpath('//book[cost/rupees > 185]'));
```

indicate that

- Traversing will begin from inside the book node itself
- Variable **$NoOfBooks** will hold a count of total number of books on the following criteria**:**

  rupees element under the **cost** node holds a value more than **185**

This is done using the following search pattern**:**

```
[cost/rupees > 185]
```

Here,

The **operator >** is used to filter the data held in an element named **rupees** under **cost** node is more than **185**

The count function is applied to the search pattern

```
count($xml->xpath('//book[cost/rupees > 185]'))
```

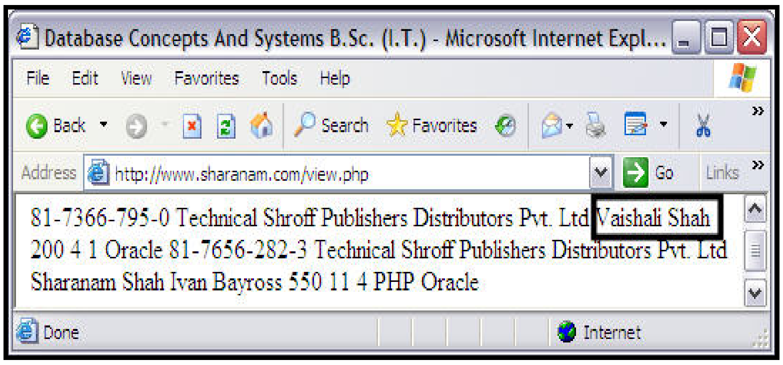
Setting Values

Using SimpleXML data can be modified or set. The object allows manipulation of all of its elements.

Create a file with the following code spec**:**

```
<?php
    include 'books.php';
    $xml = simplexml_load_string($xmlstr);

/* Modifying the Author Name of the First Book */
    $xml->book[0]->authors->name = 'Vaishali Shah';
    echo $xml->asXML();
?>
```

Output:


Database Concepts And Systems B.Sc. (I.T.) - Microsoft Internet Expl...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.sharanam.com/view.php Go Links
Done Internet

```
81-7366-795-0 Technical Shroff Publishers Distributors Pvt. Ltd Vaishali Shah
200 4 1 Oracle 81-7656-282-3 Technical Shroff Publishers Distributors Pvt. Ltd
Sharanam Shah Ivan Bayross 550 11 4 PHP Oracle
```

Diagram 22.6.1: Changing the name of the author

In the above code spec, the**:**

```
$xml->book[0]->authors->name = 'Vaishali Shah';
```

changes the author name (authors->name) of the First Book (book**[**0**]**) to 'Vaishali Shah'.

The following code spec will display the books XML document with the changes applied**:**

```
echo $xml->asXML();
```

The function **asXML** returns a well-formed XML string based on **SimpleXML** element.

Syntax:

```
mixed <SimpleXMLElement>->asXML([string <Filename>] )
```

The **asXML** method formats the parent object's data in XML version 1.0. The parameter **Filename** if specified, the function writes the data to the file rather than returning it. If the filename is not specified, this function returns a string on success and FALSE on error.

If the parameter is specified, it returns TRUE if the file was written successfully and FALSE otherwise.

Create a file with the following code spec:

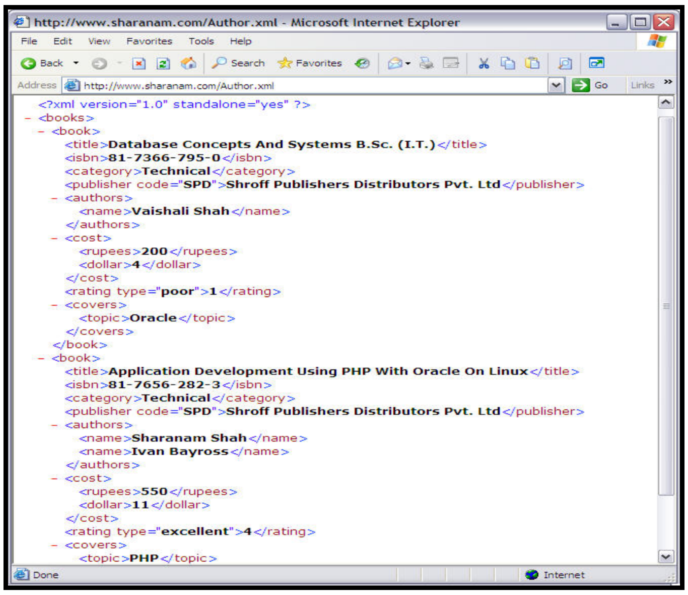
```
<?php
    include 'books.php';
    $xml = simplexml_load_string($xmlstr);

/* Modifying the Author Name of the First Book */
    $xml->book[0]->authors->name = 'Vaishali Shah';
    echo $xml->asXML(Author.xml);
?>
```

The following code spec will display the books XML document with the changes applied and will also create an XML file **Author.xml:**

```
echo $xml->asXML(Author.xml);
```

View the **Author.xml** file in any web browser such I.E. or Mozilla. Refer to diagram 22.6.2

Output:


http://www.sharanam.com/Author.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.sharanam.com/Author.xml
Go Links
Done
Internet

