NEW SYLLABUS
CBCS PATTERN

S.Y. B.B.A. (C.A.)
SEMESTER - IV

ADVANCE PHP

SWATI JADHAV
GAJANAN DESHMUKH
SARITA BYAGAR

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

SPPU New Syllabus

A Book Of

ADVANCE PHP

For B.B.A. (CA) : Semester - IV

[Course Code CA - 404 [Option]]

CBCS Pattern

As Per New Syllabus, Effective from June 2020

Mrs. Swati S. Jadhav
M.C.S, NET, M.Phil,
Assistant Professor,
Department of Computet Science,
MES Abasheb Garware College

Mr. Gajanan A. Deshmukh
B.Sc (Comp.sci), MCA
HOD: BCA DEPT.
Smt. Kashibai Navale College of Commerce, Pune -04.

Ms. Sarita Byagar
M.C.S, NET,
Assistant Professor,
Department of Computer Science,
Indira College of Commerce and Science.

Price ₹ 200.00

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

N4956

Advance PHP **ISBN 978-93-90596-16-4**

First Edition : January 2021

Published By :
NIRALI PRAKASHAN
Abhyudaya Pragati, 1312, Shivaji Nagar,
Off J.M. Road, Pune – 411005
Tel - (020) 25512336/37/39, Fax - (020) 25511379
Email : niralipune@pragationline.com

Polyplate

Printed By :
YOGIRAJ PRINTERS AND BINDERS
Survey No. 10/1A, Ghule Industrial Estate
Nanded Gaon Road
Nanded, Pune - 411041
Mobile No. 9404233041/9850046517

➢ DISTRIBUTION CENTRES

PUNE

Nirali Prakashan (For orders within Pune) : 119, Budhwar Peth, Jogeshwari Mandir Lane, Pune 411002, Maharashtra
Tel : (020) 2445 2044; Mobile : 9657703145
Email : niralilocal@pragationline.com

Nirali Prakashan (For orders outside Pune) : S. No. 28/27, Dhayari, Near Asian College Pune 411041
Tel : (020) 24690204; Mobile : 9657703143
Email : bookorder@pragationline.com

MUMBAI

Nirali Prakashan : 385, S.V.P. Road, Rasdhara Co-op. Hsg. Society Ltd.,
Girgaum, Mumbai 400004, Maharashtra; Mobile : 9320129587
Tel : (022) 2385 6339 / 2386 9976, Fax : (022) 2386 9976
Email : niralimumbai@pragationline.com

➢ DISTRIBUTION BRANCHES

JALGAON

Nirali Prakashan : 34, V. V. Golani Market, Navi Peth, Jalgaon 425001, Maharashtra,
Tel : (0257) 222 0395, Mob : 94234 91860; Email : niralijalgaon@pragationline.com

KOLHAPUR

Nirali Prakashan : New Mahadvar Road, Kedar Plaza, 1st Floor Opp. IDBI Bank, Kolhapur 416 012
Maharashtra. Mob : 9850046155; Email : niralikolhapur@pragationline.com

NAGPUR

Nirali Prakashan : Above Maratha Mandir, Shop No. 3, First Floor,
Rani Jhanshi Square, Sitabuldi, Nagpur 440012, Maharashtra
Tel : (0712) 254 7129; Email : niralinagpur@pragationline.com

DELHI

Nirali Prakashan : Room No. 2, Ground Floor, 4575/15 Onkar Tower, Aggarwal Road, Daryaganj
New Delhi 110002 Mob : +91 9555778814 / +91 9818561840
Email : niralidelhi@pragationline.com

BENGALURU

Nirali Prakashan : Maitri Ground Floor, Jaya Apartments, No. 99, 6th Cross, 6th Main,
Malleswaram, Bengaluru 560003, Karnataka; Mob : 9449043034
Email: niralibangalore@pragationline.com

Other Branches : Hyderabad, Chennai

Preface ...

We take this opportunity to present this book entitled as **"Advance PHP"** to the students of B.B.A (CA) - Fourth Semester. The object of this book is to present the subject matter in a most concise and simple manner. The book is written strictly according to the New Syllabus (CBCS Pattern).

The book has its own unique features. It brings out the subject in a very simple and lucid manner for easy and comprehensive understanding of the basic concepts, its intricacies, procedures and practices. This book will help the readers to have a broader view on Advance PHP. The language used in this book is easy and will help students to improve their vocabulary of Technical terms and understand the matter in a better and happier way.

We sincerely thank Shri. Dineshbhai Furia and Shri. Jignesh Furia of Nirali Prakashan, for the confidence reposed in us and giving us this opportunity to reach out to the students as well as teachers.

We have given our best inputs for this book. Any suggestions towards the improvement of this book and sincere comments are most welcome on niralipune@pragationline.com.

Authors

Syllabus ...

1. Introduction to Object Oriented Programming in PHP **[Lecture 6]**

- 1.1 Classes
- 1.2 Objects
- 1.3 Introspection
- 1.4 Serialization
- 1.5 Inheritance
- 1.6 Interfaces
- 1.7 Encapsulation

2. Web Techniques **[Lecture 4]**

- 2.1 Server information
- 2.2 Processing forms
- 2.3 Sticky forms
- 2.4 Setting response headers

3. XML **[Lecture 8]**

- 3.1 Introduction XML
- 3.2 XML document Structure
- 3.3 PHP and XML
- 3.4 XML parser
- 3.5 The document object model
- 3.6 The simple XML extension
- 3.7 Changing a value with simple XML

4. Ajax with PHP **[Lecture 6]**

- 4.1 Understanding java scripts for AJAX
- 4.2 AJAX web application model
- 4.3 AJAX –PHP framework
- 4.4 Performing AJAX validation
- 4.5 Handling XML data using php and AJAX
- 4.6 Connecting database using php and AJAX

5. Introduction to Web Services **[Lecture 10]**

- 5.1 Definition of web services
- 5.2 Basic operational model of web services, tools and technologies enabling web services
- 5.3 Benefits and challenges of using web services.
- 5.4 Web services Architecture and its characteristics
- 5.5 Core building blocks of web services
- 5.6 Standards and technologies available for implementing web services
- 5.7 Web services communication models
- 5.8 Basic steps of implementing web services.

6. PHP Framework (Joomla / Druple) [Lecture 14]

6.1 Introduction to Joomla/Druple

6.1.1 Introduction

6.1.2 Joomla/Druple features

6.1.3 How joomla/Drupleworks ?

6.1.4 The platformComponents, Modules and Plugins

6.2 Administering Joomla/Druple

6.2.1 Presentation Administration

6.2.2 Content Administration

6.2.3 System Administration

6.3 Working with Joomla/Druple

6.3.1 Adding articles

6.3.2 Adding menus to point to content

6.3.3 Installing new templates

6.3.4 Creating templates

6.3.5 Adding a Module and Component

6.3.6 Modifying the existing templates

6.3.7 Creating templates with web editors

6.3.8 Creating real templates

***

Contents ...

***

1…

Introduction to Object Oriented Programming in PHP

Objectives…

- To learn the benefits of encapsulating code into functions and classes.
- To define and use user-defined classes with properties and methods.
- To understand OOP concepts of visibility, inheritance and interface.
- To study built-in functions for examining classes' and object's characteristics.
- To learn the concept of byte stream representation of an object using serialization.

1.1 INTRODUCTION

- Like C++ and Java, PHP also supports Object Oriented Programming (OOP) concepts.
- OOP was introduced in PHP3 and improved further. OOP supports cleaner designs, easier maintenance and greater code reuse. It also speeds up the development of large applications.
- It includes the fundamental connection between data and the code that works on that data. The class and object are the fundamental construct behind object oriented programming.
- Objects can represent real world entity like student, employee or it can represent conceptual entity link bank account, file. Each object has different values for properties it possesses.
- Objects with similar characteristics are grouped together into a single unit called **class**. The class can contain methods which represents the way in which the object can interact with its data.

- **Advantages of Object-Oriented Programming**:
  1. **Modularization:** The application can be divided into modules.
  2. **Re-usability:** An application can be implemented by adding new modules to the existing modules. This speeds up the application development time.

1.2 CLASSES

- PHP allows you to group together objects having similar characteristics and behaviour into a single unit called as *class.*
- Class is a programmer defined data type, which includes variables and functions.
- Variables within a class are called *properties* which has a name and a value; functions are called *methods* which represent the actions associated with the class. Methods can accept parameters of any valid data type.
- Class is a collection of objects. An object has properties and behaviour.
- Class names are case insensitive and must conform to the rules for PHP identifiers.
- We define our own class by starting with the keyword **'class'** followed by the name class name. Inside a class, you can define variables and functions as shown below:
- **Syntax:**

```
<?php
class class_name
{
    // Properties
    // Methods
}   //end of class
?>
```

1.2.1 Declaring Properties

- Property of a class can be defined as private, public or protected. Property declaration is optional. It's a good programming style to declare it. You can add new properties at any time.
- You can assign default values to properties, but those default values must be simple constants.
- Example:

```
public $name= "Ritika";
public $age = 18;
```

- Using access modifiers, you can change the visibility of properties.
- Properties that are accessible outside the class using object should be declared as *public*; properties on an instance that can only be accessed by methods within the same class should be declared *private*. Finally, properties declared as *protected* can only be accessed by the object's class methods and the class methods of the classes inheriting from the class.

- Defining the visibility of class properties is optional; if a visibility is not specified, it is *public* by default.
- Example:

```
class Demo
{
   private $name = "Atharv";
   protected $age = 20;
   public $class = "SYBBA(CA)";
}
```

- Here `$name` is accessible inside class methods only. `$age` is accessible inside class methods and class methods of classes inheriting from class Demo. `$class` is accessible outside the class using an object of the class type.

1.2.2 Declaring Methods

- A method is a function defined inside a class. In PHP, most methods act only on data of a class in which the method resides.
- Within a method, the `$this` variable contains a reference to an object which invokes the method. Using `$this`, variables of an object can be accessed inside the member function of a class.
- Using access modifiers, you can change the visibility of methods. Methods that are accessible outside the class using object should be declared as *public*; methods on an instance that can only be called by methods within the same class should be declared *private*. Finally, methods declared as *protected* can only be called from within the object's class methods and the class methods of classes inheriting from the class.
- Defining the visibility of class methods is optional; if a visibility is not specified, a method is *public*.
- For Example: PHP script to define a class Employee. With methods to accept and display employee details.

```
<?php
class Employee
{
   private $emp_code, $emp_name, $emp_designation;  // property declaration
   public function accept($c,$n,$d)                //method definition
   {
      $this->emp_code=$c;          //$this is used to access properties of
                                      current invoking object
      $this->emp_name=$n;
```

```
        $this->emp_designation=$d;
    }
    function display()   //default visibility is public
    {
        echo "<br>Emp Code: ".$this->emp_code;
        echo "<br>Emp Name: ".$this->emp_name;
        echo "<br>Emp Designation: ".$this->emp_designation;
    }
}
```

- If you call `$emp->accept()` then inside accept method, $this hold the same value as `$emp`. Methods use the `$this` variable to access the properties of the current object. As you can see, the `accept()` and `display()` methods use `$this` to access and set `emp_code, emp_name and emp_designation` properties of the current object.

1.2.3 Static Property and Static Method

- PHP allows you to define static properties, which are variables of class and can be accessed by referencing the property with the class name.
- Also to access methods in terms of a class rather than an object, use **static** keyword. Any method declared as static is accessible without the creation of an object. Static functions are associated with the class, and not with an instance of the class. They are permitted to access only static methods and static variables.
- Inside a class, you can refer to the static property using the **self** keyword.
- **Syntax**: `self::static_var_name;`

Program 1.1: Write a PHP script to illustrate the concept of static property and static method.

```
<?php
class example_static
{
    static $count;   //static property
    public static function updateCount()
    {
        //static method
        return self::$count++;
    }
}
example_static::$count = 1;
for($i = 0; $i < 5; ++$i)
{
     echo 'The value is: '.example_static::updateCount() . "<BR>";
}
?>
```

Output:

```
The value is: 1
The value is: 2
The value is: 3
The value is: 4
The value is: 5
```

1.3 OBJECTS

- An individual instance of the data structure defined by a class is called as Object. You define a class once and then declare objects of that class. Building an object using a class is known as instantiation. Instantiating an object requires:
  - Memory allocation into which load the object.
  - The data that will populate the values of the properties.
- An object has its own individual identity. Each object contains data and code to manipulate the data.
- Objects of different classes can interact with each other without knowing details of other's data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.

Declaring an Object:

- To declare an object of a given class, the `new` keyword is used:
- **Syntax:**

```
$object = new Class;
```

- **Example:**

```
$emp = new Employee;
```

Accessing Properties and Methods:

- Class methods and properties can be directly accessed through the object instance using the object access operator (->) notation.
- **Syntax:**

```
$object->property_name     //if visibility of property is public
$object->method_name([arg, ... ])
```

- **Example:**

```
$emp->accept(1,"Atharv", "Manager");
$emp->display();
```

- **Note:** Classes are manipulated at design time when you make changes to the methods or properties and objects are manipulated at runtime when values are assigned to their properties and their methods are invoked.

Initializing an object using Constructor:

- To initialize the object, Constructor is special type of function of a class which is automatically executed as any object of that class is created or instantiated. Usually it starts with two underscore characters.
- **Syntax:**

```
function __construct(args..)
{
   //definition
}
```

- You may provide a list of arguments following the class name when instantiating an object. These arguments are passed to a constructor, that initializes the properties of the class.

Destroying an object using Destructor:

- When an object is destroyed, such as when the requested page has completed running, when the variable falls out of scope, when it is explicitly set to null, or the end of the script is reached, its destructor is called. Destructor does not take any parameters.
- **Syntax:**

```
function __destruct()
{
   //definition
}
```

- This helps to perform any last minute clean up, such as closing file handle or database connections that might have been opened using class.

Program 1.2: Write a PHP script to define a class Employee with `emp_code, emp_name and emp_designation` as data members. Define constructor and destructor for the class. Also define display() function to display employee details.

```
<?php
class Employee
{
   private $emp_code;
   private $emp_name;
   private $emp_designation;
   function __construct($c,$n,$d)     //constructor definition
   {
      $this->emp_code=$c;
      $this->emp_name=$n;
```

```
        $this->emp_designation=$d;
        echo "<br> Object is created";
    }
    function display()
    {
        echo "<br>Emp Code: ".$this->emp_code;
        echo "<br>Emp Name: ".$this->emp_name;
        echo "<br>Emp Designation: ".$this->emp_designation;
    }
    function __destruct()   //destructor definition
    {
        echo "<br>Object is destroyed";
    }
}
$c=1;
$n="Atharv ";
$d="Manager";
//constructor called automatically when object is created
$emp=new Employee($c,$n,$d);
$emp->display();
?>
```

Output:

Object is created
Emp Code: 1
Emp Name: Atharv
Emp Designation: Manager
Object is destroyed

In above PHP script, the statement `$emp=new Employee($c,$n,$d)` invokes the constructor function automatically when object is created and at the end of the script destructor is called automatically.

Program 1.3: Write a PHP script to define a class Rectangle with length and breadth as data members. Define constructor and destructor for the class. Also define `area()` and `perimeter()` functions to display area and perimeter of a rectangle.

```
<?php
class Rectangle
{
    function Rectangle($l,$b)
    {
        $this->l=$l;
```

```
        $this->b=$b;
        echo "<br> Object is initialized";
        echo "<br> Length = $this->l, Breadth= $this->b";
    }
    function area()
    {
        $a=$this->l*$this->b;
        echo "<br>Area of rectangle =$a";
    }
    function perimeter()
    {
        $p=2*($this->l+$this->b);
        echo "<br>Perimeter of rectangle =$p";
    }
    function __destruct()   //destructor definition
    {
        echo "<br>Object is destroyed";
    }
}
$l=3.2;
$b=5.4;
$r=new Rectangle($l,$b);
$r->area();
$r->perimeter();
?>
```

Output:

Object is initialized
Length = 3.2, Breadth = 5.4
Area of rectangle = 17.28
Perimeter of rectangle = 17.2
Object is destroyed.

In above PHP script, the statement `$r=new Rectangle($l,$b)` invokes the constructor function automatically when object is created. The object $r of a class Rectangle invokes `area()` and `perimeter()` method using -> symbol and displays the output. At the end of the script destructor is called automatically.

1.4 INTROSPECTION

- It is the ability of a program to examine an object's characteristics, such as its name, parent class (if any), properties and methods.
- With introspection, you can write code that operates on any class or object. You don't need to know which methods or properties are defined when you write your code; instead, you can discover that information at runtime.

Examining Classes:

- To determine whether a class exists, use the `class_exists()` function, which takes in a string and returns a Boolean value.
  Syntax: `$yes_no = class_exists(classname);`
- Alternately, you can use the `get_declared_classes()` function, which returns an array of defined classes and you can verify the class name in the returned array:
  Syntax: `$classes = get_declared_classes();`
- You can get the methods and properties that exist in a class (including those that are inherited from super classes) using the `get_class_methods()` and `get_class_vars()` functions. These functions take a class name and returns an array:
  Syntax:
  ```
  $methods = get_class_methods(classname);
  $properties = get_class_vars(classname);
  ```
- **Note:** `get_class_vars()` returns only properties that have default values; there's no way to discover uninitialized properties.
- Use `get_parent_class()` to find a class's parent class name.
  Syntax: `$superclass = get_parent_class(classname);`

Examining an Object:

- To get the class to which an object belongs, first check if it is an object using the `is_object()` function, then get the class with the `get_class()` function:
  Syntax:
  ```
  $yes_no = is_object(var);
  $classname = get_class(object);
  ```
- Before calling a method on an object, you can check that it exists using the `method_exists()` function:
  Syntax: `$yes_no = method_exists(object, method);`
  Calling an undefined method triggers a runtime exception.
- `get_object_vars()` returns an array of properties that are set in an object.
  Synatx: `$array = get_object_vars(object);`

Program 1.4: Write a PHP script to illustrate the use of functions to examine characteristics of classes and objects.

```
<?php
class A
{
  private $x=10, $y;
  function A($a)
```

```
    {
        $this->y=$a;
        echo "<BR> Constructor A is called";
    }
    function displayA()
    {
        echo "<BR> x=$this->x y=$this->y";
    }
}
class B extends A
{
    private $p,$q;
    function B($l,$m,$n)
    {
        parent::A($l);
        $this->p=$m;
        $this->q=$n;
        echo "<BR> Constructor B is called";
    }
    function displayB()
    {
        echo "<BR> p=$this->p q=$this->q";
    }
}
$obj = new B(20,30,40);
$obj->displayA();
$obj->displayB();
echo "<BR>Functions examining classes<BR>";
echo "<BR> Class A exists or not: ";
echo class_exists('A');
echo "<BR> Class C exists or not: ";
echo class_exists('C');
echo "<BR> Get existing all classes:<BR>";
$allclasses= get_declared_classes();
echo "<BR> Get methods of class A:<BR>";
print_r(get_class_methods('A'));
echo "<BR> Get methods of class B:<BR>";
print_r(get_class_methods('B'));
```

```
echo "<BR> Get variables of class A<BR>";
var_dump(get_class_vars('A'));
echo "<BR> Get variables of class B(no property with default value)<BR>";
print_r(get_class_vars('B'));
echo "<BR> Parent of class B";
echo "<br>".get_parent_class($obj);  //from object of class B
echo "<br>".get_parent_class('B');   // from class name
echo "<BR> Parent of class A";  //no parent so no output
echo get_parent_class('A');

//Functions examining objects
echo "<BR>obj is obejct or not: ";
echo is_object($obj);
echo "<BR>obj1 is obejct or not: ";  //obj1 is not declared an object
echo is_object($obj1);
echo "<BR>Get obj's class name: ";
echo get_class($obj);
echo "<BR>Method displayA() exists or not: ";
echo method_exists($obj, 'displayA');
echo "<BR>Method displayC() exists or not: ";
echo method_exists($obj, 'displayC');
echo "<BR> Get variables of class B using object(no property with default
value) :<BR>";
print_r(get_object_vars($obj));
?>
```

Output:

Constructor A is called

Constructor B is called

x=10 y=20

p=30 q=40

Functions examining classes

Class A exists or not: 1 // Class A exists

Class C exists or not: // Class C does not exists

Get existing all classes:

Get methods of class A:

```
Array ( [0] => A [1] => displayA )  // Methods of a class  A are returned as array elements
Get methods of class B:
Array ( [0] => B [1] => displayB [2] => A [3] => displayA )
// Methods of a class B are its own methods along with methods of base class A are returned as array elements

Get variables of class A
array(0) { }       //plz check it should output array( [x] => 10)
Get variables of class B(no property with default value)
Array( )         //plz check it should output array( [x] => 10)
Parent of class B
A         // from object of class B
A         // from class name B
Parent of class A:              //As class A doesn't have base class, no output is displayed
obj is obejct or not: 1        //obj is declared so output is 1
obj1 is obejct or not:      //obj1 is not declared  so error is displayed

Get obj's class name: B
Method displayA( ) exists or not: 1     //displayA() method exists so output is 1
Method displayC( ) exists or not:     //displayC() method doesn't exists so no output

Get variables of class B using object(no property with default value) :
Array ()                    //plz check it should output array( [x] => 10)
```

1.5 SERIALIZATION

- The conversion of any PHP value to a byte stream representation is called as Serialization. This byte stream can be stored in a file.
- **Syntax**:

```
string serialize(value);
```

- When an object is serialized, it will save all variables in an object. Also, it stores the name of the class. The methods in an object will not be saved.
- To convert a byte stream representation back to a PHP value, use `unserialize()` function.
- When you unserialize an object, its class must be defined before unserialization occurs. Unserializing an object whose class is not yet defined puts the object into `stdClass`, which is useless.
- **Syntax**:

```
value unserialize(string);
```

Program 1.5: Write a PHP script to illustrate the use of serialize() and unserialize() functions.

```
<?php
class A
{
  private $x=10, $y;
  function A($a)
  {
       $this->y=$a;
       echo "<BR> Object is Initialized";
  }
  function displayA()
  {
       echo "<BR> x=$this->x y=$this->y";
  }
}
$obj=new A(20);
echo "<BR> Object before writing to a file";
$obj->displayA();
$bytestream=serialize($obj);   //$obj converted to a bytestream
echo "<BR><BR> Bytestream representation: $bytestream";
$fp=fopen("test.txt","w");
fwrite($fp,$bytestream);      //bytestream is written to a file
fclose($fp);
$fp=fopen("test.txt","r");
$bytestream1=fread($fp,filesize("test.txt")); //read bytestream from a file
fclose($fp);
echo "<BR>";
$obj1=unserialize($bytestream1);  //converts bytestream back to an object
echo "<BR> Object after reading from a file";
$obj1->displayA();
echo "<BR><BR>Object using var_dump:<BR> ";
var_dump($obj1);                  // displays details of $obj1
?>
```

Output:

Object is Initialized
Object before writing to a file
`x=10 y=20`

Bytestream representation: `O:1:"A":2:{s:4:"Ax";i:10;s:4:"Ay";i:20;}`
Object after reading from a file
`x=10 y=20`
Object using `var_dump`:
`object(A)#2 (2) { ["x":"A":private]=> int(10) ["y":"A":private]=> int(20) }`

In previous PHP script, `$obj` is converted to a byte stream using serialize() function. Then this `byte stream` can be used anywhere. In this case is stored in a file. Read this byte stream stored in a file and use `unserialize()` function to convert it back to an object. Then any member functions can be invoked using this object.