```
  <?xml version="1.0" standalone="yes" ?>
- <books>
  - <book>
      <title>Database Concepts And Systems B.Sc. (I.T.)</title>
      <isbn>81-7366-795-0</isbn>
      <category>Technical</category>
      <publisher code="SPD">Shroff Publishers Distributors Pvt. Ltd</publisher>
    - <authors>
        <name>Vaishali Shah</name>
      </authors>
    - <cost>
        <rupees>200</rupees>
        <dollar>4</dollar>
      </cost>
      <rating type="poor">1</rating>
    - <covers>
        <topic>Oracle</topic>
      </covers>
    </book>
  - <book>
      <title>Application Development Using PHP With Oracle On Linux</title>
      <isbn>81-7656-282-3</isbn>
      <category>Technical</category>
      <publisher code="SPD">Shroff Publishers Distributors Pvt. Ltd</publisher>
    - <authors>
        <name>Sharanam Shah</name>
        <name>Ivan Bayross</name>
      </authors>
    - <cost>
        <rupees>550</rupees>
        <dollar>11</dollar>
      </cost>
      <rating type="excellent">4</rating>
    - <covers>
        <topic>PHP</topic>
```

Diagram 22.6.2: Changing the name of the author

Reading An External XML Document

To read an XML file into an object, **simplexml_load_file()** is used.

Syntax:

```
object simplexml_load_file(string <Filename> [, string <Classname>
                            [, int <Options>]] )
```

This function converts the XML document in the file specified by Filename to an object of class **SimpleXMLElement**. If an error is encountered while doing this, the function returns FALSE.

The optional **Classname** parameter can be used to make the **simplexml_load_file()** function return an object of the specified class.

PHP 5.1.0 and **Libxml 2.6.0** onwards, the **Options** parameter can also be used to specify additional **Libxml** parameters.

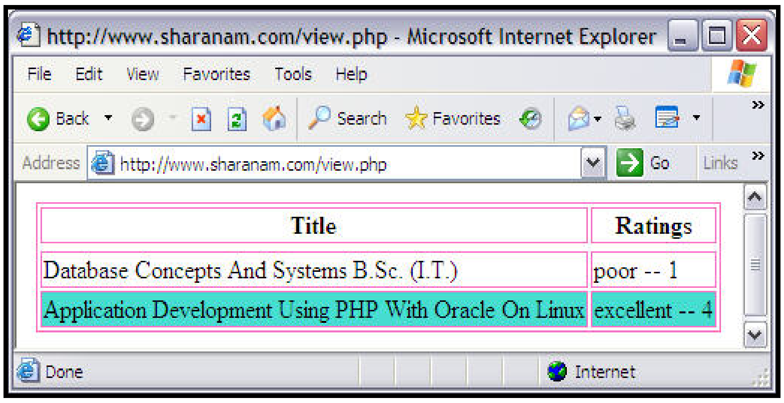
Example:
To demonstrate using an external XML file named books.xml, the following changes are made in the example explained earlier.

```
<?php
    if (file_exists('books.xml'))
      $xml = simplexml_load_file('books.xml');
?>
<TABLE BORDER=1 Align="center" BORDERCOLOR="FF6EC7">
    <TR>
        <TH>Title</TH>
        <TH>Ratings</TH>
    <TR>

<?php
    foreach ($xml->book as $book)
    {
    /* Comparing the element name and giving the border color */
        if ((string)$book->authors->name[0] == "Sharanam Shah")
            $bgcolor = "Turquoise";
?>
    <TR BGCOLOR="<?php echo $bgcolor; ?>">
        <TD VALIGN="top">
            <?php
            /* Extracting Titles from books.xml file */
                echo $book->title . '<BR />';
            ?>
        </TD>

        <TD VALIGN="top">
            <?php
            /* Accessing attributes of ratings from books.xml file
            */
                echo $book->rating['type'] . " -- ". $book->rating;
            ?>
        </TD>
    </TR>
<?php
    }
?>
</TABLE>
```

Create a file named books.xml with the following contents:

```
<?xml version='1.0' standalone='yes'?>
<books>
    <book>
        <title>Database Concepts And Systems B.Sc. (I.T.)</title>
        <isbn>81-7366-795-0</isbn>
        <category>Technical</category>
        <publisher code="SPD">Shroff Publishers Distributors Pvt.
                                  Ltd</publisher>
        <authors>
            <name>Ivan Bayross</name>
        </authors>
        <cost>
            <rupees>200</rupees>
            <dollar>4</dollar>
        </cost>
        <rating type="poor">1</rating>
        <covers>
            <topic>Oracle</topic>
        </covers>
    </book>

    <book>
        <title>Application Development Using PHP With Oracle On Linux</title>
        <isbn>81-7656-282-3</isbn>
        <category>Technical</category>
        <publisher    code="SPD">Shroff    Publishers    Distributors    Pvt.
Ltd</publisher>
        <authors>
            <name>Sharanam Shah</name>
            <name>Ivan Bayross</name>
        </authors>
        <cost>
            <rupees>550</rupees>
            <dollar>11</dollar>
        </cost>
        <rating type="excellent">4</rating>
        <covers>
            <topic>PHP</topic>
            <topic>Oracle</topic>
        </covers>
    </book>
</books>
```

Output:


http://www.sharanam.com/view.php - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.sharanam.com/view.php Go Links
Done Internet

Title	Ratings
Database Concepts And Systems B.Sc. (I.T.)	poor -- 1
Application Development Using PHP With Oracle On Linux	excellent -- 4

Diagram 22.7: Checking the existence of file and loading the file

In the above code spec the following command checks for the existence of the file books.xml.

If the books.xml file exist, then the **simplexml_load_file()** function will interpret the XML file contents to an object **$xml**.

```
if (file_exists('books.xml'))
   $xml = simplexml_load_file('books.xml');
```

Modifying Data In An XML File

Create an XML file with the following code spec:

```
<?xml version="1.0" encoding="UTF-8" ?>
<books>
    <book id="1" title="Database Concepts And Systems B.Sc. I.T.">
        <authors>
            <name>Ivan Bayross</name>
            <name>None</name>
        </authors>
        <rating type="poor">5</rating>
    </book>
    <book id="2" title="Application Development Using PHP With Oracle On
                                        Linux">
        <authors>
            <name>Sharanam Shah</name>
            <name>Ivan Bayross</name>
        </authors>
        <rating type="excel">4</rating>
    </book>
</books>
```

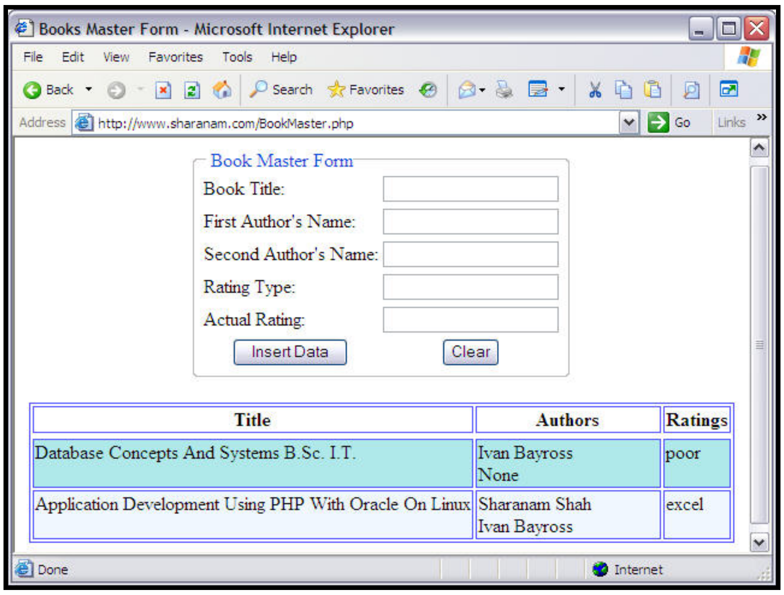
Create BookMaster.php file with the following code spec:

```
<HTML>
    <HEAD>
        <TITLE>Books Master Form </TITLE>
        <SCRIPT>
            function setEditMode(id, title, name1, name2, rating, ratingtype)
            {
                document.forms[0].hidBookID.value=id;
                document.forms[0].title.value=title;
                document.forms[0].authorsname1.value=name1;
                document.forms[0].authorsname2.value=name2;
                document.forms[0].ratingtype.value=ratingtype;
                document.forms[0].rating.value=rating;
            }
        </SCRIPT>
    </HEAD>
    <BODY>
        <FORM NAME="frmBooksMaster" ACTION="UpdateRecord.php"
                 METHOD="POST">
            <INPUT TYPE="HIDDEN" VALUE="" NAME="hidBookID">
            <TABLE ALIGN="center">
                <TR><TD>
                <FIELDSET>
                    <LEGEND>Book Master Form</LEGEND>
                    <TABLE ALIGN="center">
                        <TR>
                            <TD>Book Title:</TD>
                            <TD><INPUT TYPE="text" NAME="title"></TD>
                        </TR>
                        <TR>
                            <TD>First Author's Name:</TD>
                            <TD><INPUT TYPE="text"
                                        NAME="authorsname1"></TD>
                        </TR>
                        <TR>
                            <TD>Second Author's Name:</TD>
                            <TD><INPUT TYPE="text"
                                        NAME="authorsname2"></TD>
                        </TR>
                        <TR>
                            <TD>Rating Type:</TD>
                            <TD><INPUT TYPE="text"
                                        NAME="ratingtype"></TD>
                        </TR>
                        <TR>
                            <TD>Actual Rating:</TD>
                            <TD><INPUT TYPE="text" NAME="rating"></TD>
                        </TR>
                        <TR ALIGN="CENTER">
                            <TD><INPUT TYPE="submit" Value="Update Data">
                            </TD>
                            <TD><INPUT TYPE="reset" Value="Clear"></TD>
                        </TR>
                    </TABLE>
                </FIELDSET>
            </TD></TR>
        </TABLE>
        </FORM>
    </BODY>
</HTML>

<?php
    if (file_exists('books.xml'))
        $xml = simplexml_load_file('books.xml');
?>

<TABLE BORDER=1 ALIGN="center" BORDERCOLOR="4D4DFF">
    <TR>
        <TH>Title</TH>
        <TH>Authors</TH>
        <TH>Ratings</TH>
    <TR>
<?php
    foreach ($xml->book as $book)
    {
        if ((string)$book->authors->name[0] == "Ivan Bayross")
            $bgcolor = "ADEAEA";
        else
            $bgcolor = "F0F8FF";

?>

    <TR BGCOLOR="<?php echo $bgcolor; ?>"
        onMouseOver="this.bgColor='#EFEEC9';"
        onMouseOut="this.bgColor='<?php echo $bgcolor; ?>';"
        onMouseDown="setEditMode(<?php
            echo "'" . $book['id'] .
                "', '" . $book['title'] .
                "', '" . $book->authors->name[0] .
                "', '" . $book->authors->name[1] .
                "', '" . $book->rating .
                "', '" . $book->rating['type'] .
            "'" ?>);">
        <TD VALIGN="top">
            <?php
                echo $book['title'] . '<BR />';
            ?>
        </TD>
        <TD WIDTH=150 VALIGN="top">
            <?php
                foreach ($book->authors as $authors)
                {
                    foreach ($authors->name as $name)
                        echo $name . '<BR />';
            ?>
        </TD>
<?php
                }
?>
        <TD VALIGN="top">
            <?php
                foreach ($book->rating as $rating)
                {
                    echo $rating['type'];
            ?>
        </TD>
<?php
                }
?>
    </TR>
<?php
    }
?>
</TABLE>
```

Create another file named UpdateRecord.php with the following code spec:

```
<?php
    if (file_exists('books.xml'))
        $xml = simplexml_load_file('books.xml');

    foreach ($xml->book as $book)
    {
        if ((string)$book['id'] == $_POST['hidBookID'])
        {
            $book['title']=$_POST['title'];
            $book->authors->name[0] = $_POST['authorsname1'];
            $book->authors->name[1] = $_POST['authorsname2'];
            $book->rating = $_POST['rating'];
            $book->rating['type'] = $_POST['ratingtype'];
        }
    }