1.6 INHERITANCE

- Inheritance is an important feature of Object-Oriented Programming.
- The capability of a class to derive properties and methods from another class is called Inheritance.
- The class whose properties and method are inherited into a new class is called as base class and the class which inherits is called as derived class.
- The derived class do not have to define all the properties and functions again and again, as these are inherited from base class that possesses it. This allows re-usability of the code and speeds up the overall development process.
- To inherit the properties and methods from one class into another class, use the **extends** keyword in the class definition, followed by the name of the base class.
- The public property or method of base class will be public in the derived class. The protected members will remain protected after inheritance. The private property will not be inherited, but it can be manipulated through public inherited methods of base class. The private methods of base class can be invoked by methods of derived class.
- **Syntax:**

```
class derived_classname extends base_classname
{
    //properties
    //methods
}
```

- **Example:**

```
class Employee
{
   private $emp_code, $emp_name, $emp_designation;
}
class EmpoyeeSalary extends Employee
{
   private $basic_pay, $earning, $deduction;
}
```

- The `classEmpoyeeSalary` contains the `$basic_pay`, `$earning` and `$deduction`, as well as the `$emp_code`, `$emp_name and $emp_designation` properties inherited from the Employee class.
- If a derived class has a property or method with the same name as one in its parent class, the property or method in the derived class takes precedence over, or overrides, the property or method in the parent class. Referencing the property returns the value of the property on the child, while referencing the method calls the method on the child.

- **Note:** Override a method in a subclass when the parent class's implementation is different from that required by the subclass. To access an overridden method on an object's parent class, use the parent::*method*() notation:
- **Example:**

```
parent::display();
```

- Similarly in the derived class, parent class constructor can be invoked as,

```
parent::__construct();
```

Program 1.6: Write a PHP script to define a class Employee with emp_code, emp_name and emp_designation as data members. Define constructor, destructor and display() as member functions. Derive a class EmployeeSalary from Employee with constructor and member function CalculateSalary() to calculate the net salary of an employee.

```
<?php
class Employee
{
   private $emp_code, $emp_name, $emp_designation;
   function __construct($c,$n,$d)     //constructor definition
   {
      $this->emp_code=$c;
      $this->emp_name=$n;
      $this->emp_designation=$d;
      echo "<br> Object is created";
   }
   function display()
   {
      echo "<br>Emp Code: ".$this->emp_code;
      echo "<br>Emp Name: ".$this->emp_name;
      echo "<br>Emp Designation: ".$this->emp_designation;
   }
   function __destruct()   //destructor definition
   {
      echo "<br>Object is destroyed";
   }
}
class EmployeeSalary extends Employee
{
   private $basic_pay, $earnings, $deduction;
   function EmployeeSalary($c,$n,$d,$b,$e,$dd)
```

```
    {
        parent::__construct($c,$n,$d);
        $this->basic_pay=$b;
        $this->earnings=$e;
        $this->deduction=$dd;
    }
    function CalculateSalary()
    {
        echo "<br> Net Salary(Rs.) :".($this->basic_pay+ $this->earnings-
    $this->deduction);
    }
}
$c=1;
$n="Atharv ";
$d="Manager";
$b=10000;
$e=500;
$dd=750;
$emp=new EmployeeSalary($c,$n,$d,$b,$e,$dd);
$emp->display();
$emp->CalculateSalary();
?>
```

Output:

Object is created

Emp Code: 1

Emp Name: Atharv

Emp Designation: Manager

Net Salary(Rs.) :9750

Object is destroyed

1.6.1 Final Method

- If you declare the method using **final** keyword in superclass/parent class, then subclass/child class can't override that method. Method should not be overridden due to security reason.
- Properties and constants cannot be declared final, only classes and methods may be declared as final.

Program 1.7: Write a PHP script to illustrate the concept of final method.

```
<?php
class A
{
   private $message= "Hi";
   final function display()
   {
          echo $this->message;
   }
}
class B extends A
{
   function display()
   {
      echo "Hello";
   }
}
?>
```

Output:

Fatal error: Cannot override final method A::display()
n D:\xampp\htdocs\OOP\finalmethod.php on line 12

In the derived class B, display() function can't be redefined as it is declared as final in base class A, so you will get error message.

Program 1.8: Write a PHP script to illustrate the concept of final method.

```
<?php
class A
{
   private $message= "Hi<br>";
   final function display()
   {
          echo $this->message;
   }
}
class B extends A
```

```
{
   function displayB()
   {
      echo "Hello";
   }
}
$obj=new B;
$obj->display();
$obj->displayB();
?>
```

Output:

Hi
Hello

- Here $obj, object of class B can invoke display() function of it's base class A without overriding it. Also $obj invokes its own member function display().

1.6.2 Abstract Class and Abstract Method

- Abstraction is a way of hiding information. In abstraction, there should be at least one method that must be declared but not defined.
- Any class that contains at least one abstract method must also be declared as abstract.
- **Syntax :**

```
abstract class classname
{
   //abstract method declaration
   // non-abstract method definition
}
```

Characteristics of abstract class:

- There must be an abstract keyword that must be written before this class for it to be an abstract class.
- This class cannot be instantiated. Only the class that implements all methods of an abstract class can be instantiated. There can be more than one methods that can be left undefined.
- If the derived class which inherits the abstract class doesn't define all abstract method then the derived class can't be instantiated.

Program 1.9: Write a PHP script to illustrate the concept of abstract class and abstract method.

```
<?php
abstract class A
{
   abstract function one();
   public function two()
```

```
    {
        echo "Non-abstract method";
    }
}
class B extends A
{
    public function one()
    {
        echo "Abstract Function one defined by subclass<br/>";
    }
}
$obj = new B();
$obj->one();
$obj->two();
?>
```

Output:

Abstract Function one defined by subclass.

Non-abstract method.

1.7 INTERFACES

- An Interface allows the users to create programs, specifying the public methods that a class must implement. The interface contains no data variables.
- It does not tell how these methods should be implemented. The method implementation depends upon the class which implements them.

Characteristics of an Interface:

- Interface is similar to a class except that it cannot contain code.
- An interface can define method names and arguments, but not the contents of the methods.
- All methods declared in an interface must be public.
- Any classes implementing an interface must implement all methods defined by the interface.
- A class can implement multiple interfaces.
- An interface is declared using the "interface" keyword.
- Interfaces can't maintain Non-abstract methods.

Advantage of an Interface:

- It separates the implementation and defines the structure.
- It is used to define a generic template.

Program 1.10: Write a PHP script to illustrate the concept of interface.

```
<?php
interface myInterface
{
   public function method_one();
   public function method_two();
}
class myClass implements myInterface
{
   public  function method_one()
   {
      echo "<BR> Method one is implemented";
   }
   public  function method_two()
   {
      echo "<BR> Method two is implemented";
   }
}
$obj=new myClass ();
$obj->method_one();
$obj->method_two();
?>
```

Output:

Method one is implemented.

Method two is implemented.

1.8 ENCAPSULATION

- A class is kind of a container or capsule, which contains properties and a set of methods to manipulate these properties. This is called as **Encapsulation**. It binds together code and the data it manipulates.
- The variables or data of a class are hidden from any other class and can be accessed only through any member function of that class. The data in a class is hidden from other classes is called as **Data Hiding**.
- This allows a class to change its internal implementation without affecting overall functionality. The details of implementation of class methods are hidden from the users of the object.

- This is a fundamental concept of object oriented programming. Encapsulation refers to: protection of a class's. Internal data from the code outside that class and the hiding of the details of implementation. Since other classes don't have direct access to the properties, you can change the way the methods get and set the properties.
- Generally, all data members of a class should be declared private. Any access needed to this data by code outside the class should be done through a public method. The protected members can be accessed by methods which are declared as public as well as derived class's public methods.

Advantages of Encapsulation:

1. You can change implementation details at any time without affecting code that uses class.
2. Outside class can't modify property values of an object built from your class without your knowledge.

PRACTICE PROGRAMS

Program 1.11: Write a PHP script to define a class with two numbers as data members and define members functions to perform arithmetic operations.

```
<?php
class cal
{
   private $no1, $no2;
   function cal($x,$y)
   {
      $this->no1=$x;
      $this->no2=$y;
   }
   function add()
   {
      return ($this->no1+$this->no2);
   }
   function sub()
   {
      return ($this->no1-$this->no2);
   }
   function mul()
   {
      return ($this->no1*$this->no2);
   }
   function div()
```

```
    {
        if($this->no2!=0)
        return ($this->no1)/($this->no2);
    }
}
$x=4;
$y=5;
$obj=new cal(4,5);
echo "Number1= $x <BR> Number2= $y";
echo "<BR> Addition is : ".$obj->add();
echo "<BR> Subtraction is : ".$obj->sub();
echo "<BR> Multiplication is : ".$obj->mul();
echo "<BR> Division is : ".$obj->div();
?>
```

Output:

Number1= 4
Number2= 5
Addition is : 9
Subtraction is : –1
Multiplication is : 20
Division is : 0.8

Program 1.12: Write a PHP script to define an interface which has methods area( ), volume( ). Define constant PI. Create a class Cylinder which implements interface methods and calculate area and volume. Also declare array of objects and call member functions area() and volume().

```
<?php
define('PI', '3.14');

interface Myinterface
{
    public function area();
    public function volume();
}
class cylinder implements Myinterface
{
    private $r, $h;
    function cylinder($r,$h)
    {
        $this->r=$r;
```

```
        $this->h=$h;
    }
    public function area()
    {
        $a=2*PI*$this->r*($this->r+$this->h);
        echo "<br>Area of cylinder= :$a";
    }
    public function volume()
    {
        $v=PI*$this->r*$this->r*$this->h;
        echo "<br>Volume of cylinder= :$v";
    }
}
$r=3;
$h=4;
$obj=new cylinder($r,$h);      //constructor gets invoked for $obj object
echo "<br> Cylinder 1:";
$obj->area();
$obj->volume();
$c[]=array();                  // array of objects
$c[0]=new cylinder(2,3);       //constructor gets invoked for $c[0] object
echo "<br><br> Cylinder 2:";
$c[0]->area();
$c[0]->volume();
$c[1]=new cylinder(6,7);       //constructor gets invoked for $c[1] object
echo "<br><br> Cylinder 3:";
$c[1]->area();
$c[1]->volume();
?>
```

Output:

Cylinder 1:
Area of cylinder= :131.88
Volume of cylinder= :113.04

Cylinder 2:
Area of cylinder= :62.8
Volume of cylinder= :37.68

Cylinder 3:
Area of cylinder= :489.84
Volume of cylinder= :791.28

- In above PHP script, `$obj` is initialized by calling constructor as follows:
  `$obj=new cylinder($r,$h);`
- The following statement initializes object from an array:
  `$c[0]=new cylinder(2,3);`
- The objects `$c[0]` and `$c[1]` invokes `area()` and `volume()` functions using `->` operator. The `area()` and `volume()` functions uses respective object's data members for computation of area and volume.

Summary

- PHP supports object oriented concepts. Class can be defined using a keyword class. Using new keyword, object can be declared.
- Using object, we can access its properties and methods using `->` operator.
- The constructor are used to initialize the object and destructor is used release the memory when it goes out of scope.
- We also learned about different functions related to observing characteristics of an object and a class.
- Use `serialize()` and `unserialize()` functions to convert object to byte stream and byte stream to object respectively.
- Using extends keyword, the properties and methods of one class can be inherited into another class. Also interface can be written using keyword interface, which can be implemented by a class.
- In abstraction, there should be at least one method that must be declared but not defined. The class that inherit this abstract class need to define that method.
- PHP also supports to define a class with static property and static method.
- The ability to hide the details of implementation of class methods from the users of the object is called as Encapsulation.
- All data members of a class should be declared private. Any access needed to those variables by code outside the class should be done through a public method.

Check Your Understanding

1. Which notation is used to access variables of an object?
   (a) :: (b) =
   (c) -> (d) .
2. Which notation should be used to refer to a method in the context of a class rather than an object you use?
   (a) -> (b) _
   (c) $ (d) ::
3. Which method is invoked just before an object is garbage collected?
   (a) _collect() (b) _garbage()
   (c) _destruct( ) (d) _destructor()
4. Which one of the following property scopes is not supported by PHP?
   (a) friendly( ) (b) final
   (c) public (d) static

5. Which function is used to check if class exists or not?
   (a) exist() (b) exist_class()
   (c) class_exists( ) (d) _exist()
6. Which keyword is used to refer to properties or methods within the class itself?
   (a) private (b) public
   (c) protected (d) $this
7. Objects are also known as _____.
   (a) reference (b) class
   (c) instance (d) template
8. A member function typically accesses members of _____ object only.
   (a) current (b) next
   (c) previous (d) all of the above
9. The practice of separating the user from the true inner workings of an application through well-known interfaces is known as_____.
   (a) Polymorphism (b) Inheritance
   (c) Encapsulation (d) Dynamic Message passing
10. PHP recognizes constructor by the name _____
    (a) classname() (b) __constrct
    (c) function _construct() (d) function __construct()
11. Which one of the following can be used to instantiate an object in PHP assuming class name to be Foo?
    (a) $obj = new $foo; (b) obj = new foo;
    (c) $obj = new foo (); (d) obj = new foo();
12. Which one of the following is the right way to invoke a method?
    (a) $object->methodName(); (b) object->methodName();
    (c) object->methodName; (d) $object::methodName;

ANSWERS

1. (c)	2. (d)	3. (c)	4. (a)	5. (c)	6. (d)	7. (c)	8. (a)
9. (c)	10. (d)	11. (c)	12. (a)				

Practice Questions

Q. I Answer the following questions in short.

1. State the purpose of extends keyword?
2. How to create object in PHP ?
3. Which function is used to check if class is exists or not?
4. State the purpose of self keyword?
5. How static method is invoked?
6. What is the use of abstract class?

7. State the purpose of parent keyword?
8. Write characteristics of abstract class.
9. Write characteristics of an interface.
10. What is serialization?

Q. II Answer the following questions.

1. How to call a constructor of a parent class from a child class? Explain with suitable example.
2. What is Inheritance ? Explain with suitable example.
3. Explain the concept of static methods and static property with suitable example.
4. What is an Interface? Illustrate the concept with suitable example.
5. What is abstract class and abstract method? Illustrate with suitable example.
6. What is method overriding? Explain with suitable example.
7. Explain different functions with syntax to examine characteristics of an object and a class.
8. Write a PHP Script to create a class Worker that has data members as `Worker_Name, No_of_Days_worked, Pay_Rate`. Define parameterized constructor. Also write necessary member function to calculate and display the salary of worker.
9. Write a PHP program to create a class temperature which contains data members as Celsius and Fahrenheit. Define parameterized constructor to initialize all values of temperature object. Convert Celsius to Fahrenheit and Convert Fahrenheit to Celsius using member functions. Display conversion on next page.
10. Define a class Employee having private members – id, name, department, salary. Define parameterized constructors. Create a subclass called “Manager" with private member bonus. Create 3 objects of the Manager class and display the details of the manager having the maximum total salary (salary + bonus).

Q III Define the terms

1. Class
2. Object
3. Encapsulation
4. Constructor
5. Destructor
6. Inheritance
7. Serialization
8. Method Overriding
9. Final method
10. Abstract class

***

2... Web Techniques

Objectives...

- To understand server information.
- To study about processing forms.
- To learn how to use sticky forms.
- To study how to set response headers.

2.1 INTRODUCTION

- Form processing is essential component in almost every web applications, through forms as interface user communicates with server.
- In PHP program, form is used to display the form. Which involves HTML tags such as text boxes, checkbox, radio buttons, buttons etc. After creating a form you can insert data into it and that information is sent to the server, which will then pass on to database and it gets stored into it, another way is to pull the data from the database and display on to form, to retain information on to the form we use the concept of sticky forms.
- Along with forms we use GET, POST variables to send the data to the server. Similarly, `$_SERVER[]` global array is also used for server information, it is used for setting response headers of page.

2.2 SERVER INFORMATION

- In addition to `PHP_SELF` and `REQUEST_METHOD` the `$_SERVER` auto global array contains number of useful elements that provide information on to the web server and current request
- `$_SERVER` array contains useful information from the web server. `$_SERVER` is a super global variable it contains information about headers, path, host and script locations.
- With these global indices, the `$_SERVER` array contains values for the meta variable listed with the specification of CGI (Common Gateway Interface).

- Following is the list of entries in `$_SERVER` that comes from CGI.

Sr. No.	Element/Code	Description
1.	`$_SERVER['PHP_SELF']`	Returns the filename of the currently executing script.
2.	`$_SERVER['GATEWAY_INTERFACE']`	Returns the version of the Common Gateway Interface (CGI) the server is using.
3.	`$_SERVER['SERVER_ADDR']`	Returns the IP address of the host server.
4.	`$_SERVER['SERVER_NAME']`	Returns the name of the host server (such as www.nirali.com).
5.	`$_SERVER['SERVER_SOFTWARE']`	Returns the server identification string (such as Apache/2.2.24).
6.	`$_SERVER['SERVER_PROTOCOL']`	Returns the name and revision of the information protocol (such as HTTP/1.1).
7.	`$_SERVER['REQUEST_METHOD']`	Returns the request method used to access the page (such as POST).
8.	`$_SERVER['REQUEST_TIME']`	Returns the timestamp of the start of the request (such as 1377687496).
9.	`$_SERVER['QUERY_STRING']`	Returns the query string if the page is accessed via a query string.
10.	`$_SERVER['HTTP_ACCEPT']`	Returns the Accept header from the current request.
11.	`$_SERVER['HTTP_ACCEPT_CHARSET']`	Returns the `Accept_Charset` header from the current request (such as utf-8, ISO-8859-1).
12.	`$_SERVER['HTTP_HOST']`	Returns the Host header from the current request.
13.	`$_SERVER['HTTP_REFERER']`	Returns the complete URL of the current page (not reliable because not all user-agents support it).
14.	`$_SERVER['HTTPS']`	Is the script queried through a secure HTTP protocol?
15.	`$_SERVER['REMOTE_ADDR']`	Returns the IP address from where the user is viewing the current page.
16.	`$_SERVER['REMOTE_HOST']`	Returns the Host name from where the user is viewing the current page.