    $xml->asXML('books.xml');
    header("location:BookMaster.php");
?>
```

Run the file BookMaster.php in I.E. and the page appears as shown in diagram 22.8.1.


Books Master Form - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.sharanam.com/BookMaster.php
Go Links
Done
Internet

Book Master Form

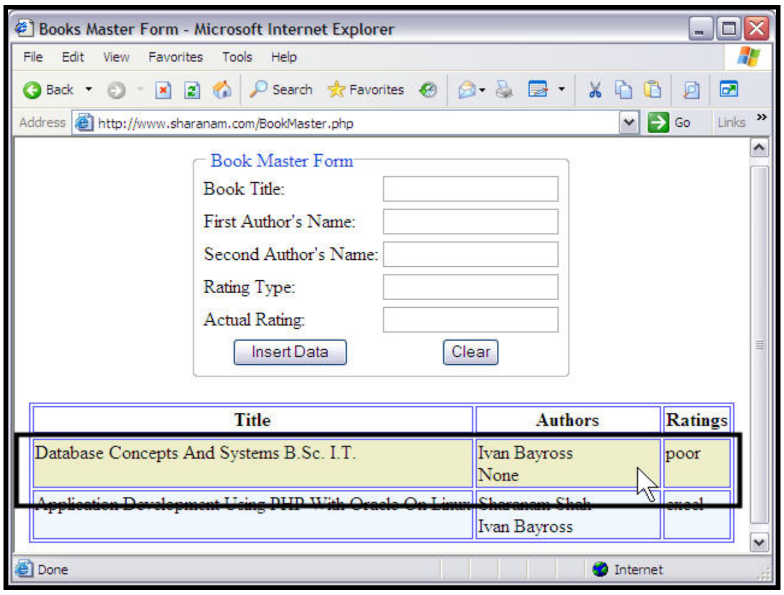
Book Title:	
First Author's Name:	
Second Author's Name:	
Rating Type:	
Actual Rating:	

Insert Data Clear

Title	Authors	Ratings
Database Concepts And Systems B.Sc. I.T.	Ivan Bayross None	poor
Application Development Using PHP With Oracle On Linux	Sharanam Shah Ivan Bayross	excel

Diagram 22.8.1: Book Master Form - BookMaster.php

Putting the mouse over the records changes the row color. Click on the record that requires modification. Refer to diagram 22.8.2.


Books Master Form - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.sharanam.com/BookMaster.php
Go Links
Done
Internet

Book Master Form

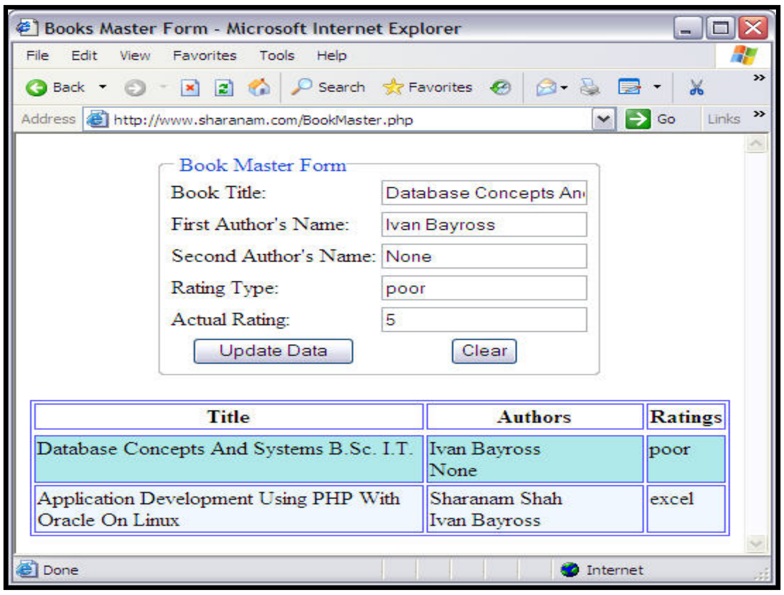
Book Title:	
First Author's Name:	
Second Author's Name:	
Rating Type:	
Actual Rating:	

Insert Data Clear

Title	Authors	Ratings
Database Concepts And Systems B.Sc. I.T.	Ivan Bayross None	poor
Application Development Using PHP With Oracle On Linux	Sharanam Shah Ivan Bayross	excel

Diagram 22.8.2: Mouse over in Book master Form

The records will be available in their respective fields for modification. Refer to diagram 22.8.3.



Books Master Form - Microsoft Internet Explorer

http://www.sharanam.com/BookMaster.php

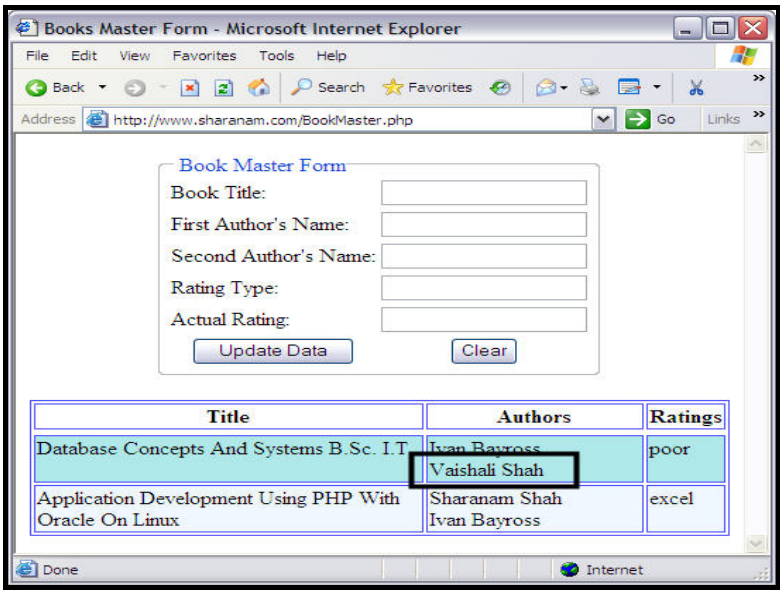
Book Master Form

Book Title:	Database Concepts An
First Author's Name:	Ivan Bayross
Second Author's Name:	None
Rating Type:	poor
Actual Rating:	5

Title	Authors	Ratings
Database Concepts And Systems B.Sc. I.T.	Ivan Bayross None	poor
Application Development Using PHP With Oracle On Linux	Sharanam Shah Ivan Bayross	excel

Diagram 22.8.3: Data available in the Book Master Form

Make the desired changes in the appropriate form elements. Click Update Data. This updates the data in the XML file and the page is re-rendered with updated data in the grid below as shown in diagram 22.8.4.



Books Master Form - Microsoft Internet Explorer

http://www.sharanam.com/BookMaster.php

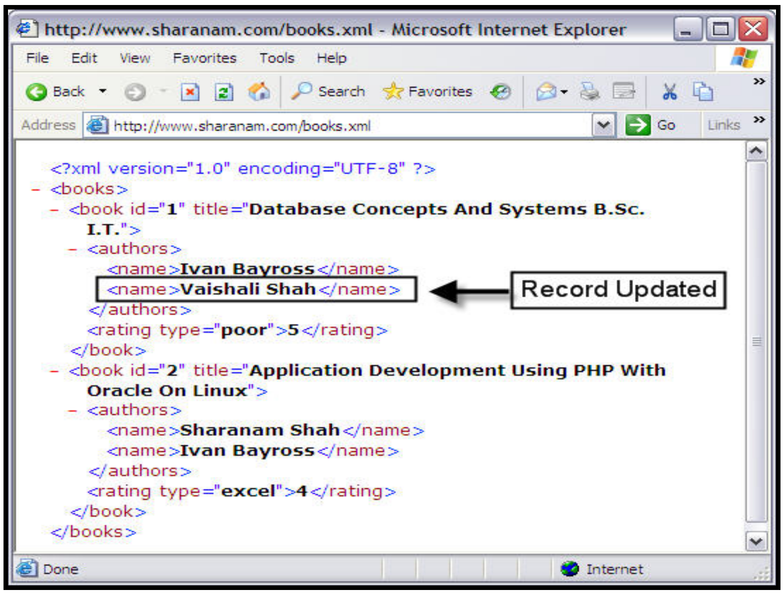
Book Master Form

Book Title:	
First Author's Name:	
Second Author's Name:	
Rating Type:	
Actual Rating:	

Title	Authors	Ratings
Database Concepts And Systems B.Sc. I.T.	Ivan Bayross Vaishali Shah	poor
Application Development Using PHP With Oracle On Linux	Sharanam Shah Ivan Bayross	excel

Diagram 22.8.4: The Record updated

The XML file i.e. **books.xml** can be verified for data modification. Refer to diagram 22.8.5.


http://www.sharanam.com/books.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.sharanam.com/books.xml Go Links
Record Updated
Done
Internet

```
<?xml version="1.0" encoding="UTF-8" ?>
- <books>
  - <book id="1" title="Database Concepts And Systems B.Sc.
      I.T.">
    - <authors>
        <name>Ivan Bayross</name>
        <name>Vaishali Shah</name>
      </authors>
      <rating type="poor">5</rating>
    </book>
  - <book id="2" title="Application Development Using PHP With
      Oracle On Linux">
    - <authors>
        <name>Sharanam Shah</name>
        <name>Ivan Bayross</name>
      </authors>
      <rating type="excel">4</rating>
    </book>
  </books>
```

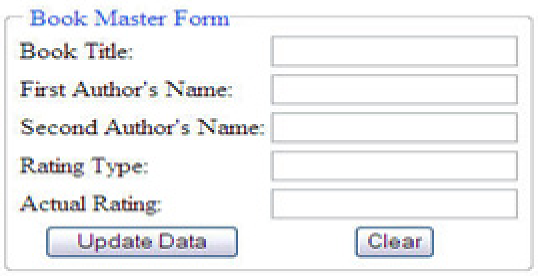
Diagram 22.8.5: Record updated in books.xml file

Clicking Clear while entering a record, simply clears or empties the text boxes.

Explanation:

❑ BookMaster.php file

The Form:



A function **setEditMode** is created, which accepts parameters based on the fields available in the **books.xml** file. It is called when a record is clicked via the GRID.

```
function setEditMode(id, title, name1, name2, rating, ratingtype)
{
    ...
}
```

A form is created which calls the file **UpdateRecord.php** as the **ACTION** attribute of the FORM points to it. The method it uses is **POST**. The code spec that does this is as follows**:**

```
<FORM NAME="frmBooksMaster" ACTION="UpdateRecord.php"
        METHOD="POST">
```

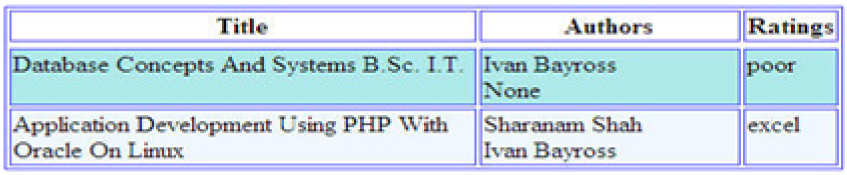
A hidden form element is created to hold the Book ID.