Sr. No.	Element/Code	Description
17.	`$_SERVER['REMOTE_PORT']`	Returns the port being used on the user's machine to communicate with the web server.
18.	`$_SERVER['SCRIPT_FILENAME']`	Returns the absolute pathname of the currently executing script.
19.	`$_SERVER['SERVER_ADMIN']`	Returns the value given to the SERVER_ADMIN directive in the web server configuration file (if your script runs on a virtual host, it will be the value defined for that virtual host) (such as `someone@w3schools.com`).
20.	`$_SERVER['SERVER_PORT']`	Returns the port on the server machine being used by the web server for communication (such as 80).
21.	`$_SERVER['SERVER_SIGNATURE']`	Returns the server version and virtual host name which are added to server-generated pages.
22.	`$_SERVER['PATH_TRANSLATED']`	Returns the file system based path to the current script.
23.	`$_SERVER['SCRIPT_NAME']`	Returns the path of the current script.
24.	`$_SERVER['SCRIPT_URI']`	Returns the URI of the current page.

- The Apache server also creates entries in the $_SERVER array for each HTTP header in the request. For each key, the header name is converted to uppercase, hyphens (-) are turned into underscores (_), and the string "`HTTP_`" is prepended. For example, the entry for the User-Agent header has the key "`HTTP_USER_AGENT`".
- The two most common and useful headers are:

1. **`HTTP_USER_AGENT`:** The string the browser used to identify itself (e.g., "Mozilla/5.0 (Windows 2000;U) Opera 6.0 [en]").
2. **`HTTP REFFERER`:** The page the browser said it came from to get to the current page (e.g., "`http://www.example.com/last_page.html`").

Program 2.1: PHP program to demonstrate $_SERVER information.

```
<!DOCTYPE html>
<html>
<body>
<?php
echo "php self=".$_SERVER['PHP_SELF'];
echo "<br>";
echo "server name=".$_SERVER['SERVER_NAME'];
echo "<br>";
echo "server host=".$_SERVER['HTTP_HOST'];
echo "<br>";
echo "http user agent=".$_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo "script name=".$_SERVER['SCRIPT_NAME'];
echo "<br>";
echo "server software".$_SERVER['SERVER_SOFTWARE'];
echo "<br>";
echo "gateway interface=".$_SERVER['GATEWAY_INTERFACE'];
echo "<br>";
echo "server protocol".$_SERVER['SERVER_PROTOCOL'];
echo "<br>";
echo "server port=".$_SERVER['SERVER_PORT'];
echo "<br>";
echo "remote addr=".$_SERVER['REMOTE_ADDR'];
?>
</body>
</html>
```

Output:

```
php self=/phpexamples/phpserver.php
server name=localhost
server host=localhost
http user agent=Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
script name=/phpexamples/phpserver.php
server softwareApache/2.4.41 (Ubuntu)
gateway interface=CGI/1.1
server protocolHTTP/1.1
server port=80
remote addr=127.0.0.1
```

Program 2.2: PHP program to display all the $_SERVER parameters.

```
<?php
foreach ($_SERVER as $parm => $value)
echo "<br>"."$parm = '$value'\n";
?>
```

Output:

```
HTTP_HOST = 'localhost'
HTTP_USER_AGENT = 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0'
HTTP_ACCEPT = 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8'
HTTP_ACCEPT_LANGUAGE = 'en-US,en;q=0.5'
HTTP_ACCEPT_ENCODING = 'gzip, deflate'
HTTP_CONNECTION = 'keep-alive'
HTTP_UPGRADE_INSECURE_REQUESTS = '1'
HTTP_CACHE_CONTROL = 'max-age=0'
PATH = '/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin'
SERVER_SIGNATURE = '
Apache/2.4.41 (Ubuntu) Server at localhost Port 80
'
SERVER_SOFTWARE = 'Apache/2.4.41 (Ubuntu)'
SERVER_NAME = 'localhost'
SERVER_ADDR = '127.0.0.1'
SERVER_PORT = '80'
REMOTE_ADDR = '127.0.0.1'
DOCUMENT_ROOT = '/var/www/html'
REQUEST_SCHEME = 'http'
CONTEXT_PREFIX = ''
CONTEXT_DOCUMENT_ROOT = '/var/www/html'
SERVER_ADMIN = 'webmaster@localhost'
SCRIPT_FILENAME = '/var/www/html/phpexamples/serverentries.php'
REMOTE_PORT = '47756'
GATEWAY_INTERFACE = 'CGI/1.1'
SERVER_PROTOCOL = 'HTTP/1.1'
REQUEST_METHOD = 'GET'
QUERY_STRING = ''
REQUEST_URI = '/phpexamples/serverentries.php'
SCRIPT_NAME = '/phpexamples/serverentries.php'
PHP_SELF = '/phpexamples/serverentries.php'
REQUEST_TIME_FLOAT = '1606737480.339'
REQUEST_TIME = '1606737480'
```

2.3 PROCESSING FORMS

- Forms are essential parts in web development. Forms are used to communicate between users and the server.
- Form is a used to get information of the user to the server and let the server do something in response to the user's input.
- It's very easy to process forms with PHP, you can process information collected by HTML form and you can use PHP code to make decisions based on this information to create dynamic web pages.
- Before you process the information, you need to create an HTML form that will send information to PHP script.

- There are two ways for sending data from web page using GET and POST.
- GET and POST are defined in form element's method attribute. Along with this you need to specify action attribute with the name of the file.

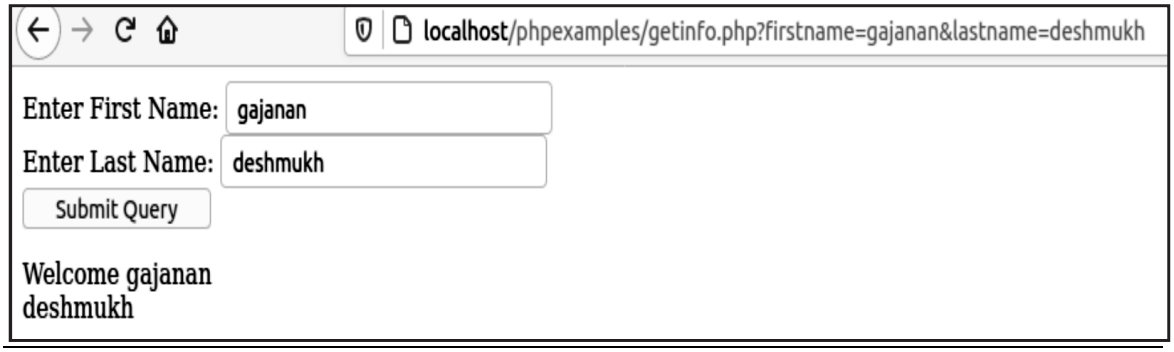
1. $_GET[ ]:

- It is used to retrieve information from the form control through parameter sent in the URL. The page and the encoded information are separated by the ? character.
- It is an associative array of variables passed to the current script via the URL parameters. you can use when there is small amount of data, it is mostly used in pagination, page number is shown in the url and you can easily get the page number from URL using `$_GET`.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- The GET method is restricted to send up to 1024 characters only. GET cannot be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY_STRING environment variable. The PHP provides `$_GET` associative array to access all the sent information using GET method.

Program 2.3: PHP program to demonstrate `$_GET[]` in PHP.

```
<?php
if( $_GET["firstname"] || $_GET["lastname"] )
{
   echo "Welcome ". $_GET['firstname']. "<br />";
   echo $_GET['lastname'];
   exit();
}
?>
<html>
<body>
<form action = "<?php $_PHP_SELF ?>" method = "GET">
Enter First Name: <input type = "text" name = "firstname" /><br>
Enter Last Name: <input type = "text" name = "lastname" /><br>
<input type = "submit" />
</form>
</body>
</html>
```

Output:



2. $_POST[ ]:

- It is used to retrieve the information from the form control through HTTP POST method.
- It is an associative array of variables passed to the current script via the HTTP POST method.
- You can use POST method when you are sending large data to server or if you have sensitive information like passwords, credit card details etc.
- Unlike the GET method, it does not have a limit on the amount of information to be sent. The information sent from an HTML form using the POST method is not visible to anyone.

Program 2.4: PHP program to demonstrate `$_POST[]` method.

```
<?php
if( $_POST["firstname"] || $_POST["lastname"] )
{
   echo "Welcome ". $_POST['firstname']. "<br />";
   echo $_POST['lastname'];
   exit();
}
?>
<html>
<body>
<form action = "<?php $_PHP_SELF ?>" method = "POST">
   Enter First Name: <input type = "text" name = "firstname" /><br>
   Enter Last Name: <input type = "text" name = "lastname" /><br>
<input type = "submit" />
</form>
</body>
</html>
```

Output:

- **Difference between GET and POST Method:**

GET Method	POST Method
1. The GET method sends its variables in the web browsers URL, which makes it easy to see and possibly change the information that was sent.	1. The POST method sends variables securely and it has no limitations on the amount of data to be sent. And variables are not displayed in URL. Also it is not possible to bookmark the page.
2. So this method should not be used when sending passwords or other sensitive information. It also should not be used for any actions that cause a change in the server, such as placing an order or updating a database. However, because the variables are displayed in the URL, it is possible to bookmark the page.	2. The biggest difference between GET and POST is that GET requests are idempotent, GET request for a particular URL, including form parameters, is the same as two or more requests for that URL. Thus, web browsers can cache the response pages for GET requests, because the response page doesn't change regardless of how many times the page is loaded. Because of idempotence.
3. The GET method has limit on the amount of data to be sent. If you send long variables using GET, you are likely to lose large amount of data. Another point is all the variables which you pass are visible in URL separating with ? So GET method is not secure.	3. POST requests are not idempotent. This means that they cannot be cached, and the server is recontacted every time the page is displayed.
4. `Form name="form1" method="GET"`	4. `Form name="form1" method="POST"`

- Aprart from $_GET[] and $_POST[] super global array following methods are also used to process forms.

3. isset():

This function is used to determine whether variable or form control is having value or not.

Program 2.5: PHP program to demonstrate form processing using isset() in PHP :

```
<?php
if (isset($_POST['submit']))
{
   if ((!isset($_POST['ename'])) || (!isset($_POST['age'])) ||
   (!isset($_POST['address'])) || (!isset($_POST['emailaddress'])) ||
   (!isset($_POST['password'])) || (!isset($_POST['gender'])))
   {
      $error = "*" . "Please fill all the required fields";
   }
   else
   {
      $name = $_POST['ename'];
      $age = $_POST['age'];
      $address = $_POST['address'];
      $emailaddress = $_POST['emailaddress'];
      $password = $_POST['password'];
      $gender = $_POST['gender'];
   }
}
?>
<html>
<head>
<title>Employee Form Processing</title>
</head>
<body>
<h1>Employee Form Processing </h1>
<fieldset>
<form id="form1" method="post" action="">
<?php
if (isset($_POST['submit']))
{
   if (isset($error))
```

```
    {
        echo "<p style='color:red;'>"
        . $error . "</p>";
    }
}
?>
Employee Name:
<input type="text" name="ename"/>
<br>
<br>
Employee Age:
<input type="number" name="age"/>
<br>
<br>
Employee Address:
<input type="text" name="address"/>
<br>
<br>
Employee Email:
<input type="email" name="emailaddress"/>
<br>
<br>
Employee Password:
<input type="password" name="password"/>
<br>
<br>
Employee Gender:
<input type="radio"value="Male"
name="gender"> Male
<input type="radio" value="Female"
name="gender">Female
<br>
<br>
<input type="submit" value="Submit" name="submit" />
</form>
</fieldset>
<?php
if(isset($_POST['submit']))
{
```

```
        if(!isset($error))
        {
            echo "name=".$name."<br>";
            echo "age=".$age."<br>";
            echo "address=".$address."<br>";
            echo "email=".$emailaddress."<br>";
            echo "password=".$password."<br>";
            echo "gender".$gender."<br>";
        }
    }
    ?>
    </body>
    </html>
```

Output:

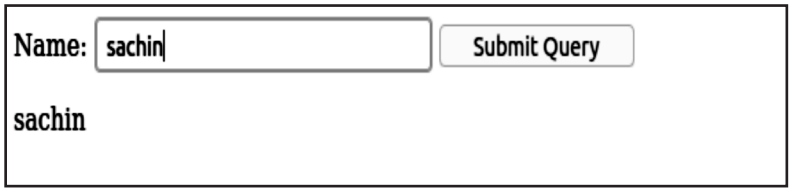
4. $_REQUEST[]:

- PHP $_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.
- It is also used to access information while using database.

Program 2.6: PHP program to demonstrate $_REQUEST[].

```
<html>
<body>
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
Name: <input type="text" name="fname">
<input type="submit">
</form>
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST")
{
   $name = $_REQUEST['fname'];
   if (empty($name))
   {
      echo "Name is empty";
   }
   else
   {
      echo $name;
   }
}
?>
</body>
</html>
```

Output:



2.3.1 Self Processing Pages

- PHP_SELF is a super global variable that returns the current script which is being executed. This variable returns name and path of the current file. You can use this variable in action field of the HTML form.
- The most common use of PHP_SELF variable is in the action field of the form tag <form action="">. Action field instructs where to submit data when user clicks on submit button. Assume you have file name called php_example.php and you want to load the same page after the form submission.
- Then code will be, <form method="post" action="php_example.php">
- we can also use PHP_SELF variable instead of php_example.php, so the code becomes like this,

```
<form name="form1" method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
```

Program 2.7: PHP program to demonstrate PHP_SELF.

```
<html>
<body>
<form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
<input type="text" name="name"><br>
<input type="submit" name="submit" value="Submit Form"><br>
</form>
</body>
</html>
<?php
if(isset($_POST['submit']))
{
   $name = $_POST['name'];
   echo "after submission entered name is: <b> $name </b>";
}
?>
```

Output:

PHP_SELF exploits can be avoided by following functions:

1. **`htmlspecialcharacters()`:** This function converts special characters to HTML entities. It will replace HTML characters like with < and >. It prevents scripting attacks who try to exploit code by inserting HTML or Java script code.
2. **`htmlentities()`:** This function is used to encode HTML entities. Now if the user try's to exploit the `PHP_SELF` variable the attempt will fail.

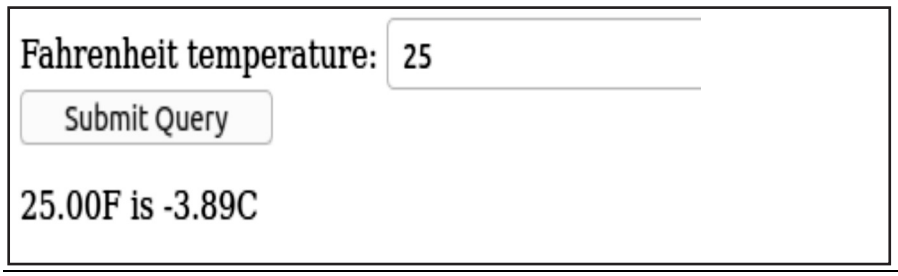
2.4 STICKY FORMS

- When you are working with forms on the web page you may come to know that after submitting the information when you want to go back and correct it you may see an empty form.
- If you are sent back to the form that retains the information which already entered you are supposed to be using sticky forms.
- A sticky form is simply HTML form that remembers how you filled it out. It is nice feature, it reduces efforts of user to reenter data into form.
- Most of the web sites uses sticky form concept in which results of query are accompanied by a search form whose default values are those of previous query.

Program 2.8: Program of converting temperature from Fahrenheit to celcius.

```
<html>
<head><title>Temperature Conversion</title></head>
<body>
<?php
$fahr = $_GET['fahrenheit'];
?>
<form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="GET">
Fahrenheit temperature:
<input type="text" name="fahrenheit" value="<?php echo $fahr ?>" />
<br />
<input type="submit" name="Convert to Celsius!" />
</form>
<?php
if (! is_null($fahr))
{
   $celsius = ($fahr - 32) * 5/9;
   printf("%.2fF is %.2fC", $fahr, $celsius);
}
?>
</body>
</html>
```

Output:



- In the above example, sticky form is used with value="<?php echo $fahr ?>" here after submitting the form data the input value will remain visible onto form.

2.4.1 Sticky Multivalued Parameters

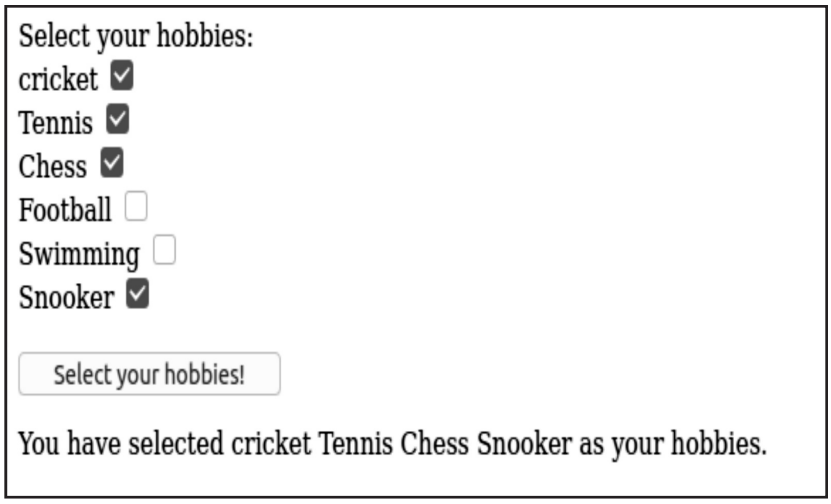
- Now, you may think can we use sticky multiple form elements? The answer is yes, but you need to check whether each possible value in the form was one of the submitted value.

Program 2.9: Php program for multivalued checkbox.

```
<html>
<head><title>Hobbies</title></head>
<body>
<?php
// fetch form values, if any
$attrs = $_GET['attributes'];
if (! is_array($attrs))
{
   $attrs = array( );
}
// create HTML for identically-named checkboxes
functionmake_checkboxes ($name, $query, $options)
{
   foreach ($options as $value => $label)
   {
      printf('%s <input type="checkbox" name="%s[]" value="%s" ',
      $label, $name, $value);
      if (in_array($value, $query))
      {
          echo "checked ";
      }
      echo "/><br />\n";
   }
}
```

```
$hobby_likes = array('cricket'=> 'cricket', 'Tennis'=> 'Tennis',
'Chess' => 'Chess', 'Football' => 'Football',
'Swimming' => 'Swimming', 'Snooker' => 'Snooker'
);
?>
<form action="<?php $_SERVER['PHP_SELF'] ?>" method="GET">
Select your hobbies:<br />
<?phpmake_checkboxes('attributes', $attrs, $hobby_likes); ?>
<br />
<input type="submit" name="s" value="Select your hobbies!" />
</form>
<?php
if (array_key_exists('s', $_GET))
{
   $likes = join (" ", $_GET['attributes']);
   echo "You have selected $likes as your hobbies.";
}
?>
</body>
</html>
```

Output:



- In above program, `make_checkboxes()` function has passed with three arguments which is used to set sticky multivalued checkboxes.