```
<INPUT TYPE="HIDDEN" VALUE="" NAME="hidBookID">
```

The buttons **Update Data** and **Clear** are created**:**

```
<INPUT TYPE="submit" Value="Update Data">
<INPUT TYPE="reset" Value="Clear"></TD>
```

The GRID:



Title	Authors	Ratings
Database Concepts And Systems B.Sc. I.T.	Ivan Bayross None	poor
Application Development Using PHP With Oracle On Linux	Sharanam Shah Ivan Bayross	excel

The following code spec will look out for the file **books.xml**. If it exists then it will load the file and spawn an object named **$xml** as**:**

```
if (file_exists('books.xml'))
    $xml = simplexml_load_file('books.xml');
```

The following code spec does the switching of row color based on the following condition**:**

```
if ((string)$book->authors->name[0] == "Ivan Bayross")
    $bgcolor = "ADEAEA";
else
    $bgcolor = "F0F8FF";
```

<u>This is done to make the GRID data more readable</u>.

The following code spec changes the color of the data row when mouse rolls over a record (Refer to diagram 22**.**8**.**2).

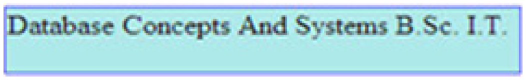
```
    <TR BGCOLOR="<?php echo $bgcolor; ?>"
        onMouseOver="this.bgColor='#EFEEC9';"
        onMouseOut="this.bgColor='<?php echo $bgcolor; ?>';"
        onMouseDown="setEditMode(<?php
            echo "'" . $book['id'] .
                "', '" . $book['title'] .
```

```
                    "', '" . $book->authors->name[0] .
                    "', '" . $book->authors->name[1] .
                    "', '" . $book->rating .
                    "', '" . $book->rating['type'] .
                "'" ?>);">
```

The above code spec also makes a provision to call the javascript function **setEditMode** by passing necessary parameters. This function will only be called on Mouse Down event of the data row (The row that holds a record).

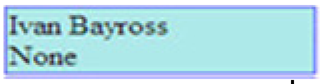
The following code spec displays the records.

```
<TD VALIGN="top">
    <?php
        echo $book['title'] . '<BR />';
    ?>
</TD>
```



Database Concepts And Systems B.Sc. I.T.

```
<TD WIDTH=150 VALIGN="top">
    <?php
        foreach ($book->authors as $authors)
        {
            foreach ($authors->name as $name)
                echo $name . '<BR />'b
    ?>
</TD>
<?php
        }
?>
```



Ivan Bayross
None

```
<TD VALIGN="top">
    <?php
        foreach ($book->rating as $rating)
        {
            echo $rating['type'];
    ?>
</TD>
```

poor

- UpdateRecord.php file

The following code spec will look out for the file **books.xml**. If it exists then it will load the xml file and spawn the xml object to hold the XML document**:**

```
if (file_exists('books.xml'))
    $xml = simplexml_load_file('books.xml');
```

The following code creates a **foreach** loop where **$book** will be an alias for **$xml->book:**

```
foreach ($xml->book as $book)
{
    ...
}
```

The following code held inside the **foreach** loop will check if the record selected for modification based on the value held by hidden form element **hidBookID** is the one under iteration**:**

```
if ((string)$book['id'] == $_POST['hidBookID'])
{
    ...
}
```

Once the record that requires modification based on the value held by hidden form element **hidBookID** is located using the **foreach** loop that traverses across all the records available in the XML document.

The following code spec does the actual modification in the **xml** object spawned earlier**:**

```
$book['title'] = $_POST['title'];
$book->authors->name[0] = $_POST['authorsname1'];
$book->authors->name[1] = $_POST['authorsname2'];
$book->rating = $_POST['rating'];
$book->rating['type'] = $_POST['ratingtype'];
```

The following **asXML()** function is then called after the **foreach** loop to update the xml file i**.**e**. books.xml:**

```
$xml->asXML('books.xml');
```

This is possible as the function is passed the xml filename as a parameter.

The following code spec makes sure that the BookMaster**.**php form is called immediately after the record is successfully updated in the xml file**:**

```
header("location:BookMaster.php");
```

Creating An XML Document Using MySQL And PHP

XML has emerged as a standard for data exchange. This means that data can be brought into an XML file from a database such as MySQL and plugged into another database such as Oracle.

The example that follows demonstrates extracting data from a MySQL DB table and writing it to a XML file.

To use the code spec, PHP and MySQL must be able to communicate with each other and MySQL's host name, username and password needs to be known.

Example:
Create a database named sharanam as follows:

```
CREATE DATABASE sharanam;
```

Output:

```
Query OK, 1 row affected (0.00 sec)
```

Create a table named books as follows:

```
CREATE TABLE books(
    title VARCHAR(60),
    isbn VARCHAR(15),
    category VARCHAR(10),
    publishercode VARCHAR(10),
    publisher VARCHAR(50),
    authorsname1 VARCHAR(25),
    authorsname2 VARCHAR(25),
    costrupees VARCHAR(10),
    costdollar VARCHAR(10),
    ratingtype VARCHAR(5),
    rating VARCHAR(5),
    covertopic1 VARCHAR(15),
    covertopic2 VARCHAR(15));
```

Output:

```
Query OK, 0 rows affected (0.09 sec)
```

Insert the following records into the table books:

```
INSERT INTO books VALUES("Database Concepts And Systems B.Sc. I.T.",
                         "81-7366-795-0", "Technical", "SPD", "Shroff
                         Publishers Distributors Pvt. Ltd", "Ivan Bayross",
                         "None", "200", "4", "poor", "1", "Oracle", "None");
INSERT INTO books VALUES("Using MySQL On Linux", "81-7656-951-8",
                         "Technical", "BpB", "BpB Publications", "Ivan
                         Bayross", "None", "150", "3", "good", "3", "MySQL",
                         "Linux");
INSERT INTO books VALUES("Application Development Using PHP With Oracle On
                         Linux", "81-7656-282-3", "Technical", "SPD",
                         "Shroff Publishers Distributors Pvt. Ltd", "Sharanam
                         Shah", "Ivan Bayross", "550", "11", "excellent", "4",
                         "PHP", "Oracle");
```

```
INSERT INTO books VALUES("Using PHP 5.1 For Prfessionals", "81-7656-939-9",
                         "Technical", "SPD", "Shroff Publishers Distributors
                         Pvt. Ltd", "Sharanam Shah", "Ivan Bayross", "750",
                         "15", "good", "3", "PHP", "Linux");
INSERT INTO books VALUES("Novel", "81-7656-100-4", "Non Technical", "BpB",
                         "BpB Publications", "Sharanam Shah", "None",
                         "100", "2", "avg", "2", "Politics", "None");
INSERT INTO books VALUES("Professional Oracle Projects On Linux",
                         "81-7656-920-8", "Technical", "BpB", "BpB
                         Publications", "Sharanam Shah", "None", "500",
                         "10", "good", "3", "Oracle", "Linux");
```

Output: (For every INSERT INTO statement)

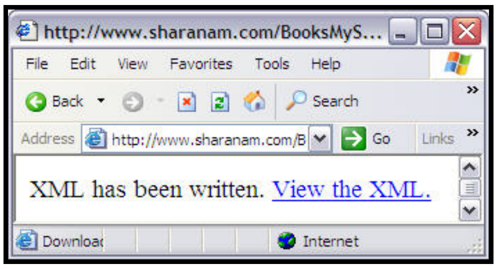
```
Query OK, 0 rows affected (0.09 sec)
```

The following PHP code spec creates an XML file:

```
<?php
    $db_name = "sharanam";
    $connection = mysql_connect('localhost', "root", "sct2306") or die("Could
                                                            not connect.");
    $table_name = 'books';
    $db = mysql_select_db($db_name);
    $query = "SELECT * FROM " . $table_name;
    $result = mysql_query($query) or die("Could not complete database
                                          query");
    $num = mysql_num_rows($result);
    if ($num != 0)
    {
        $file = fopen("results.xml", "w");
        $_xml ="<?xml version=\"1.0\" encoding=\"UTF-8\" ?>\r\n";
        $_xml .="<books>\r\n";

        while ($row = mysql_fetch_array($result))
        {
            $_xml .="\t<book title=\"" . $row["title"] . "\">\r\n";
                $_xml .="\t\t<isbn>" . $row["isbn"] . "</isbn>\r\n";
                $_xml .="\t\t<category>" . $row["category"] .
                            "</category>\r\n";
                $_xml .="\t\t<publisher code=\"" . $row["publishercode"] . "\">"
                            . $row["publisher"] . "</publisher>\r\n";
                $_xml .="\t\t<authors>\r\n";
                    $_xml .="\t\t\t<name>" . $row["authorsname1"] .
                            "</name>\r\n";
                    $_xml .="\t\t\t<name>" . $row["authorsname2"] .
                            "</name>\r\n";
                $_xml .="\t\t</authors>\r\n";
                $_xml .="\t\t<cost>\r\n";
```

```
                    $_xml .="\t\t\t<rupees>" . $row["costrupees"] .
                          "</rupees>\r\n";
                    $_xml .="\t\t\t<dollar>" . $row["costdollar"] .
                          "</dollar>\r\n";
                 $_xml .="\t\t</cost>\r\n";
                 $_xml .="\t\t<rating type=\"".$row["ratingtype"]."\">" .
                       $row["rating"] . "</rating>\r\n";
                 $_xml .="\t\t<covers>\r\n";
                    $_xml .="\t\t\t<topic>" . $row["covertopic1"] .
                          "</topic>\r\n";
                    $_xml .="\t\t\t<topic>" . $row["covertopic2"] .
                          "</topic>\r\n";
                 $_xml .="\t\t</covers>\r\n";
              $_xml .="\t</book>\r\n";
           }