2.5 SETTING RESPONSE HEADERS

- The HTTP response that a server sends back to the client contains header that identify the type of content in the body of the response. The server that sends response, how many bytes are in the body, when the response was sent etc. The Servers Apache and PHP take care of headers, identifying the document as HTML, calculating the length of HTML page, and so on.
- You can set the expiration time of page, redirect the client browser, or to generate the HTTP error, you can use header() function.
- Headers must be written at the top of the file before HTML tags.
- **Content-type:** The content type header identifies the type of the document which will get returned. It is normally text/html but you can specify text/plain also.

```
<?php
header('content-type:text/plain');
?>
```

- **Redirections:** Redirections will send the browser to the new URL.

```
<?php
header('location:"https://www.google.com");
?>
```

- If there is a partial URL like ".../sample.html" the redirection is handled by internally by web server. Mostly we should specify absolute URL.
- **Expiration:** Server explicitly informs browser when document will get expire. Proxy and browser cache hold document until the time or it gets expire.
- Expires header is used to set the expiration time:

```
header( 'Expires:  Mon, 20 Jan 2020 06:00:00GMT');
```

- To expire the web page before five hours of page generation time, use `time()` method and `gmstrftime()` to generate expiration string:

```
$now = time();
$string = gmstrftime("%a, %d %b %Y %H:%M:%S GMT", $now + 60*60*5);
header("Expires: $string");
```

- If you wish that your document never gets expire then use time and `gmstrftime()`:

```
$now = time( );
$string = gmstrftime("%a, %d %b %Y %H:%M:%S GMT", $now + 365*86440);
header("Expires: $string");
```

- This is the best way to prevent a browser or proxy cache from storing your document:

```
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
header("Cache-Control: no-store, no-cache, must-revalidate");
header("Cache-Control: post-check=0, pre-check=0", false);
header("Pragma: no-cache");
```

- The header() function is the built-in function in PHP used to send raw HTTP header. These are functions which manipulates information sent to the client by web browser before any output has been sent.
- The PHP header() function send a HTTP header to client or browser in a raw form, this raw data is sent with request made by server as header information.
- **Syntax:** `void header( $header, $replace=TRUE , $http_response_code)`
- This header function has three parameters `$header` parameter holds string. There are two types of header calls. The first header starts with $HTTP which is used to check HTTP status code to send. Another is location, it is compulsory parameter.
- `$replace` parameter is used to replace previous header or add second header. The default value is true. If `$replace` is false value then it force multiple headers of same type.
- **`$http_response_code`:** It is optional parameter. It forces the HTTP response code to the specified value.

Program 2.10: PHP program for header function.

```
<?php
// PHP program to describes header function
// Redirect the browser
header("Location: https://www.google.com");
// The below code does not get executed
// while redirecting
exit;
?>
```

Output:

This will change location of header, and you will be redirected to URL given.

Program 2.11: PHP program to display header list.

```
<?php
header("Expires: Wed, 11 Mar 2020 10:04:20 GMT");
header("Cache-Control: no-cache");
header("Pragma: no-cache");
?>
<html>
<body>
<p>Code for php header!</p>
<!-- PHP program to display header list -->
<?php
print_r(headers_list());
?>
</body>
</html>
```

Output:

```
Code for php header!
Array ( [0] => Expires: Wed, 11 Mar 2020 10:04:20 GMT [1] => Cache-Control: no-cache [2] => Pragma: no-cache )
```

- The above example helps to prevent caching by sending header information which override browser setting to not-cache.
- Note that only one header is allowed to send at a time to prevent header injection attacks.

HTTP Authentication:

- If you want PHP to protect parts of your web site with password instead of storing in external file you can use `$_SERVER['PHP_AUTH_USER']` and `$_SERVER['PHP_AUTH_PASSWORD']` global variables that contains username and password.
- If you want to deny access to page, you have to send www-Authenticate header which identifies the authentication realm as part of response with status code 401.

```
header('www-Authenticate : Basic realm="website name"');
header('HTTP/1.0 404 Unauthorized Access');
echo "enter valid username and password";
exit;
```

Program 2.12: Basic HTTP Authentication example.

```
<?php
if (!isset($_SERVER['PHP_AUTH_USER']))
{
   header("WWW-Authenticate: Basic realm=\"Private Area\"");
   header("HTTP/1.0 401 Unauthorized");
   // only reached if authentication fails
   print "Sorry - you need valid credentials granted access to the private
   area!\n";
   exit;
}
else
{
   // only reached if authentication succeeds
   print "Welcome to the private area, {$_SERVER['PHP_AUTH_USER']} - you
   used {$_SERVER['PHP_AUTH_PW']} as your password.";
}
?>
```

Output:

Welcome to the private area, gd4184 - you used password as your password.

Summary

- In this chapter we have learnt $_SERVER is an auto global array which contains number of useful elements that provide information on to the web server and current request.
- $_SERVER array contains useful information from the web server. $_SERVER is a super global variable it contains information about headers, path, host and script locations. It has different parameters like PHP_SELF, SERVER_INFO, SERVER_NAME, SERVER_PORT, SERVER_ID etc.
- It is very easy to process forms with php, you can process information collected by HTML form and you can use PHP code to make decisions based on this information to create dynamic web pages.
- $_GET and $_POST are used to process form data. $_GET is not so secure and it is used to send very less data also it is idempotent. Whereas $_POST is secured and you can send huge amount of data with it.
- Generally $_POST is recommended. Apart from it $_REQUEST is also used.
- Sticky form is used to store form information which you have already filled in.
- Self processing page is used with $PHP_SELF variable which is used to pass form data to same page.
- Header information is set with header() function. There are other headers you can use to set header like Content-type, expiration, redirection.
- If you want PHP to protect parts of your web site with password instead of storing in external file you can use $_SERVER['PHP_AUTH_USER'] and $_SERVER['PHP_AUTH_PASSWORD'] global variables that contains username and password.

Check Your Understanding

1. Which of the following PHP variables are used to process form data?
   (a) $_GET & $_PUT (b) $_GET & $_POST
   (c) $_SET & $_GET (d) READ & WRITE
2. When $_GET variable is used to collect form data, the data is visible _____
   (a) to nobody (b) to only one user
   (c) to root user (d) to everyone
3. How many predefined variables are used in php to authenticate use?
   (a) 3 (b) 4
   (c) 2 (d) 1
4. Which of the following PHP variables are use for authentication?
   (i) $_SERVER['PHP_AUTH_USER'];
   (i) $_SERVER['PHP_AUTH_ROOT'];
   (i) $_SERVER['PHP_AUTH_PWD'];
   (i) $_SERVER['PHP_AUTH_PW'];
   (a) (i) and (ii) (b) (i) and (iii)
   (c) (i) and (iv) (d) (ii) and (iii)

5. Which of the following PHP function is used for authentication?
   (a) inspect() (b) intersect()
   (c) header() (d) footer()
6. Which of the following should not be used while sending crucial information?
   (a) $_POST (b) $_GET
   (c) $_PROCESS (d) $_ISSET
7. Which of the following is not $_SERVER parameter?
   (a) SERVER_INFO (b) PHP_SERVER
   (c) SERVER_NAME (d) SERVER_PORT
8. Which is true about $PHP_SELF ?
   (a) It passes php form information to next page.
   (b) It stores web page information.
   (c) It is used to debug page.
   (d) It is used to pass page information to itself.
9. What is correct statement about sticky forms?
   (a) It is simply HTML form that remembers data how you filled it.
   (b) It is used to submit data to another page.
   (c) It is used to pass data to server.
   (d) None of above.
10. What is use of $isset() function?
   (a) It is used to check whether variable is set or not.
   (b) It is used to set variable.
   (c) It is used to set new value.
   (d) All of above.

ANSWERS

1. (b)	2. (d)	3. (c)	4. (c)	5. (c)	6. (b)	7. (b)	8. (d)
9. (a)	10. (a)						

Practice Questions

Q. I Answer the following questions in short.

1. What is $_SERVER variable ?
2. List different HTML form elements.
3. What is use of isset() method ?

4. Why we use `$_REQUEST[]`?
5. Describe following headers:
   (i) Expiration
   (ii) Redirection
   (iii) Location
   (iv) Content-type

Q. II Answer the following questions.

1. Explain different parameters used for `$_SERVER` information?
2. What is sticky form? Explain.
3. How to set response headers? Justify?
4. How to perform form processing php? Explain.
5. What is self processing form? Explain.
6. Differentiate between `$_GET` and `$_POST` variables.
7. Explain what is HTTP authentication?

Q. III Define the terms:

1. forms
2. self processing page
3. sticky form
4. html element
5. GET method
6. POST method

***

3...

XML

Objectives...

- To understand concept of XML.
- To study document structure of XML.
- To understand how PHP works with XML.
- To learn about XML Parser.
- To study Document object model and XML extension.

3.1 INTRODUCTION XML

- XML is extensible Markup Language; it is a standardized data format. XML has similar tags like HTML but XML documents are parsed. XML is used as standard data format in various fields like publishing, engineering and medicine.
- It allows content authors to mark up their data with customized machine readable tags, so that data can be easily classified and searched.
- XML helps to enforce formal structure on content, and it provides a portable format that can be used to easily exchange information between different systems.
- It is basically designed to store and transport data. It was initially designed to be self descriptive. XML tags are not predefined like HTML tags.
- XML is a public standard it was developed by World Wide Web consortium and it's visible as open standard.
- It is a language that helps the document authors to describe the data in the document, by marking it up with custom tags. XML encourages authors to design and create their own tags, so that it increases flexibility and usability.
- It is one of the most popular languages to describe and store structured information on the web.
- XML data is physically stored as a text document. It allows to make it portable because all systems can read and process text files, text editors like notepad can be used to

create XML documents, Mozilla Firefox, internet explorer supports it and they can be used to read and display data.

- Most of the XML documents consist of elements like HTML tags, entities, and regular data.

```
<book isbn="19256245612">
<title>Advance PHP </title>
<authors>
<author>swatijadhav</author>
<author>gajanandeshmukh</author>
</authors>
</book>
```

- XML requires every tab to be closed. In XML, tags can be nested but can't be overlapped.
- XML also requires that the document begin with processing instructions that identifies the version of XML being used, for example: `<?xml version="1.0">`
- Another requirement of well-formed XML document is that there can be only one element at the top level of the file. Let's take an example,

```
<?xml version=" 1.0">
<library>
<title>programming in java</title>
<title>operating systems</title>
<title>advance php</title>
</library>
```

- Document Type Definition (DTD) and schema. DTD and schema are used to validate the document to ensure that they follow rules for their type of document.

3.2 XML DOCUMENT STRUCTURE

- XML document is made up of different components each serving a specific purpose. Each XML document must have root element which is the starting of XML document which contains all other elements.

```
<root>
<section>
<sub-section></sub-section>
<sub-section></sub-section>
<section>
```

```
<section>
<sub-section></sub-section>
<sub-section></sub-section>
<section>
</root>
```

- XML document must begin with a prolog that appears before the root element. It contains metadata about the element, like character encoding, document structure and style sheets.

```
<?xml version="1.0" encoding="UTF-8"?>
```

- XML tags are case sensitive and they should begin with opening tag < > and must end with < / > tag.

  e.g. `<name></name>`

- An element in XML is formed by characters between the start tag and end tag.

The major portion of XML document includes the following:

1. **XML Prolog:**

- The Prolog contains optional information such as the XML version the document conforms to, information about the character encoding used to encode the contents of the document, and a reference to either a Document Type Definition (DTD) or XML Schema document which describes the grammar and vocabulary of the document.
- The XML Schema document is the more modern way to describe XML grammar and vocabulary.
- XML Schemas and DTDs are usually stored in external documents and the prolog can reference both XML Schemas and DTDs.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE maillist SYSTEM "maillist.dtd">
```

2. **XML Body:**

- The body contains a single top-level element called the root element, which contains all other elements and other markup information.
- Following simple example explains the prolog and body, where `addresses` is root element and encloses entire body, of XML document.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE maillist SYSTEM "maillist.dtd">
<addresses>
   <address></address>
   <address></address>
   <address></address>
</addresses>
```

3. DTD and XML Schema Documents:

- DTD and XML Schema documents are rules that define the elements that can exist in a particular document or group of documents, and the relationships among the various elements.
- A DTD or XML Schema can be part of the content of an XML document or can be separate from it and referred to by the XML documents.
- Best practice calls for the DTD and XML Schema documents to be separate from the XML content for reuse and maintainability.
- Here is an example of a DTD:

```
<?xml encoding="US-ASCII"?>
<!-- DTD for a an XML document that stores Customer names and numbers-->
<!ELEMENT customer(name, cust-num)>
<!ELEMENT name(#PCDATA)>
<!ELEMENT cust-num(#PCDATA)>
```

- An example of XML Schema:

```
<?xml version="1.0"?>
<!-- XML Schema document for a an XML document that stores Customer names
and numbers-->
<xsd:schemaxmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="">
<xsd:element name="customer" minoccurs="0" maxoccurs="unbounded">
<xsd:complextype>
<xsd:sequence>
<xsd:element name="custnum" type="xsd:int"/>
<xsd:element name="Name" type="xsd:string"/
</xsd:sequence>
</xsd:complextype>
</xsd:element>
</xsd:schema>
```

4. XML Elements:

- Elements represent the logical components of documents. They can contain data or other elements. For example, a customer element can contain a number of column (field) elements and each column element can contain one data value.
- An element is composed of a start tag, data, and an end tag. Here is an example of an element:
  `<cust-name>suresh</cust-name>`, here we have starting tag, data and ending tag.
- Let's take look at example containing other element's:

```
<phone>
   <entry>
      <name>suresh</name>
      <extension>120</extension>
   </entry>
</phone>
```

- In this example, name and extension are child elements of entry, which is a child element of phone. Similarly, phone is a parent element of entry, which is a parent element of name and extension.
- The top-level element, which is parent of all elements, is referred as root element.
- A child element that can't have its own child elements as defined by DTD or XML schema is referred as leaf element.
- A typical XML elements that describe hierarchy uses following terms:
  - Root element
  - Parent element
  - Child element
  - Leaf element

5. XML Attributes:

- Elements can have additional information called attributes attached to them. Attributes describe properties of elements. Here is an example of an element with an attribute, emp-num:

```
<employee id="1">
<name>suresh</name>
</employee>
```

6. XML empty elements:

- An empty element is an element that is complete by itself; it never contains other elements. Rather than being composed of a start tag, data, and an end tag, the empty element is a combined start and end tag.
- Typically, an empty element is used as a flag of some kind or perhaps as an element that contains its information in its attributes. In this example, extension has been converted to an empty tag:

```
<phone>
<entry>
<name>suresh</name>
<extension number="120"/>
</entry>
</phone>
```

7. Comments:

- Comments can be placed anywhere in a document and they are not considered to be a part of textual content of XML document.
- **Syntax**: `<!-- comment tag -->`

3.3 PHP AND XML

- Native XML support in PHP4 was limited to some specific technologies; new version of PHP5 has introduced several new features and enhancements to PHP language.

Core XML Extensions:

1. Tree based parsers:

- Tree based parsers allow you to construct or load existing XML documents so you can navigate or modify them. To achieve this, entire XML document is created or loaded into memory as tree. But the entire document must reside in memory.

2. DOM Extension:

- The DOM extension is the PHP5 replacement of `domxml`, which is now supported only under PHP4.
- DOM extension was created to address shortcomings of `domxml` while adhering to W3C DOM specifications; DOM extension has large and complex API.
- The DOM extension allows you to access all node types, allows you to create and modify complex documents, and gives you navigation and functionality.

3. SimpleXML Extension:

- Using new functionality offered by PHP5, `SimpleXML` provides an extremely simple and lightweight tool to manipulate XML documents. Compared to the DOM extension, `SimpleXML` has easy to learn API because you can view the document as tree of objects, where objects are synonymous with element nodes.
- Accessing child elements is simple, you can access attributes similarly just like how access an array. To some extent, `SimpleXML` allows content editing.
- To create a `SimpleXML` object from an XML document stored in a string, pass the string to `simplexml_load_string()`. It returns a `SimpleXML` object. `simplexml_load_file()` this function is used to load external XML file.

Program 3.1: PHP program to read the data from XML file using PHP code.

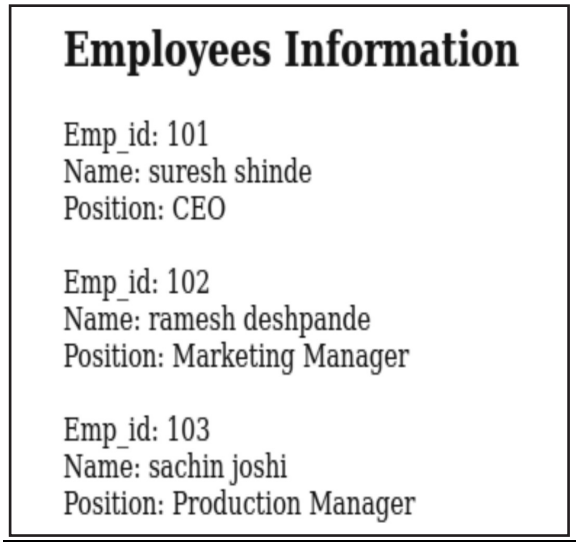
Employee.xml

```
<?xml version="1.0" encoding="utf-8"?>
<employees dept = "civil">
   <record emp_id = "101">
      <name>sureshshinde</name>
      <position>CEO</position>
   </record>
   <record emp_id = "102">
      <name>rameshdeshpande</name>
      <position>Marketing Manager</position>
   </record>
   <record emp_id = "103">
      <name>sachinjoshi</name>
      <position>Production Manager</position>
   </record>
</employees>
```

```
<?php
$xml = simplexml_load_file('Employee.xml');
echo '<h2>Employees Information</h2>';
$list = $xml->record;
for ($i = 0; $i < count($list); $i++)
```

```
{
   echo 'Emp_id: ' . $list[$i]->attributes()->emp_id . '<br>';
   echo 'Name: ' . $list[$i]->name . '<br>';
   echo 'Position: ' . $list[$i]->position . '<br><br>';
}
?>
```

Output:



- Here we have used `simplexml_load_file()` function to load the XML file and assigns the content to the array variable $xml.
- `"$list = $xml->record;"` gets the contents of the record node.
- `"for ($i = 0; $i < count($list); $i++)"` is the for loop that reads the numeric array and outputs the results.
- `"$list[$i]->attributes()->emp_id;"` reads the `emp_id` attribute of the element.
- `"$list[$i]->name;"` reads the value of the name child element.
- `"$list[$i]->position;"` reads the value of the position child element.

Program 3.2: PHP program for `simplexml_load_string()` function, it returns simpleXML object.

```
<html>
<body>
<?php
$note=<<<XML
<note>
<to>Abhay Deshmukh</to>
<from>Gajanan Deshmukh</from>
<heading>Greetings of the day</heading>
<body>Hello How r u? </body>
```

```
</note>
XML;
$xml=simplexml_load_string($note);
print_r($xml);
?>
</body>
</html>
```

Output:

SimpleXML Element Object ([to] => Abhay Deshmukh [from] => Gajanan Deshmukh [heading] => Greetings of the day [body] => Hello How r u?)

3.3.1 Generating XML Document

- `SimpleXML` is an better option for parsing existing XML Documents, but you can't use to create new documents.
- The simplest way to generate an XML document is to build a PHP array whose structure mirrors that of the XML document and then to iterate through the array, printing each element with appropriate formatting.

Program 3.3: PHP program for generating XML Document.

```
<?php
$student = array('Student_Name'=> 'Santosh Mishra',
'Student_class'=> 'SYBBACA',
'Subject_Name' => 'Advance PHP');
print "<student>\n";
foreach ($student as $element => $content) {
print " <$element>";
print htmlentities($content);
print "</$element>\n";
}
print "</student>";
?>
```

Output:

Santosh Mishra SYBBACA Advance PHP.

3.4 XML PARSER

- XML parser is software library or package that provides interface for client applications to work with XML documents.
- XML parser is an event based parser, it works like an event handler. XML parser is designed to read XML and create way of programs to use XML. It validates documents and check the document is well formed.

- Typically XML document is processed by software known as XML parser. It reads XML documents using one of the two approaches, the simple API for XML (SAX) approach or the document object model approach.
- A SAX parser works by traversing XML document sequentially, from starting to end of document, and calling specific user defined functions as it encounters different types of XML constructs.
- It does not create any internal structure. Clients does not know what methods to call, they just override the methods of API and place his own code inside the method.
- PHP's XML parser is based on Expat C Library which allows you to parse but not to validate XML Documents.
- It means that you can check which XML tags are present but you can't validate whether they are in the right structure.

PHP XML parser extension includes several functions to access XML elements:

1. **`xml_parser_create()`:** It is used to create XML parser handler.
2. **`xml_parser_create_ns()`:** This function creates parser handler with namespace support.
3. **`xml_parse()`:** It is used to parse XML document.
4. **`xml_parse_into_struct()`:** It is used to covert XML nodes into array.
5. **`xml_set_element_handler()`:** To set start and end element handlers for XML parser. There are more setters to set variety of handlers.
6. **`xml_get_current_line_number()`, `xml_get_current_column_number()`:** To get current column line number and column number respectively.
7. **`xml_parser_free()`:**To cancel parser handler reference if it is not required.
8. **`Handler_functions:`** In XML Parser extension, handler functions are defined to invoke on a particular event. For example, start and end element handlers of this parser are invoked on element start and end respectively. For XML parser a start element handler must contain parameters like parser handler, element name, and its attribute array. End element handler must contain parser handler and element name. After definition, we need to set these element handlers to XML parser by using **`xml_parser_element_handler()`.**

3.4.1 XML DOM Parser

- The DOM extension allows you to access all node types, allows you to create and modify complex documents, and gives you advanced navigation and functionality.
- An advantage to this extension, if you are coming from another language that incorporates a DOM compliant parser, is that the API should already be familiar to you and easy to begin using under PHP.
- This parser makes it possible to process XML documents in PHP.

Program 3.4: PHP program to demonstrate DOM Document.

employee.xml

```
<?xml version="1.0" encoding="utf-8"?>
<employees dept = "civil">
<record emp_id = "101">
<name>suresh shinde</name>
<position>CEO</position>
<salary>50000</salary>
</record>
<record emp_id = "102">
<name>ramesh deshpande</name>
<position>Marketing Manager</position>
<salary>20000</salary>
</record>
<record emp_id = "103">
<name>sachin joshi</name>
<position>Production Manager</position>
<salary>40000</salary>
</record>
</employees>
```

employee.php

```
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("employee.xml");
print $xmlDoc->saveXML();
?>
```

Output:

suresh shinde CEO 50000 ramesh deshpande Marketing Manager 20000 sachin joshi Production Manager 40000

- The above example creates `DOMDocument object` and loads the XML from `employee.xml` into it.

- The `saveXML()` function puts the internal XML document into a string, so we can output it.
- Looping through XML: Now we want to initialize XML parser, load the XML document, and iterate though the all elements of employee.

Program 3.5: Php code to iterate though `employee.xml`.

employee.xml

```
<?xml version="1.0" encoding="utf-8"?>
<employees dept = "civil">
<record emp_id = "101">
<name>suresh shinde</name>
<position>CEO</position>
<salary>50000</salary>
</record>
<record emp_id = "102">
<name>ramesh deshpande</name>
<position>Marketing Manager</position>
<salary>20000</salary>
</record>
<record emp_id = "103">
<name>sachin joshi</name>
<position>Production Manager</position>
<salary>40000</salary>
</record>
</employees>
```

Employee.php

```
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("employee.xml");
$x = $xmlDoc->documentElement;
foreach ($x->childNodes AS $item)
{
   print $item->nodeName . " = " . $item->nodeValue . "<br>";
}
?>
```

Output:

```
#text =
record = suresh shinde CEO
#text =
record = ramesh deshpande Marketing Manager
#text =
record = sachin joshi Production Manager
#text =
```

3.5 THE DOCUMENT OBJECT MODEL

- The XML Document Object Model (DOM) class is an memory representation of XML Document. It defines a standard for accessing and manipulating documents.
- It allows you to programmatically read, manipulate, and modify XML documents.
- All XML elements can be accessed through XML DOM.
- It is a standard object model for XML, also a standard programming interface for XML.
- It is a World Wide Web Consortium (W3C) standard.
- According to the DOM everything in an XML document is a node. It means entire document is document node.
- Every XML element is an element node. The Text in the XML elements is text nodes. Attributes are attribute nodes. And Comments are comment node.
- In simple words, we can say XML DOM is a standard for how to get, change, add and delete XML elements.
- Under the DOM, the document is manipulated as tree broken down in nodes. It means that whole document is loaded or it is built in memory, where tree is broken down into smaller units all derived from node. Nodes are primary data type, and all other node types are derived from node.
- **Look at the following XML code of Book.xml file; we will represent hierarchical structure of it.**

Book.xml

```
<?xml version="1.0"?>
<bookstore>
<book category="programming">
<title lang="en">Java programming</title>
<author>Steven holzner</author>
<year>2005</year>
<price>500.00</price>
</book>
<book category="CHILDREN">
<title lang="en">Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>450</price>
</book>
<book category="literature">
<title lang="en">Mine kamph</title>
<author>Adolf hitler</author>
<year>1950</year>
<price>525.00</price>
</book>
</bookstore>
```

- Following tree diagram illustrates how memory is structured when XML data is read into DOM structure.

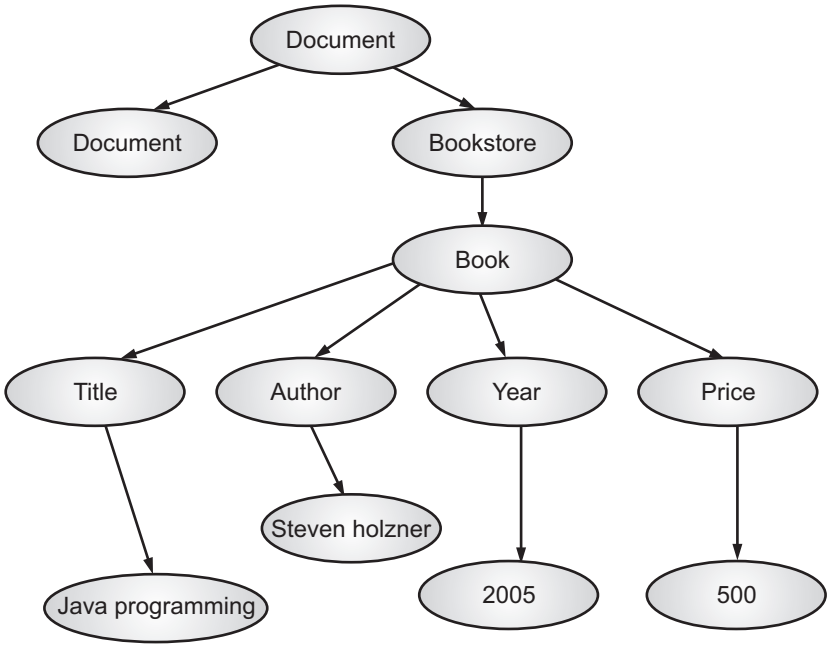


Fig. 3.1: Book.xml file tree structure

XML DOM Structure:

- Within XML document structure, each circle represents node, which is called an `XmlNode` object, The `XmlNode` object is the basic object in the DOM tree.
- The `XmlDocument` class, which extends `XmlNode`, supports methods for performing operations on document as whole.
- In addition, `XmlDocument` provides a means to view and manipulate the nodes in the entire XML `document. BothXmlNode` and `XmlDocument` have performance and usability enhancements and methods and properties to:
  - Access and modify nodes specific to the DOM, such as element node, entity reference node, and so on.
  - Retrieve entity nodes, in addition to the information the node contains, such as text in element node.
- Node objects have a set of methods and properties, as well as basic and well defined characteristics. Some of these characteristics are:
  - Nodes have a single parent node, a parent node being a node directly above them. The only nodes that do not have a parent is the Document root, as it is the top-level node and contains the document itself and document fragments.
  - Most nodes can have multiple child nodes, which are nodes directly below them.

- The following is a list of node types that can have child nodes.
  - **Document Node**: The node at the every top of the tree, Document is special, it is not an element, it does not correspond to anything in XML, but it represents entire document.
  - **DocumentFragment Node:** A Document Fragment is a node used for holding a part of a document, such as buffer for copy and paste. It does not need to meet all of the well formedness contraings – for example, it could contain multiple top level elements.
  - **EntityReference:** An entity reference is an alternative name for a series of characters. You can use an entity in the &name; format, where name is the name of the entity. There are some predefined entities in XML, furthermore you can declare entities in a DTD (Document Type Definition).
  - **Element Node:** An element node in a DOM tree represents a single XML element and its contents.
  - **Attribute:** Attr nodes each represent a single attribute with its name, value and possibly type information. They are normally only found hidden inside element nodes, and you have to use a method such as getAttributeNode(name) to retrieve them.
  - **Text Node:** A text node is the textual content of an element.

- **Following are the XML DOM properties:**

  `x.nodeName:` It is used to get the name of x.

  `x.nodeValue:` It is used to get the value of x.

  `x.parentNode:` It is used to get the parent of node of x.

  `x.childNode:` It is used to get the child of node x.

  `x.attributes:` It is used to get the attribute node of x.

- **XML DOM Methods:**

  `x.getElementsByTagName(name):` It is used to get all the elements with a specific tag name.

  `x.appendChild(node):` It is used to insert a child node to x.

  `x.removeChild(node):` It is used to remove a child node from x, where x is a node object.

3.5.1 Creating an XML Document using the DOM

- For complex operations with PHP we use DOM extension, this extension is enables by PHP5, it provides a sophisticated toolkit that complies with DOM Level 3 standard and provides comprehensive parsing capabilities to PHP.
- To create a XML using `DOMDocument`, basically, we need to create all the tags and attributes using the `createElement()` and `createAttribute()` methods and then create the XML structure with the `appendChild()`.
- In PHP, at first let's create an instance of `DomDocument` and initialize it, and set its version and character encoding.

```
$dom=new DomDocument('1.0','utf-8');
```

- To create a node following method is used,
  `$dom->createElement('books');`
- To set a node as child node of another node we use following statement.
  `$dom->appendChild('previously created node');`

Program 3.6: PHP program to create XML document using DOM.

```
<?php
$dom = new DOMDocument('1.0', 'utf-8');
$dom->preserveWhiteSpace = false;
$dom->formatOutput = true;

//create the main tags, without values
$books = $dom->createElement('books');
$book_1 = $dom->createElement('book');

// create some tags with values
$name_1 = $dom->createElement('name', 'Advance PHP');
$price_1 = $dom->createElement('price', 'Rs550');
$id_1 = $dom->createElement('id', '101');

//create and append an attribute
$attr_1 = $dom->createAttribute('version');
$attr_1->value = '1.0';
//append the attribute
$id_1->appendChild($attr_1);

//$id->removeChild($att_1); it will remove child

//create the XML structure
$books->appendChild($book_1);
$book_1->appendChild($name_1);
$book_1->appendChild($price_1);
$book_1->appendChild($id_1);
$dom->appendChild($books);

//saveXML() method returns the XML in a String
print_r ($dom->saveXML());
?>
```

Output:

Advance PHP Rs550 101

Program 3.7: PHP program to demonstrate `getElementsByTagName()` method.

```
<?php
$xml = <<< XML
<?xml version="1.0" encoding="utf-8"?>
<books>
<book>Programming in Java</book>
<book>Advance PHP</book>
<book>Software Engineering</book>
</books>
XML;
$dom = new DOMDocument;
$dom->loadXML($xml);
$books = $dom->getElementsByTagName('book');
foreach ($books as $book) {
echo $book->nodeValue, PHP_EOL;
}
?>
```

Output:

Programming in Java Advance PHP Software Engineering.

3.5.2 Working with Elements

- The DOM parser works by reading an XML document and creating objects to represent the different parts of document. Each of these object comes with specific methods and properties, which can be used to manipulate and access information about it.
- So entire document is represented as tree of these objects, with DOM parser providing a simple API to move between the different branches of the tree.

Program 3.8: PHP code to access elements of XML file using DOM.

address2.xml

```
<?xml version='1.0'?>
<address>
<street>Flat No A 201 Karve Road</street>
<county>India</county>
<city>
<name>Pune</name>
<zip>411001</zip>
</city>
<country>India</country>
</address>
```

address2.php

```
<?PHP
// initialize new DOMDocument
$doc = new DOMDocument();
// disable whitespace-only text nodes
$doc->preserveWhiteSpace = false;
// read XML file
$doc->load('address2.xml');
// get root element
$root = $doc->firstChild;
// get text node 'India
echo "Country: " . $root->childNodes->item(3)->nodeValue . "\n";
// get text node 'Pune'
echo "City: " . $root->childNodes->item(2)->childNodes->
item(0)->nodeValue . "\n";
// get text node 'Postal Code'
echo "Postal code: " . $root->childNodes->item(2)->childNodes->
item(1)->nodeValue . "\n";
?>
```

Output:

Country: India City: Pune Postal code: 411001.

- With PHP's DOM extension, we begin with initializing an instance of DOM Document object, which represents an XML document. Once object is initialized, is used to parse XML file through load() function, which accepts the path of XML file.
- The result of the load() method is a tree containing `DOMNode` objects, with every object exposing various properties and methods for accessing its parent, child, and sibling nodes.
- For example, every `DOMNode` object exposes a `parentNode` property, which can be used to access its parent node, and a `childNodes` property, which returns a collection of its child nodes. In a similar vein, every `DOMNode` object also exposes `nodeName` and `nodeValue` properties, which can be used to access the node's name and value respectively. It's thus quite easy to navigate from node to node of the tree, retrieving node values at each stage.
- To illustrate the process, consider the preceding script carefully. Once the XML document has been loaded, it calls the `DOMDocument` object's `firstChild` property, which returns a `DOMNode` object representing the root element <address>.
- This `DOMNode` object, in turn, has a `childNodes` property, which returns a collection of all the child elements of <address>. Individual elements of this collection can be accessed via their index position using the item() method, with indexing starting from zero. These elements are again represented as `DOMNode` objects; as such, their

names and values are therefore accessible via their `nodeName` and `nodeValue` properties.

- The element `<name>`, which is the first child of the `<city>` element, is accessible via the path `$root->childNodes->item(2)->childNodes->item(0)`, and the text value 'Oxford' is accessible via the path `$root->childNodes->item(2)->childNodes->item(0)->nodeValue`.

3.6 THE SIMPLE XML EXTENSION

- PHP supports for both Document Object Model and SAX parsing methods. The easiest way to work with XML data in PHP is through its Simple XML Extension. It was enabled by default in PHP5, it provides a user friendly and intuitive interface to reading and processing XML documents.
- Simple XML represents every XML document as an object and turns the element within it into hierarchical set of objects and object properties. Accessing elements becomes simple as parent→child notation to traverse the object tree until that element is reached.
- `SimpleXML` is an extension that allows us to easily manipulate and get XML data. `SimpleXML` provides an easy way of getting an element's name, attributes and textual content if you know the XML document's structure or layout.
- `SimpleXML` turns an XML document into a data structure you can iterate through like a collection of arrays and objects.

`SimpleXMLElement` methods that you can use to manipulate XML documents:

1. **`addAttribute(name, value):`** Adds an attribute named name, with the value of value, to the element.
2. **`addChild(name [, value]):`** Adds a child element called name to the element. The child element can be empty, or it can contain the text value. It returns the child element as a new `SimpleXMLElement` object.
3. **`asXML([filename]):`** Generates an XML document from the `SimpleXMLElement` object. If filename is supplied, it writes the XML to the file; otherwise it returns the XML as a string.
4. **`attributes():`** Returns an associative array of all the attributes in the element, as name= > value pairs.
5. **`children():`** Returns an array of all the element's children, as `SimpleXMLElement` objects.
6. **`getName():`** Returns the name of the element as a string.
7. **`xpath(path):`** Finds child elements that match the given XPath (XML Path Language) path string.

`SimpleXML` gives you three functions that you can use to import XML data into a `SimpleXMLElement` object:

1. **`simplexml_import_dom(node):`** Converts the supplied DOM node, into a `SimpleXMLElement` object.

2. **simplexml_load_file(filename):** Loads the XML file with name filename as a SimpleXMLElement object.
3. **simplexml_load_string(string):** Loads the supplied XML string as a SimpleXMLElement object.

3.6.1 Reading an XML Elements

Reading XML elements using simpleXML:

- You can use simplexml_load_file function to load external XML file in PHP program and create an object, then you can access any element from XML by this object as shown in below.

Program 3.9: Program to illustrate simple XML_extension.

Address.xml

```
<?xml version='1.0'?>
<address>
<Flat_No>A 202</Flat_No>
<Street>KARVE ROAD</Street>
<region>
<name>PUNE</name>
<zip>411020</zip>
<state>Maharashtra</state>
<country>INDIA</country>
</region>
</address>
```

address.php

```
<?php
// to load XML file
$xml = simplexml_load_file('/var/www/html/phpexamples/Address.xml') or die
   ("Unable to load XML!");
echo "City Name: " . $xml->region->name . "<br>";
echo "ZIP Code: " . $xml->region->zip . "<br>";
?>
```

Output:

City Name: PUNE

ZIP Code: 411020

- In the above example, we have read XML file address.xml with simple_xml_load_file() function. This function reads and parse XML file and assuming it is well formed return Simple XML object representing document root element. This object is top level of hierarchical object tree that mirrors internal structure of XML data. The elements below the root are represented with parent-> child relationship.

- While working with attributes, Simple XML has easiest way to get access to attributes. These are converted into keys and values of a PHP associative array and they can be accessed like an array elements.

3.6.2 Reading XML Elements as Loop

- We can use foreach loop to iterate through the entire XML file and read elements from XML. For each loop access all the children of an object.
- We can access the xml file contents using simpexml_load_file() function and assign it to $xml object and then we can pass this $xml object to foreach loop and read all the elements of the file using loop.

Program 3.10: PHP program to demonstrate `simplexml_load_file()` and display contents using for each method.

Book.xml

```
<bookstore>
<book category="programming">
<title lang="en">Java programming</title>
<author>Steven holzner</author>
<year>2005</year>
<price>500.00</price>
</book>
<book category="CHILDREN">
<title lang="en">Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>450</price>
</book>
<book category="literature">
<title lang="en">Mine kamph</title>
<author>Adolf hitler</author>
<year>1950</year>
<price>525.00</price>
</book>
</bookstore>
```

book.php

```
<?php
// load XML file
$xml = simplexml_load_file('/var/www/html/phpexamples/Book.xml') or die
   ("Unable to load XML!");
foreach ($xml->book as $b) {
echo "Book Title=".$b->title . "<br>";
echo "Book Author=".$b->author . "<br>";
echo "Book year=".$b->year . "<br>";
echo "Book price=".$b->price . "<br>";
}
?>
```

Output:

```
Book Title=Java programming
Book Author=Steven holzner
Book year=2005
Book price=500.00
Book Title=Harry Potter
Book Author=J K. Rowling
Book year=2005
Book price=450
Book Title=Mine kamph
Book Author=Adolf hitler
Book year=1950
Book price=525.00
```

- In above program, for each loop iterates over book elements in XML data turning each into object, attributes of book element are represented as elements of associative array.

3.6.3 Creating an XML Document with SimpleXML

Creating New XML Documents:

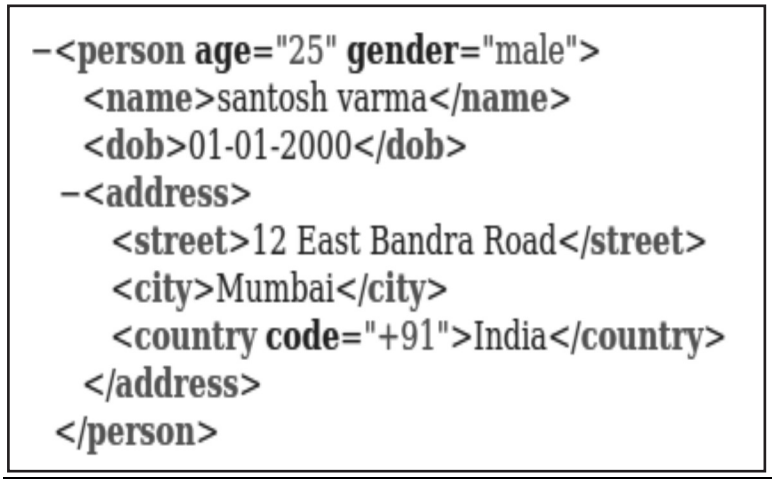
- We can also use `SimpleXML` to create new XML elements from scratch, by initializing an empty `SimpleXML` object from an XML String, and then using `addchild()` and `addAttribute()` methods to build the rest of the XML document tree.

Program 3.11: PHP code for creating XML document with SimpleXML.

```
<?php
// load XML from string
$xmlStr = "<?xml version='1.0'?><person></person>";
$xml = simplexml_load_string($xmlStr);
// add attributes
$xml->addAttribute('age', '25');
$xml->addAttribute('gender', 'male');
// add child elements
$xml->addChild('name', 'santoshvarma');
$xml->addChild('dob', '01-01-2000');
// add second level of child elements
$address = $xml->addChild('address');
$address->addChild('street', '12 East Bandra Road');
$address->addChild('city', 'Mumbai');
// add third level of child elements
```

```
$country = $address->addChild('country', 'India');
$country->addAttribute('code', '+91');
// output new XML string
header('Content-Type: text/xml');
echo $xml->asXML();
?>
```

Output:



- This script begins by initializing a string variable to hold the XML document prolog and root element. The `simple_xml_load_string()` method take care of converting string into `SimpleXML` object representing the document's root element.
- Once this object has been initialized, it's simple to add child elements and attributes to it, and to build the rest of the XML document tree programmatically.