           $_xml .="</books>";
           fwrite($file, $_xml);
           fclose($file);
           echo "XML has been written.  <a href=\"results.xml\">View the
                 XML.</a>";
       }
       else
       {
           echo "No Records found";
       }
  ?>
```

Run the file in the web browser (I.E. or Mozilla) and the page appears as shown in diagram 22.9.1.

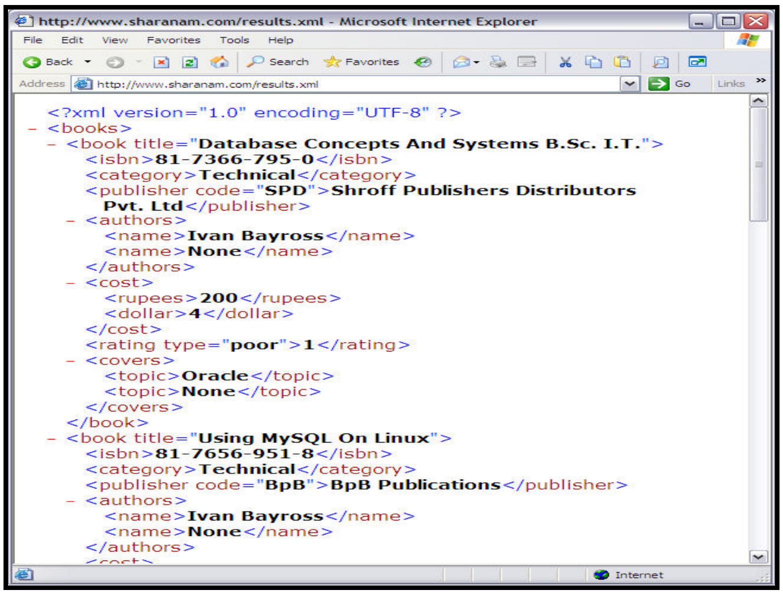

http://www.sharanam.com/BooksMyS...
File Edit View Favorites Tools Help
Back Search
Address http://www.sharanam.com/B Go Links
Download Internet

XML has been written. View the XML.

Diagram 22.9.1: XML file created

At this point, a link appears on the screen.

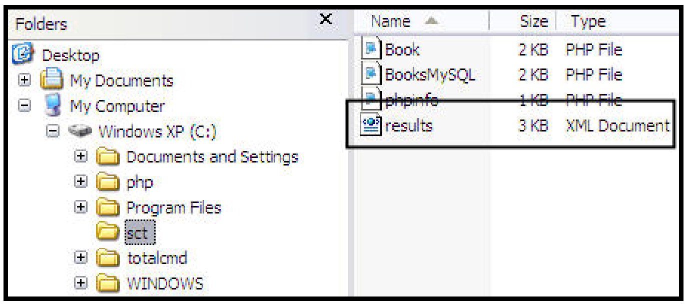
Click **View the XML** link to view the XML file created. The next page of an XML file i.e. **result.xml** file will appear as shown in diagram 22.9.2.


http://www.sharanam.com/results.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.sharanam.com/results.xml
Go Links
Internet

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <books>
  - <book title="Database Concepts And Systems B.Sc. I.T.">
      <isbn>81-7366-795-0</isbn>
      <category>Technical</category>
      <publisher code="SPD">Shroff Publishers Distributors
        Pvt. Ltd</publisher>
    - <authors>
        <name>Ivan Bayross</name>
        <name>None</name>
      </authors>
    - <cost>
        <rupees>200</rupees>
        <dollar>4</dollar>
      </cost>
      <rating type="poor">1</rating>
    - <covers>
        <topic>Oracle</topic>
        <topic>None</topic>
      </covers>
    </book>
  - <book title="Using MySQL On Linux">
      <isbn>81-7656-951-8</isbn>
      <category>Technical</category>
      <publisher code="BpB">BpB Publications</publisher>
    - <authors>
        <name>Ivan Bayross</name>
        <name>None</name>
      </authors>
```

Diagram 22.9.2: An XML file viewed in a browser

result.xml file is created in the folder where the BooksMySQL.php is located. Refer to diagram 22.9.3.



Name	Size	Type
Book	2 KB	PHP File
BooksMySQL	2 KB	PHP File
phpinfo	1 KB	PHP File
results	3 KB	XML Document

Diagram 22.9.3: Location of result.xml file

The Working:

The above code spec,

Establishes a connection with the MySQL DB:

```
$db_name = "sharanam";
$connection = mysql_connect('localhost', "root", "sct2306") or die("Could
                                                          not connect.");
```

Fires a query against a DB table named books:

```
$table_name = 'books';
$db = mysql_select_db($db_name);
$query = "SELECT * FROM " . $table_name;
$result = mysql_query($query) or die("Could not complete database query");
```

Gets the number of rows available in that table:

```
$num = mysql_num_rows($result);
```

If there exists a few records then

- Opens a file stream in write mode
- Creates a variable named **$_XML** to hold
  - The first line of an XML file
  - The Opening Root Node
- Creates a **while** loop to traverse through the records available in that table:

```
if ($num != 0)
{
    $file = fopen("results.xml", "w");
    $_xml ="<?xml version=\"1.0\" encoding=\"UTF-8\" ?>\r\n";
    $_xml .="<books>\r\n";
    while ($row = mysql_fetch_array($result))
    {
        ...
    }
```

Inside the **while** loop the data extracted via the result set is appended in a well-formed XML structure to the **$_XML** variable.

```
while ($row = mysql_fetch_array($result))
{
    $_xml .="\t<book title=\"" . $row["title"] . "\">\r\n";
    $_xml .="\t\t<isbn>" . $row["isbn"] . "</isbn>\r\n";
    $_xml .="\t\t<category>" . $row["category"] . "</category>\r\n";
        $_xml .="\t\t<publisher code=\"" . $row["publishercode"] . "\">"
                    . $row["publisher"] . "</publisher>\r\n";
        $_xml .="\t\t<authors>\r\n";
        $_xml .="\t\t\t<name>" . $row["authorsname1"] . "</name>\r\n";
            $_xml .="\t\t\t<name>" . $row["authorsname2"] . "</name>\r\n";
        $_xml .="\t\t</authors>\r\n";
        $_xml .="\t\t<cost>\r\n";
            $_xml .="\t\t\t<rupees>" . $row["costrupees"] . "</rupees>\r\n";
            $_xml .="\t\t\t<dollar>" . $row["costdollar"] . "</dollar>\r\n";
        $_xml .="\t\t</cost>\r\n";
        $_xml .="\t\t<rating type=\"".$row["ratingtype"]."\">" .
                             $row["rating"] . "</rating>\r\n";
        $_xml .="\t\t<covers>\r\n";
            $_xml .="\t\t\t<topic>" . $row["covertopic1"] . "</topic>\r\n";
            $_xml .="\t\t\t<topic>" . $row["covertopic2"] . "</topic>\r\n";
        $_xml .="\t\t</covers>\r\n";
    $_xml .="\t</book>\r\n";
}
```

Once the variable is populated with all the records

- The closing Root node is appended
- File is written to the Hard Disk Drive
- File stream is closed
- An Appropriate message is displayed

```
$_xml .="</books>";
fwrite($file, $_xml);
fclose($file);
echo "XML has been written.  <a href=\"results.xml\">View the XML.</a>";
```