3.7 CHANGING A VALUE WITH SIMPLE XML

- With the Simple XML we can easily change the content in an XML file, we have to simply assign a new value to the corresponding object property using = operator.

Program 3.12: PHP program for changing the values in XML.

contactinfo.xml

```
<?xml version="1.0"?>
<bookstore>
<book category="programming">
<title lang="en">Java programming</title>
<author>Steven holzner</author>
<year>2005</year>
<price>500.00</price>
</book>
<book category="CHILDREN">
<title lang="en">Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
```

```
<price>450</price>
</book>
<book category="literature">
<title lang="en">Mine kamph</title>
<author>Adolf hitler</author>
<year>1950</year>
<price>525.00</price>
</book>
</bookstore>
```

simplexmlexample.PHP

```
<?php
// load XML file
$xml = simplexml_load_file('/var/www/html/phpexamples/contactinfo.xml') or
   die ("Unable to load XML!");
$xml->book[1]->title = 'C++ programming';
$xml->book[1]->author = 'E Balguruswamy';
// output new XML string
header('Content-Type: text/xml');
echo $xml->asXML();
?>
```

Output:

```
-<bookstore>
 -<book category="programming">
    <title lang="en">Java programming</title>
    <author>Steven holzner</author>
    <year>2005</year>
    <price>500.00</price>
  </book>
 -<book category="CHILDREN">
    <title lang="en">C++ programming</title>
    <author>E Balguruswamy</author>
    <year>2005</year>
    <price>450</price>
  </book>
 -<book category="literature">
    <title lang="en">Mine kamph</title>
    <author>Adolf hitler</author>
    <year>1950</year>
    <price>525.00</price>
  </book>
</bookstore>
```

- As you can see in output, we have assigned new values to the book elements attributes title and author having index value 1. Here we have used as `XML()` method which converts the nested hierarchy of `simpleXML` objects and object properties back into a regular XML string.

3.7.1 Adding new Attributes and Child's to the XML document

- We can add attributes to different elements of XML by using `addAttributes()` function. Also we can add new Childs to elements by using `addchild()` method.

Program 3.13: Php program for adding new attributes and child's.

contactinfo.xml

```
<?xml version="1.0"?>
<bookstore>
<book category="programming">
<title lang="en">Java programming</title>
<author>Steven holzner</author>
<year>2005</year>
<price>500.00</price>
</book>
<book category="CHILDREN">
<title lang="en">Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>450</price>
</book>
<book category="literature">
<title lang="en">Mine kamph</title>
<author>Adolf hitler</author>
<year>1950</year>
<price>525.00</price>
</book>
</bookstore>
```

contactinfo.php

```
<?php
// load XML file
$xml = simplexml_load_file('/var/www/html/phpexamples/contactinfo.xml') or
   die ("Unable to load XML!");
$book = $xml->addChild('book');
// add 'region' attribute
$book->addAttribute('region','India');
// add <title>, <author> and <year> elements
$title = $book->addChild('title', 'C Programming');
$author = $book->addChild('author', 'YashwantKanetkar');
$page = $book->addChild('year', '2000');
// output new XML string
header('Content-Type: text/xml');
echo $xml->asXML();
?>
```

Output:

```
▼<bookstore>
  ▼<book category="programming">
     <title lang="en">Java programming</title>
     <author>Steven holzner</author>
     <year>2005</year>
     <price>500.00</price>
   </book>
  ▼<book category="CHILDREN">
     <title lang="en">Harry Potter</title>
     <author>J K. Rowling</author>
     <year>2005</year>
     <price>450</price>
   </book>
  ▼<book category="literature">
     <title lang="en">Mine kamph</title>
     <author>Adolf hitler</author>
     <year>1950</year>
     <price>525.00</price>
   </book>
  ▼<book region="India">
     <title>C Programming</title>
     <author>YashwantKanetkar</author>
     <year>2000</year>
   </book>
 </bookstore>
```

- Simple XML object exposes an `addChild()` method for adding new child elements and `addAttribute()` method for adding new attributes. Both of these methods accept name and value, generates the corresponding element or attribute, and attach it to parent object with hierarchy.
- As you can see in the above output, new attribute region is added and we have set new values to title author and year.

Summary

- XML it is a markup language used for storing and transporting `data.XML` doesn't depend upon platform and software. XML stands for Extensible Markup Language. You must write valid XML tags which have opening and closing tags. You can use PHP with XML with several methods which are used to generate, read XML data and used to Display XML data.
- XML Document has tree structure, where root element is at top and all the child elements are connected to root element.
- Document Object Model is programming interface for HTML and XML documents. It is used to define logical structure of document. Documents are modeled using objects, and the model includes not only the structure of a document but also the behavior of

a document and the objects of which it is composed of like tag elements with attributes in HTML.

- XML parser is software library or package that provides interface for client applications to work with XML document.
- XML parser is event based parser, it works like an event handler. XML parser is designed to read XML and create way of programs to use XML. It validates document and check the document is well formed.
- SAX Parser and DOM Parser are commonly used to parse XML Documents. SAX is an acronym for Simple API for XML. SAX Parser parses the XML file line by line and triggers events when it encounters opening tag, closing tag or character data in XML file. This is why SAX parser is called an event-based parser.
- DOM is an acronym for Document Object Model. Unlike SAX parser DOM parser loads the complete XML file into memory and creates a tree structure where each node in the tree represents a component of XML file. With DOM parser you can create nodes, remove nodes, change their contents and traverse the node hierarchy.
- `SimpleXML` is an extension that allows us to easily manipulate and get XML data.
- `SimpleXML` provides an easy way of getting an element's name, attributes and textual content if you know the XML document's structure or layout.
- `SimpleXML` turns an XML document into a data structure you can iterate through like a collection of arrays and objects.

Check Your Understanding

1. XML stands for _____.
   (a) Extension modern link (b) extensible markup language
   (c) extra modern language (d) x-markup language
2. Which of the following is correct statement?
   (a) All XML elements must have closing tag.
   (b) All XML documents must have DTD.
   (c) All XML documents must be in lowercase.
   (d) XML opening tag should be in uppercase.
3. XML comments are written as _____.
   (a) <! – !> (b) <!--- ?
   (c) <? – ?> (d) <$----%>
4. Which of the following is valid XML tag?
   (a) <? xml version="1.0"?> (b) <$ xml version=1.2">
   (c) <xml version="1.0"> (d) None of above
5. XML is designed to ___ and store data.
   (a) design (b) verify
   (c) transport (d) none of above
6. Well formed XML document means _____.
   (a) It contains a root element.
   (b) It contains an element.
   (c) It contains one or more elements.
   (d) Must contain one or more elements and root elements must contain all other elements.

7. DTD means?
   (a) Data Type Definition. (b) Data Text Decode.
   (c) Define Text Data. (d) None of above.
8. XML DOM object is____.
   (a) Entity. (b) Entity Reference.
   (c) Comment Reference. (d) Comment Data.
9. XML is case sensitive language.
   (a) true (b) false
   (c) Can't Say
10. Which of the following is XML Parser?
    (a) SAX parser (b) DOM Parser
    (c) CDATA Parser (d) (a) and (b)

ANSWERS

1. (b)	2. (a)	3. (b)	4. (a)	5. (c)	6. (d)	7. (a)	8. (b)
9. (a)	10. (d)						

Practice Questions

Q. I Answer the following questions in short.

1. List types of XML parser.
2. List any two methods used wit SimpleXML.
3. What is meaning of valid and invalid XML tags?
4. List XML elements.
5. What are different node types in DOM?
6. What is DOMDocument()?

Q. II Answer the following questions.

1. Explain Concept of XML?
2. What is Document Object Model in PHP?
3. How to use XML with PHP explain with example?
4. What is XML parser? What are different types of it?
5. What is SimpleXML extension?
6. Write a PHP code to display XML data through SimpleXML.
7. Write a PHP code to generate XML.
8. Write PHP script to create a CD catalog using XML file.
9. Create a XML file which gives details of books available in "ABC Bookstore" from following categories
   (1) Technical, (2) Cooking, (3) YOGA

10. Write a PHP script to generate an XML in the following format

```
<?xml version = "1.0" ?>
<BookStore>
      <Books>
            <PHP>
                  <title>Programming PHP</title>
                  <publication>O'RELLY</publication>
            </PHP>
            <PHP>
                  <title>Beginners PHP</title>
                  <publication>WROX</publication>
            </PHP>
      </Books>
</BookStore>
```

11. Create an application that reads "Book.xml" file into simple XML object. Display attributes and elements(Hint: use simple_xml_load_file() function).
12. Write a script to create "cricket.xml" file with multiple elements as shown below:

```
<CricketTeam>
        <Team country="India">
          <player>____</player>
          <runs>______</runs>
          <wicket>____</wicket>
          </Team>
</CricketTeam>
```

13. Write a script to create "vehicle.xml" file with multiple elements as given below

```
<Vehicle>
      <Type = Two Wheeler>
            <Vehicler Name >-------- </Vehicle Name >
            <Company >--------- </Company>
            <Color>--------</Color>
            <Average>------</Average>
      </Type>
</Vehicle>
```

Also add Type = "Four Wheeler" and its elements

Q. III Define the terms:

1. DTD.
2. XML Schema
3. element
4. attribute

***

4…

AJAX with PHP

Objectives…

- To learn the AJAX basic concepts.
- To study asynchronous and asynchronous communication between Web Client and Web Server.
- To learn how validation is performed using AJAX.
- To understand xml data handling using AJAX and PHP.
- To learn how to connect to database using AJAX and PHP.

4.1 INTRODUCTION AJAX

- AJAX stands for Asynchronous JavaScript and XML. AJAX is a new technique for creating dynamic, faster and interactive web based applications using XML, HTML, CSS, and Java Script.
- AJAX uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic web page content display.
- XML is commonly used format for receiving server data. It also supports plain text format.
- AJAX depends on following technologies to create interactive and dynamic webpages:
  - **JavaScript:** JavaScript function is called when an event occurs in a page.
  - **DOM:** It supports use of API for accessing and manipulating structured documents. It also represents the structure of XML and HTML documents.
  - **CSS:** It allows you to separate presentation style from the content and may be changed programmatically by JavaScript.
  - **XMLHttpRequest:** It is a JavaScript object that performs asynchronous interaction with the server.
- All the available browsers cannot support AJAX . Here is a list of major browsers that support AJAX .
  - Mozilla Firefox 1.0 and above.
  - Netscape version 7.1 and above.
  - Apple Safari 1.2 and above.
  - Microsoft Internet Explorer 5 and above.
  - Konqueror.
  - Opera 7.6 and above.

Synchronous Vs Asynchronous Communication:

- In traditional communication between web client and web server, you fill out form data through web page, hit submit button and get directed to a new page with response received from the server. This type of communication between web client and web server is called as **Synchronous Communication**.
- With AJAX, when you hit submit button, JavaScript will make a request to the server, interpret the results, and update part of the current web page without reloading the complete web page. The user would never know that anything was even transmitted to the server. This type of communication between web client and web server is called as **Asynchronous Communication.**

Features of Asynchronous Communication:

- A user can continue to use the web application while the client program requests information from the server in the background.
- Always button click is not required, mouse events can also be used to send request to the server.
- Data-driven as opposed to page-driven.

Advantage of AJAX :

- It is independent of server technology.
- It uses JavaScript. So it can be used for all browser types
- Using ajax you can develop faster and more interactive web applications.
- Ajax based application use less server bandwidth, because there is no need to reload complete page. Partially current page web can be updated.

AJAX can't be used when:

- Page need to show in a search engine
- Browser does not support JavaScript
- User wants to create secure application

Disadvantages of AJAX:

- AJAX is dependent on JavaScript. If JavaScript is disabled then, AJAX also will not useful.
- AJAX can have problems in Search engines as it uses JavaScript.
- Mostly in AJAX, GET methods is used, so there will be some security issues.
- Debugging is difficult.
- Increases size of the requests.
- The problem with browser back button, when AJAX is used.

4.2 UNDERSTANDING JAVA SCRIPTS FOR AJAX

- The user defined function written in JavaScript is called when an event occurs in a page.

- Use following steps for writing function definition in JavaScript to implement AJAX :
  1. Create an `XMLHttpRequest` Object.
  2. Send a request to a Web Server using `GET` or `POST` method.
  3. Display response received from Web server.

1. Create a `XMLHttpRequest` Object:

- All recent browser's versions have a built-in `XMLHttpRequest` object.
- **Syntax**: `variable = new XMLHttpRequest();`
- Old versions of Internet Explorer (IE5 and IE6) use an ActiveX Object.
- **Syntax**: `variable = new ActiveXObject("Microsoft.XMLHTTP");`
- While creating `XMLHttpRequest` object, check for browser's support as follows:

```
var xmlhttp;
if (window.XMLHttpRequest)
{
   // for recent browser' versions
    xmlhttp = new XMLHttpRequest();
}
else
{
   //for old IE5 and IE6
   xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```

2. Send a request to a Web Server using GET or POST method:

- For sending request to a web server use `open()` and `send()` methods of `XMLHttpRequest` Object.
- **Syntax**: `xmlhttpobject.open(method, file_name/url, boolean_value)`

  Where,

  `method` can be GET or POST.

  `file_name/url` which actually handles the request and process it.

  `boolean_value` can be True for asynchronous request or False for synchronous request
- The `open()` method prepares `XMLHttpRequest` object and `send()` method is used to send the request to web server.

GET Method:

- A GET method is used to send a request to a web server.
- **Example**:

```
xmlhttp.open("GET", "demo.php", true);
xmlhttp.send();
```

- If you want to send information or form data with the GET method, add it to the URL:
- **Example**:
```
xmlhttp.open("GET", "demo.php?gamename=cricket", true);
xmlhttp.send();
```
- If you want to send two or more form elements then use ampersand (&) sign between the two elements.
- **Example**:
```
xmlhttp.open("GET", "demo.php?gamename=Cricket&player=Sachin", true);
xmlhttp.send();
```

POST Method:

- A POST method is used to send a request to a web server.
- **Example**:
```
xmlhttp.open("POST", "demo.php", true);
xmlhttp.send();
```
- If you want to send information or form data, add an HTTP header with *setRequestHeader*() and specify the data you want to send in the send() method:
- **Example**:
```
xmlhttp.open("POST", "demo.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");
xmlhttp.send("gamename=Cricket&player=Sachin");
```
- The `setRequestHeader()` method **syntax**: `setRequestHeader(header, value);`

The File name or url - A File on a Server

- This parameter of the open() method, is an address to a file on a server which is to be invoked.
- Example: `xmlhttp.open("GET", "demo.php", true);`
- The file can be any file like .txt and .xml. You can also specify .php file which will perform processing at server side before sending response to the browser.

Boolean Value – true

- The `open()` method contains last parameter as true indicates request send to the server is asynchronous.
- When using Boolean_value=true, specify a function to execute when the response is ready in the `onreadystatechange` event:
- Example:

```
xmlhttp.onreadystatechange = function()
{
   if (xmlhttp.readyState == 4 && xmlhttp.status == 200)
   {
       document.getElementById("demo").innerHTML = xmlhttp.responseText;
   }
};
xmlhttp.open("GET", "student.txt", true);
xmlhttp.send();
```

Boolean Value – false

- The open() method contains last parameter as False indicates request send to the server is synchronous. JavaScript will not continue to execute, until the server response is ready. If the server is busy or slow, the application will hang or stop. No further operation can be performed. No need to write `onreadystatechange` function.
- Example:

```
xmlhttp.open("GET", "student.txt", false);
xmlhttp.send();
document.getElementById("demo").innerHTML = xmlhttp.responseText;
```

3. Display response received from Web server

- To get the response from the web server, use the *`responseText`* or *`responseXML`* property of the *`XMLHttpRequest`* object.
- **responseText Property:** To get the response from the server as a string, use responseText property of XMLHttpRequest object.

  Example:

```
document.getElementById("demo").innerHTML = xmlhttp.responseText;
```

- **responseXML Property:** To get the response from the server as XML file, use responseXML property of XMLHttpRequest object.

  Example:

```
document.getElementById("demo").innerHTML = xmlhttp.responseHTML;
```

- **The `onreadystatechange` event:**
  - When a request is sent to the server, you can perform some actions based on the response.
  - The `onreadystatechange` event is triggered every time the `readyState` changes.
  - The `readyState` property holds the status of the `XMLHttpRequest` object.
  - The `onreadystatechange` property stores a `callback` function to be invoked automatically each time the `readyState` property changes.
  - The `readyState` property holds current status of `XMLHttpRequest` object. The possible values are as follows:

Value	Description
0	Request not initialized
1	Server connection established
2	Request received
3	Processing request
4	Request finished and response is ready

- The `onreadystatechange` event will be triggered five times, 0-4 corresponding to each value of `readyState` property.

The status property has following possible values:

1. **`status`:** Returns the status as a number (e.g., 404 for "Not Found" and 200 for "OK").
2. **`statusText`:** Returns the status as a string (e.g., "Not Found" or "OK").

Value	Description
200	OK
404	Page not found

4.3 AJAX WEB APPLICATION MODEL

- The Web browser accepts input from the user and JavaScript `XMLHttpRequest` object's open() and send() methods sends the request to the server depending upon the event handled in the JavaScript.
- For sending request, either POST or GET method is used depending upon whether data is sensitive or not. With POST method, form data is sent through sent() function whereas with GET method, open() function sends form data to the server.
- At Web server, processing can be done using PHP script to create response for the request. During processing, there might be need to fetch required information from database, XML file or text file etc.
- The result of the processing is sent as HTML or XML file which will be updated on the current web page. The working of AJAX web application model is represented in the following diagram.

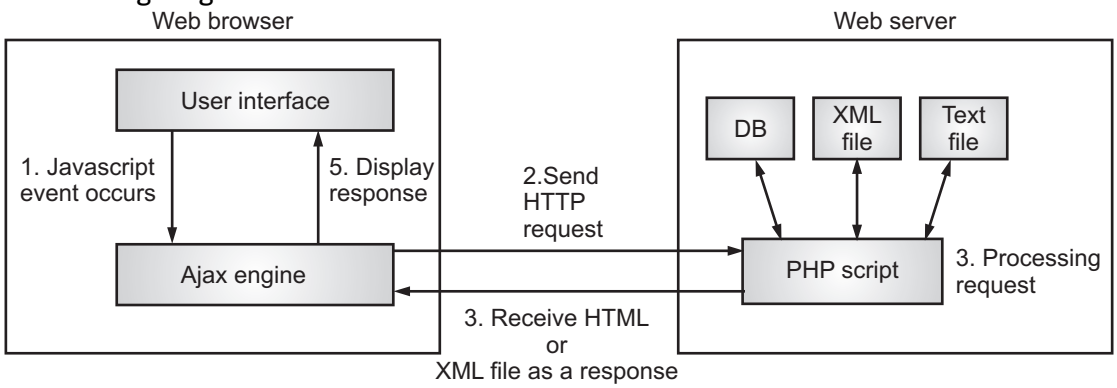


Fig. 4.1: AJAX Web application Model

- Here, communication between web browser and web server can be synchronous or asynchronous. In synchronous communication, user has to wait for the response from the server till that no other task can be performed. This may include waiting time for the user.

- In asynchronous communication, user can continue to use the web application while the client program requests information from the server in the background.

4.4 AJAX - PHP FRAMEWORK

- A PHP AJAX framework is able to deal with database, searching of data, and build pages or parts of page and return the response such as HTML file or XML file to the `XMLHttpRequest` object.
- Using AJAX, you can pass name of php file along with form data as a second argument to `XMLHttpRequest` object's open() method to perform processing at server side. The server uses PHP file for processing the request. The response sent by the server will be displayed on the current web page, so it is possible to partially update the web page. This reduces the need to do a page refresh or full page reload for every user interaction.
- The following are AJAX framework which are used for creating web applications with a dynamic link between the client and the server:
  - Quicknet is an AJAX framework that provides secure data transmission, uses PHP on the server side.
  - SAJAX PHP framework with a lot of functions, easy to integrate functions yourself.
  - XAJAX uses JSON or XML format, on the server side.

Program 4.1: Write AJAX program to read a text file and print the contents of the file when the user clicks on the Print button.

file.html:

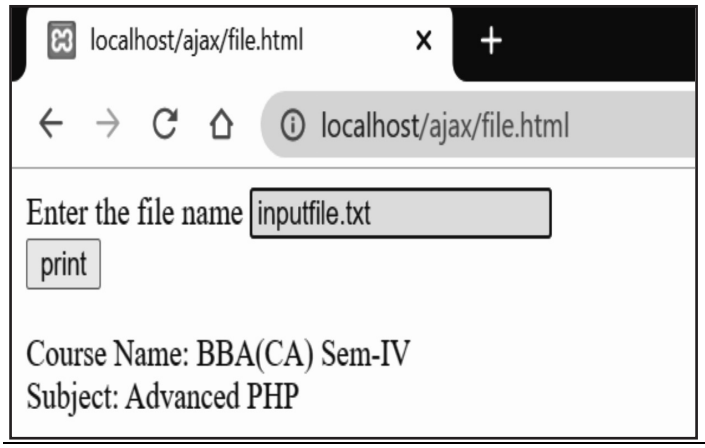
```
<html>
<head>
<script>
function displayFile()
{
   str=f1.txt.value;
   if(str=="" || str==null)
   {
      alert("Enter file name");
   }
   else
   {
      var xmlhttp;
```

```
        if(window.XMLHttpRequest)
          xmlhttp= new XMLHttpRequest();
        else
          xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
        xmlhttp.open("GET","file.php?txt="+str,false);
        xmlhttp.send();
        document.getElementById("result").innerHTML=xmlhttp.responseText;
    }
}
</script>
</head>
<body>
    <form name="f1" method="GET">
    Enter the file name
    <input type="text" name="txt" value=""/><br>
    <input type="button" value="print" onclick="displayFile()"/>
    <div id="result"></div>
    </form>
</body>
</html>
```

file.php:

```
<?php
$file1=$_GET['txt'];
if($fp=fopen($file1,"r"))
{
    while(($line=fgets($fp,80))!=false)
    {
        echo"<br>".$line;
    }
}
else
{
    echo"File does not exist.";
}
?>
```

Output:



Program 4.2: Write an AJAX program to display list of games stored in an array corresponding to search string given by user in the HTML form.

game.html:

```
<html>
<head>
<script>
function show(str)
{
   if (str.length == 0)
   {
        document.getElementById("txtHint").innerHTML = "";
        return;
    }
    else
    {
       var xmlhttp;
       if(window.XMLHttpRequest)
          xmlhttp= new XMLHttpRequest();
```

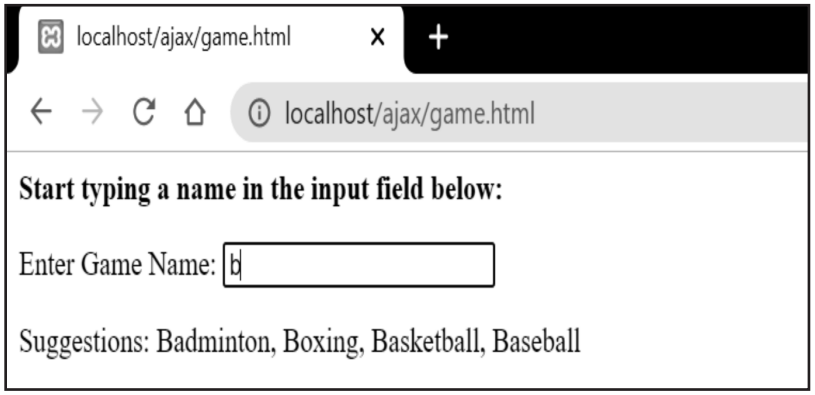
```
        else
           xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

        xmlhttp.onreadystatechange = function()
        {
            if (xmlhttp.readyState == 4 && xmlhttp.status == 200)
             {
                   document.getElementById("txtHint").innerHTML=
                   xmlhttp.responseText;
             }
         };
         xmlhttp.open("GET", "game.php?txt=" + str, true);
         xmlhttp.send();
    }
}
</script>
</head>
<body>
<p><b>Start typing a name in the input field below:</b></p>
<form>
Enter Game Name: <input type="text" name="txt" onkeyup="show(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>
</body>
</html>
```

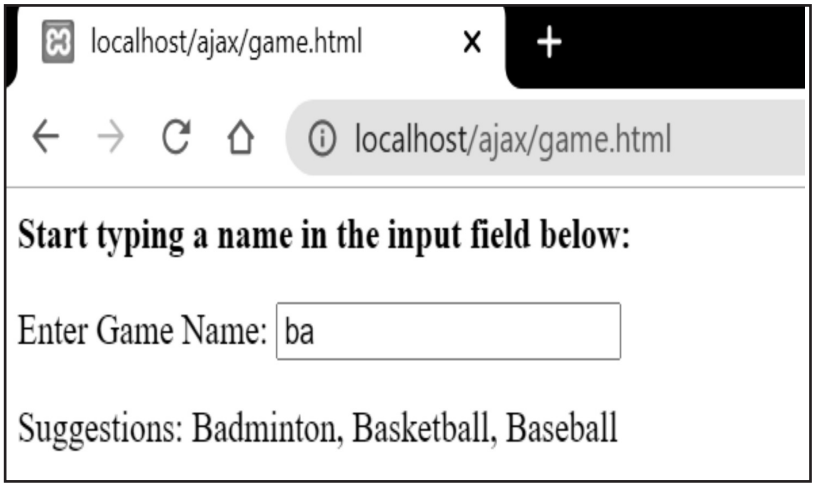
game.php:

```
<?php
// Array with game names
$a = array("Cricket", "Archery", "Badminton","Boxing","Curling", "Tennis",
"Skateboarding", "Surfing", "Hockey", "Yoga", "Gymnastics", "Karate",
"Weightlifting", "Volleyball","Wrestling", "Basketball",
"Baseball","Cycling", "Running", "Climbing");
// get the txt parameter from URL
$game = $_REQUEST["txt"];
$hint = "";
// lookup all hints from array if $game is not empty
if ($game !== "")
{
    $len=strlen($game);
    foreach($a as $name)
    {
        if (stristr($game, substr($name, 0, $len)))
        {
            if ($hint === "")
```

```
            {
                $hint = $name;
            }
            else
            {
                $hint .= ", $name";
            }
        }
    }
}
// Output "no suggestion" if no hint was found or output correct values
echo $hint === "" ? "no suggestion" : $hint;
?>
```

Output:





- **1st Output Screen:** The character 'b' is entered, so all possible strings which are starting with 'b' has been displayed

- **2nd Output Screen:** Now characters 'ba' are entered, so all possible strings which are starting with 'ba' has been displayed.
- When search characters are entered by user, these characters are sent to the server. In PHP file input characters are matched from the beginning of each string from the array which stores game names and all possible matching strings are displayed.

4.5 PERFORMING AJAX VALIDATION

- The form validation is one of the important tasks for web based applications. JavaScript `XMLHttpRequest` object along with PHP can be used for form data validation.
- When user enters HTML form data, it is send to **.php** file using AJAX for validation purpose. Once the validation is done, PHP will send response to the browser and it will be displayed on the same web page.
- Here browser can send synchronous or asynchronous request to the server for form data validation.

Program 4.3: Write AJAX program to carry out validation for a username entered in textbox. If the textbox is blank, print 'Enter username'. If the number of characters is less than three, print Username is too short'. If value entered is appropriate the print 'Valid username'.

validation.html:

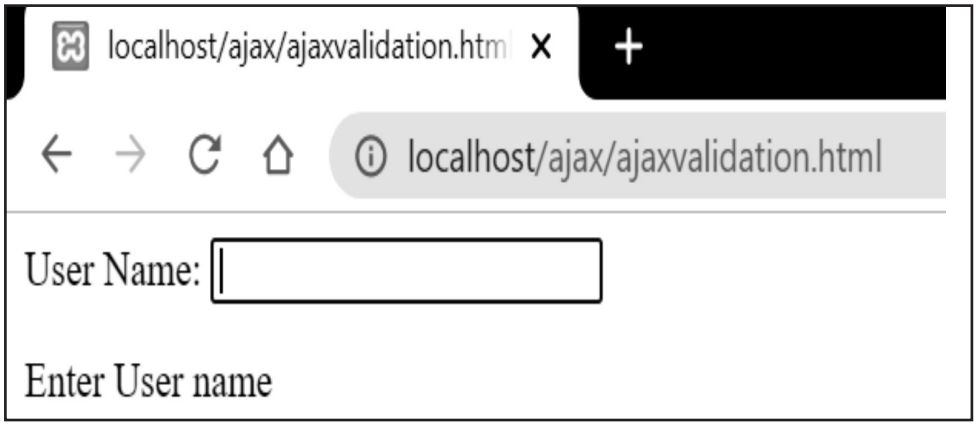
```
<html>
<head>
<script type="text/javascript">
function show(str)
{
    var xmlhttp;
    if(window.XMLHttpRequest)
       xmlhttp= new XMLHttpRequest();
    else
       xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    xmlhttp.onreadystatechange = function()
    {
       if (xmlhttp.readyState == 4 && xmlhttp.status == 200)
       {
         document.getElementById("result").innerHTML= xmlhttp.responseText;
       }
    };
```

```
    xmlhttp.open("GET","AJAX validation.php?txt="+str,false);
    xmlhttp.send();
}
</script>
<body>
<form>
User Name:
<input type="text" name="txt" onkeyup="show(this.value)" /><br>
</form>
<div id='result'></div>
</body>
</html>
```

validation.php:

```
<?php
$user= $_GET['txt'];
if($user==="")
  $str="Enter User name";
else if(strlen($user)<3)
  $str= "User name is too short";
else
  $str= "Valid Username";
echo $str === "" ? "no suggestion" : $str;
?>
```

Output:



- **1st Output Screen:** When user name is empty, message sent by the server is "Enter User name".
- **2nd Output Screen:** When length of user name is less than 3, message sent by the server is "User name is too short".
- **3rd Output Screen:** When length of user name is greater than 3 , message sent by the server is "Valid User name".
- The AJAX validation.html file accepts username and sends it to server using JavaScript `XMLHttpRequest` object's open() and send() method. In `validation.php` file, validation is performed and response is sent back to the browser and displayed on the same web page.

4.6 HANDLING XML DATA USING PHP AND AJAX

- You can accept input information like some characters which are to be searched in the XML file. The search information can be sent to open() method along with .php file which will be executed at server to process the request.
- In .php file, to process the request, read XML file data and check for all possible matches for search string and send the search result to browser as a HTML file. When the response will be ready, search result will be displayed on the same web page using AJAX.

- The steps to write AJAX code to retrieve information from XML file using PHP:
  1. Accept search string from user.
  2. Send search string to open() method along with .php file name.
  3. In .php file, read XML file data.
  4. Find all possible match for search string with XML file data.
  5. Display search result in PHP script to create HMTL response for the given request.
  6. This output will be sent to browser and will be displayed on same web page using AJAX.
- All above steps are illustrated in the following program.

Program 4.4: Write AJAX program to accept the movie name and print all possible suggestions for movie name from XML file. Also display movie details. Consider `movie.xml` file as follows:

movie.xml:

```
<?xml version="1.0"?>
<Movie_Store>
<Movie>
       <Category>Biography</Category>
       <MovieName>Dangal</MovieName>
       <ReleaseYear>2016</ReleaseYear>
  </Movie>
   <Movie>
      <Category>Action</Category>
      <MovieName>Tanhaji</MovieName>
      <ReleaseYear>2020</ReleaseYear>
   </Movie>
   <Movie>
      <Category>horror</Category>
      <MovieName>Bhoot</MovieName>
      <ReleaseYear>2020</ReleaseYear>
   </Movie>
   <Movie>
       <Category>Family</Category>
       <MovieName>The Sky Is Pink</MovieName>
       <ReleaseYear>2019</ReleaseYear>
   </Movie>
</Movie_Store>
```

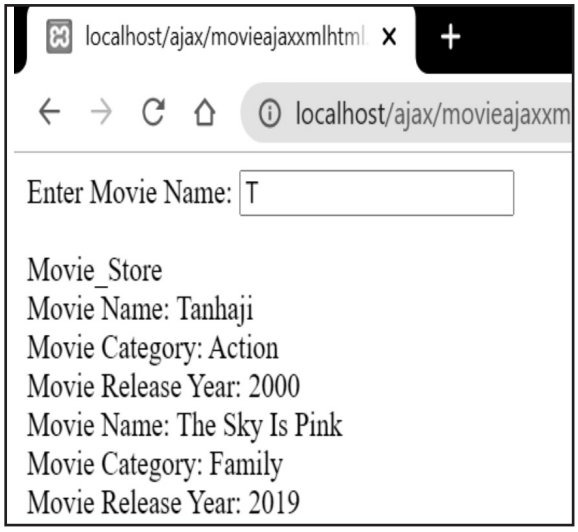
movie.html:

```
<html>
<head>
<script type="text/javascript">
function display()
{
    n=f1.txt.value;
    var xmlhttp;
    if(window.XMLHttpRequest)
       xmlhttp= new XMLHttpRequest();
    else
       xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
   xmlhttp.onreadystatechange=function()
   {
     if(xmlhttp.readyState==4 && xmlhttp.status==200)
     {
        document.getElementById('result').innerHTML = xmlhttp.responseText;
        // step 6
     }
   };
   xmlhttp.open("GET","movie.php?txt="+n,true); // step2
   xmlhttp.send();
}
</script>
<body>
<form name="f1">
Enter Movie Name:
<input type="text" name="txt" onKeyUp="display()"/><br>  // step1
</form>
<div id='result'></div>
</body>
</html>
```

movie.php

```
<?php
$msearch=$_GET['txt'];
$xml=simplexml_load_file("movie.xml");
echo $xml->getName() . "<br>";
$hint = "";
$len=strlen($msearch);
if ($msearch !== "")
{
  foreach($xml->children() as $movie)          // step3
  {
       if (strstr($movie->MovieName, substr($msearch, 0, $len)))  // step 4
       {
           $hint .= " Movie Name: $movie->MovieName <br>
           Movie Category: $movie->Category <br>
           Movie Release Year: $movie->ReleaseYear<br>";
       }
  }
}
// Output "no suggestion" if no hint was found or output correct values

echo $hint === "" ? "no suggestion" : $hint;
?>
```

Output:



- **1st Output Screen:** When user types 'T' character then onkeyUp() event occurs. XMLHttpRequest object is sent to the server which executes .php file. The .php file reads XML file contents and 'T' is matched with the XML file data and matching result is sent to the browser.
- **2nd Output Screen:** When search characters changes everytime, .php file be executed and displays new search result.

4.7 CONNECTING DATABASE USING PHP AND AJAX

- You can accept input information like some characters which are to be searched in the database.
- The search information can be sent to open() method along with .php file which will be executed at server to process the request.
- In .php file, request will be processed by establishing database connection, execute SQL query to find the matching records and send the search result back to the browser as a HTML file.
- When the response will be ready, search result will be displayed on the same web page using AJAX.
- The steps to write AJAX code to retrieve information from database using PHP:
  1. Accept search string from user.
  2. Send search string to open() method along with .php file name.
  3. In .php file, make database connection and select required database.
  4. Execute the query to fetch matching records from database.
  5. Display required information in PHP script to create HMTL response for the given request.
  6. This output will be sent to browser and will be displayed on same web page using AJAX.

  All above steps are illustrated in the following program.

Program 4.5: Consider table employee(emp_id, emp_name, designation, salary). Write program to accept the employee name and print the employee's details. Assume employee(emp_id, emp_name, designation, salary) table exists in MySQL and has sufficient number of records.

Employee.html:

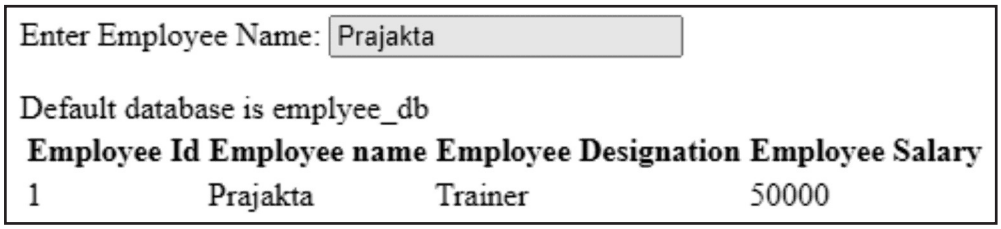
```
<html>
<head>
    <script type="text/javascript">
   function display()
   {
      n = f1.txt.value;
      var xmlhttp;
      if (window.XMLHttpRequest)
         xmlhttp = new XMLHttpRequest();
      else
         xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
         xmlhttp.onreadystatechange = function ()
         {
            if (xmlhttp.readyState == 4 && xmlhttp.status == 200)
            {
                document.getElementById('result').innerHTML =
                xmlhttp.responseText;   //step 6
            }
         };
         xmlhttp.open("GET", "Employee.php?txt=" + n, true);   //step 2
         xmlhttp.send();
    }
    </script>
   <body>
    <form name="f1">
    Enter Employee Name:
     <input type="text" name="txt" onKeyUp="display()" /><br>
    </form>
    <div id='result'></div>
</body>
</html>
```

Employee.php:

```
<?php
$name=$_GET['txt'];
$host = 'localhost';
$user = 'root';
$pass = '';
$db='emplyee_db';
$mysqli = new mysqli($host,$user,$pass,$db);
if($mysqli -> connect_errno )
{
    die('Database connection error: ' . mysqli_error());
}
$mysqli -> select_db($db);
if ($result = $mysqli -> query("SELECT DATABASE()"))
{
  $row = $result -> fetch_row();
  echo "Default database is " . $row[0];
  $result -> close();
}
$query= "select * from employee where emp_name='$name'";
$result = $mysqli -> query($query);
if(mysqli_num_rows($result) > 0)
{
    echo "<table>";
    echo "<tr>";
    echo "<th>Employee Id</th>";
    echo "<th>Employee name</th>";
    echo "<th>Employee Designation</th>";
    echo "<th>Employee Salary</th>";
    echo "</tr>";
    while($row = mysqli_fetch_array($result))
    {
        echo "<tr>";
        echo "<td>". $row['emp_id']. "</td>"; //step 5
```

```
            echo "<td>". $row['emp_name']. "</td>";
            echo "<td>". $row['designation']. "</td>";
            echo "<td>". $row['salary']. "</td>";
            echo "</tr>";
        }
        echo "</table>";

        mysqli_free_result($result);
    }
    else
        echo "No records found for the given match.";
    $mysqli -> close();
    ?>
```

Output:



- Every time the content of search input is changed or keyup event occurs, AJAX code uses open() and send() methods to send a request to the "dbemp.php" file which retrieves the records from employee table related to the search string. All records from result set are inserted inside a and displayed on the current web page at browser.

PRACTICE PROGRAMS

Program 4.6: Write a PHP script using AJAX concept, to check user name and password are valid or Invalid (use database to store user name and password).

uservalidation.html:

```
<html>
<head>
<script type="text/javascript">
function validate()
{
   user_name=document.getElementById('uname').value;
```

```
        password=document.getElementById('pass').value;
        var xmlhttp;
        if(window.XMLHttpRequest)
           xmlhttp= new XMLHttpRequest();
        else
           xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
        xmlhttp.onreadystatechange = function()
        {
           if (xmlhttp.readyState == 4 && xmlhttp.status == 200)
           {
             document.getElementById("result").innerHTML = xmlhttp.responseText;
           }
        };
        xmlhttp.open("POST","uservalidation.php",false);
        xmlhttp.setRequestHeader("Content-type","application/x-www-form-
        urlencoded");
       xmlhttp.send("uname="+user_name+"&pass="+password);
       document.getElementById("result").innerHTML = xmlhttp.responseText;
    }
    </script>
    </head>
    <body>
    <form method="POST">
    Enter username: <input type="text" id="uname" name="uname">
    <span id="a"></span><br><br>
    Enter password: <input type="text" id="pass" name="pass">
    <span id="a"></span><br><br>
    <input type="submit" value="proceed" onClick=validate()>
    </form>
    </body>
    </html>
```

uservalidation.php:

```
    <?php
    $uname=$_POST['uname'];
    $password=$_POST['pass'];
    $host = 'localhost';
    $user = 'sybba';
```

```
$pass = 'sybba123';
$conn = mysql_connect($host, $user, $pass);
mysql_select_db('user_db',$conn );
$q=mysql_query("select * from users");
$found=0;
while($row=mysql_fetch_array($q))
{
   if($row[0]==$uname && $row[1]==$password)
      $found=1;
}
if($found==0)
   echo "INVALID USERNAME AND PASSWORD";
else if($found==1)
   echo "VALID USERNAME AND PASSWORD";
?>
```

- If GET method is used then send form data by writing it after PHP file name and with POST method, it can be send in send() method and not with open() function. This can be written as follows:

```
xmlhttp.open("GET","uservalidation.php?uname="+user_name+"&pass="+password,
false);
xmlhttp.send();
```

Program 4.7: Write AJAX program to get player details from XML file when user select a player name. Create XML file for storing details of player (Country, player name, wickets, runs).

player.xml:

```
<?xml version="1.0"?>
<playerinfo>
   <player>
   <Country>India</Country>
   <Player_name>Sachin</Player_name>
   <Wickets>9</Wickets>
   <Runs>450</Runs>
   </player>
   <player>
   <Country>India</Country>
```

```
    <Player_name>Dhoni</Player_name>
    <Wickets>19</Wickets>
    <Runs>200</Runs>
    </player>
    <player>
    <Country>India</Country>
    <Player_name>Virat</Player_name>
    <Wickets>6</Wickets>
    <Runs>321</Runs>
    </player>
</playerinfo>
```

xmlplayer.html:

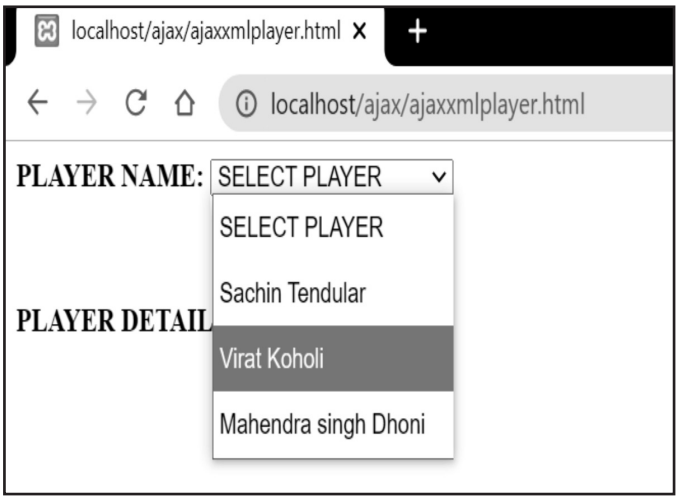
```
<html>
<head>
<script>
function show(str)
{
    if(window.XMLHttpRequest)
    {
        ob=new XMLHttpRequest();
    }
    else
    {
        ob=new ActiveXObject("Microsoft.XMLHTTP");
    }
    ob.onreadystatechange=function()
    {
        if(ob.readyState==4 && ob.status==200)
        {
            document.getElementById("a").innerHTML=ob.responseText;
        }
    }
    ob.open("GET","AJAX xmlplayer.php?q="+str,true);
    ob.send();
}
</script>
</head>
<body>
<form>
<b>PLAYER NAME:</b>
<select name="player_name" onchange="show(this.value)">
```

```
<option value="">SELECT PLAYER </option>
<option value="Sachin">Sachin Tendular</option>
<option value="Virat">Virat Koholi</option>
<option value="Dhoni">Mahendra singh Dhoni</option>
</select>
</form>
<br><br>
<div id="a"><b>PLAYER DETAILS : </b></div>
</body>
</html>
```

xmlplayer.php:

```
<?php
$q=$_GET["q"];
$dom=new DOMDocument();
$dom->load("player.xml");
$x=$dom->getElementsByTagName('Player_name');
for($i=0;$i<=$x->length-1;$i++)
{
    if($x->item($i)->nodeType==1)
    {
        if($x->item($i)->childNodes->item(0)->nodeValue==$q)
        {
            $y=($x->item($i)->parentNode);
        }
    }
}
$player=($y->childNodes);
for($i=0;$i<$player->length;$i++)
{
    if($player->item($i)->nodeType==1)
    {
        echo "<b>".$player->item($i)->nodeName.":</b>";
        echo $player->item($i)->childNodes->item(0)->nodeValue;
        echo "<br>";
    }
}
?>
```

Output:



- The user will get dropdown list of player names. When user selects any name from the list, then using PHP script name of particular player will be checked into `player.xml` file and details will be displayed.

Program 4.8: Write a PHP script using AJAX concept, to check name, mobile number and email address are valid or invalid.

userinfovalidation.html:

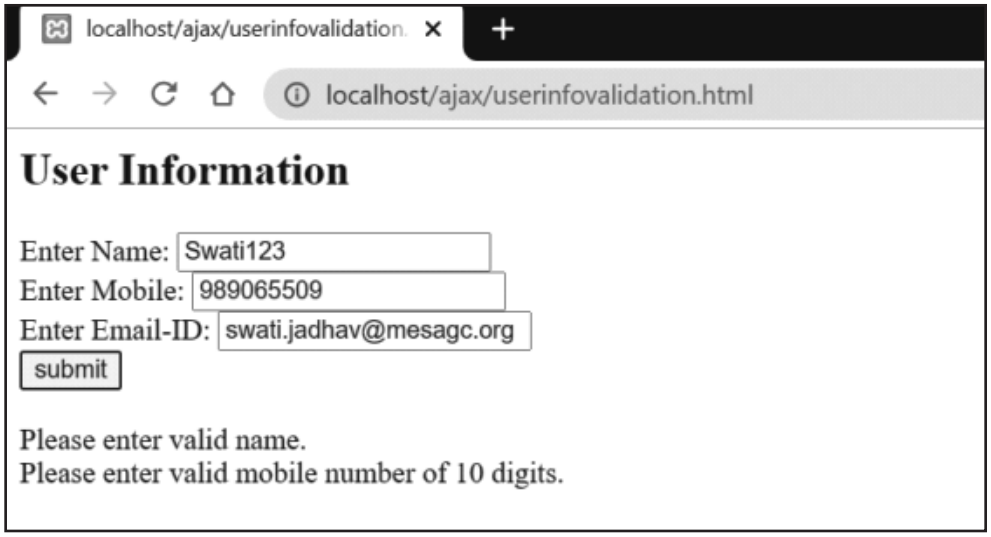
```
<html>
<head>
<h2> User Information</h2>
<body>
<script type="text/javascript">
```

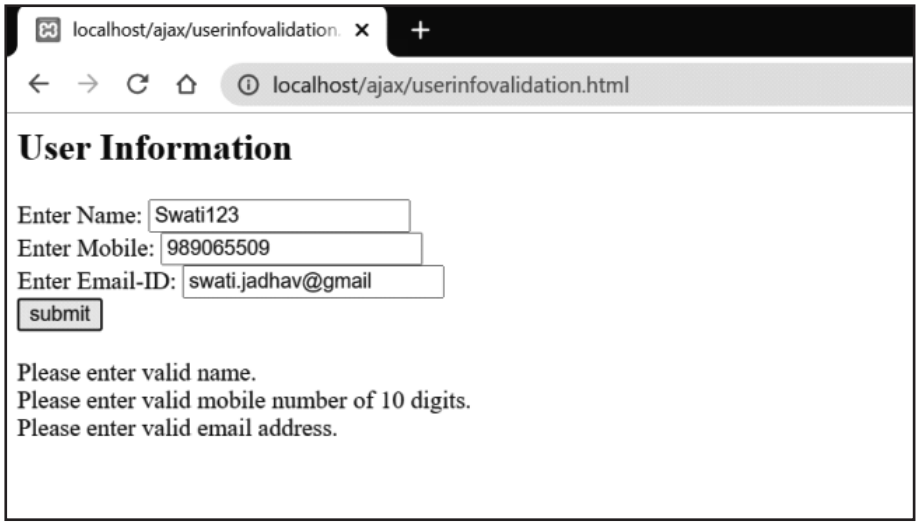
```
function validate()
{
   var xmlhttp;
   if(window.XMLHttpRequest)
       xmlhttp= new XMLHttpRequest();
   else
       xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
   name=f1.txt1.value;
   mno=f1.txt2.value;
   email=f1.txt3.value;
    xmlhttp.open("GET","userinfovalidation.php?txt1="+name+"&txt2="+mno+"&tx
    t3="+email,false);
   xmlhttp.send();
   document.getElementById("result").innerHTML = xmlhttp.responseText;
}
</script>
<form name="f1">
Enter Name:
<input type="text" name="txt1" /><br>
Enter Mobile:
<input type="text" name="txt2"/><br>
Enter Email-ID:
<input type="text" name="txt3" /><br>
<input type="button" value="submit" onclick="validate()"/>
<div id="result"></div>
</form>
</body>
</html>
```

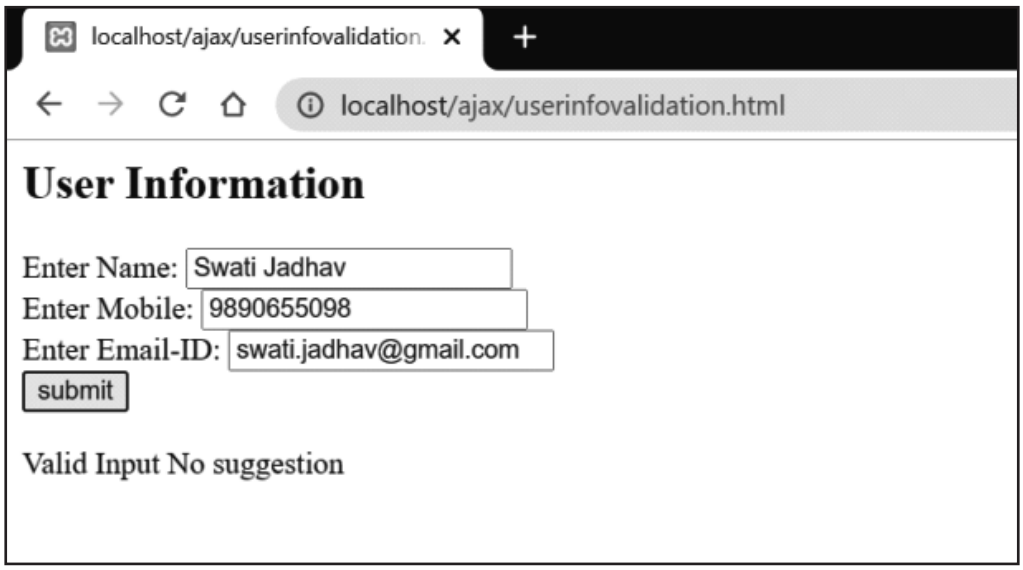
userinfovalidation.php:

```
<?php
$name= $_GET['txt1'];
$mno= $_GET['txt2'];
$email= $_GET['txt3'];
$hint="";
```

```
if($name==="")
  $hint="<br>Enter name";
else if(!preg_match("/^([a-zA-Z' ]+)$/",$name))
  $hint ='<br>Please enter valid name.';
if($mno==="")
  $hint .="<br>Enter mobile number";
else if(strlen($mno)==10)
{
    if(!preg_match("/^([0-9]+)$/",$mno))
    $hint .='<br>Please enter valid mobile number.';
}
else
    $hint .='<br>Please enter valid mobile number of 10 digits.';
if($email==="")
  $hint .="<br>Enter email id";
else
    if(!preg_match("/^[_A-z0-9-]+(\.[_A-z0-9-]+)*@[A-z0-9-]+(\.[A-z0-9-
    ]+)*(\.[A-z]{2,3})$/", $email))
    $hint .='<br>Please enter valid email address.';
    echo $hint === "" ? "<br>Valid Input No suggestion" : $hint;
?>
```







- **1st Output Screen:** Input name contains digits and mobile number contains 9 digits, which is invalid. so respective error messages are disaplyed.
- **2nd Output Screen:** Input name contains digits, mobile number contains 9 digits and email address is also not correct. so respective error messages are disaplyed.
- **3rd Output Screen:** Input name, mobile number and email address are correct, so no error message is displayed.

Summary

- AJAX stands for Asynchronous JavaScript and XML. AJAX is a new technique for creating dynamic, faster and interactive web-based applications using XML, HTML, CSS, and Java Script.
- AJAX uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript.
- The request to the serve can be synchronous or asynchronous. Synchronous request blocks the user until a response is retrieved whereas asynchronous doesn't block the user.
- `XMLHttpRequest` object's `open()` and `send()` methods are used to send request to server.
- The browser can request database information to be retrieved, XML file data, text file, form data validation etc.
- AJAX can be used for interactive communication with XML and database.
- AJAX has two applications models i.e. synchronous and asynchronous.
- `XMLHttpRequest` is used for asynchronous communication between client and server.
- The `XMLHttpRequest` object can send HTTP request, and receive responses.
- There are various PHP frameworks available, like AJAX Core, CakePHP, XAJAX, SAJAX, XOAD, Zephyr, Feather AJAX 1.1 and Tigermouse to integrate AJAX with PHP.
- The SAJAX is an AJAX based framework which generates AJAX enabled JavaScript from many server side languages like PHP, ASP, ColdFusion, Io, Lua, Perl, Python and Ruby.

Check Your Understanding

1. What are all the technologies used by AJAX?
   (a) JavaScript (b) XMLHttpRequest
   (c) Document Object Model (DOM) (d) Cascading Style Sheets (CSS)
   (e) All of the above
2. How can you find out that an AJAX request has been completed?
   (a) readyState=4 (b) readyState=3
   (c) readyState=1 (d) readyState=2
3. What are all the browsers support AJAX ?
   (a) Internet Explorer 5.0 and above
   (b) Opera 7.6 and above
   (c) Netscape 7.1 and above
   (d) Safari 1.2 and above
   (e) All of the above

4. What is the name of object used for AJAX request?
   (a) XmlHttpRequest (b) XMLHttpRequest
   (c) XMLHTTPRequest (d) xmlhttprequest
5. What is the difference between synchronous and asynchronous requests?
   (a) Synchronous request blocks the user until a response is retrieved whereas asynchronous doesn't block the user.
   (b) Asynchronous request blocks the user until a response is retrieved whereas synchronous doesn't block the user.
   (c) Both are same
   (d) None of the above
6. What are the properties of XMLHttpRequest?
   (a) onReadyStateChange - It is called whenever readystate attribute changes.
   (b) readyState - It represents the state of the request.
   (c) responseText - It returns response as text.
   (d) responseXML - It returns response as XML.
   (e) All of the above
7. AJAX Stands for_____.
   (a) Abstract JSON and XML
   (b) Another Java Abstraction for X-Windows
   (c) Another Java and XML Library
   (d) Asynchronous Javascript and XML
8. The XMLHttpRequest object is used to exchange data with a server.
   (a) True (b) False
9. What will be the primary step when you have to send a request using JavaScript to a server?
   (a) You will call the connect() method of the XMLHttp object.
   (b) You will call the execute() method of the XMLHttp object.
   (c) You will call the open() method of the XMLHttp object.
   (d) None of above.
10. The responseText property returns the response as a string format
   (a) True (b) False

ANSWERS

1. (e)	2. (a)	3. (e)	4. (b)	5. (a)	6. (e)	7. (d)	8. (a)
9. (c)	10. (a)						

Practice Questions

Q. I Answer the following questions in short.

1. Which are different technologies used in AJAX ?
2. Where AJAX cannot be used?
3. What are the advantages of AJAX ?
4. How can you find out that an AJAX request has been completed?
5. What is the difference between synchronous and asynchronous requests?
6. What are the security issues with AJAX ?
7. Write syntax for creating XMLHttpRequest object.
8. What are different possible value for readyState property?
9. Which methods are used in AJAX to send the request to server?

Q. II Answer the following questions.

1. What are the differences between AJAX and JavaScript?
2. What are the disadvantages of AJAX ?
3. How GET and POST methods are used in AJAX ? Explain in detail with syntax.
4. How data is sent to server with POST method? Explain with example
5. How to fetch the response received from server? Explain different properties used for this?
6. Explain properties of XMLHttpRequest object.
7. Write AJAX program to accept username and password from user. Also validate username and password from database using PHP script. If both are matching in the database then display "Login Successful" otherwise display "Invalid Login details".
8. Write AJAX program to accept book title from user. When user clicks submit button, display book details from XML file if matches otherwise display "No Match Found" using PHP.
9. Write AJAX program to request an XML file from server and print the details of books when user clicks on the Print button.
10. Write AJAX program to fetch suggestions when is user is typing in a textbox. (Hint create array of suggestions and matching string will be displayed).

Q III Define the terms:

1. Synchronous communication
2. Asynchronous communication
3. responseXML
4. responseText
5. XMLHttpRequest Object

***

5…
Introduction to Web Services

Objectives…

- To learn web services.
- To study model of web service, tools and technologies used to enable web services.
- To understand Benefits and challenges of using web services.
- To study Web services Architecture and its characteristics.
- To understand Core building blocks of web services.
- To learn what are different Standards and technologies available for implementing web services.
- To study different Web services communication models.
- To learn Basic steps of implementing a web service.

5.1 INTRODUCTION

- Web Services allow companies to reduce the cost of doing e-business, to deploy solutions faster and to open up new opportunities.
- Web Services allow applications to be integrated more rapidly, easily and less expensively than ever before. Integration occurs at a higher level in the protocol stack, based on messages centered more on service semantics and less on network protocol semantics, thus enabling loose integration of business functions. These characteristics are ideal for connecting business functions across the Web both between enterprises and within enterprises.
- They provide a unifying programming model so that application integration inside and outside the enterprise can be done with a common approach, leveraging a common infrastructure.
- Web Services are a technology for deploying and providing access to business functions over the Web; J2EE, CORBA and other standards are technologies for implementing Web Services.
- As the use of Web Services grows and the industry matures, more dynamic models of application integration will develop. Eventually systems integration through Web Services will happen dynamically at runtime. Just-in-time integration will herald a new era of business-to-business integration over the Internet.

- In this chapter we will learn what are web services, what are technologies involved in it and its architecture. Also we will see how to implement it.

5.2 DEFINING WEB SERVICES

- Web service is an interface that describes collection of operations that are network accessible through standardized XML messaging.
- Web service is described using a standard, formal XML notion, called its service description. It covers all the details necessary to interact with service, including message formats, transport protocol and location.
- The interface hides the implementation details of the service, allowing it to be used independently of the hardware or software platform on which it is implemented and also independently of the programming language in which it is written.
- This allows and encourages Web Services-based applications to be loosely coupled, component-oriented, cross-technology implementations. Web Services fulfill a specific task or a set of tasks. They can be used alone or with other Web Services to carry out a complex aggregation or a business transaction.
- Web service is a standardized medium to propagate communication between the client and server applications on the World Wide Web. A web service is a software module that is designed to perform a certain set of tasks.
- The web services can be searched for over the network and can also be invoked accordingly.
- When invoked, the web service would be able to provide the functionality to the client, which invokes that web service.
- Web services include any software, application, or cloud technology that provides standardized web protocols (HTTP or HTTPS) to interoperate, communicate, and exchange data messaging – usually XML (Extensible Markup Language) – throughout the internet.
- In other words, web services are XML-centered data exchange systems that use the internet for A2A (Application-to-Application) communication and interfacing. These processes involve programs, messages, documents, and/or objects.
- A key feature of web services is that applications can be written in various languages and are still able to communicate by exchanging data with one another via a web service between clients and servers. A client summons a web service by sending a request via XML, and the service then responses with an XML response. Web services are also often associated with SOA (Service Oriented Architecture) which uses a set of technologies that make up the Web Services Technology Stack. This set of technologies currently consists of SOAP, WSDL, and UDDI.

- To break that down, a web service comprises these essential functions:
  - Available over the internet or intranet networks.
  - Standardized XML messaging system.
  - Independent of a single operating system or programming language.
  - Self-describing via standard XML language.
  - Discoverable through a simple location method.
- A web service supports communication among numerous apps with HTML, XML, WSDL, SOAP, and other open standards. XML tags the data, SOAP transfers the message, and WSDL describes the service's accessibility.
- A web service sits between two sets of java, .net, or PHP apps providing a way for these applications to communicate over a network. On one side, for example, a java app interacts with the java, .net, and PHP apps on the other end by way of the web service communicating an independent language.
- Web services offer different benefits across business operations. The technology helps IT professionals and web architects streamline connectivity by minimizing development time. And with this simplified infrastructure, company executives begin to see higher ROI (Return On Investment). In a B2B operation where both parties understand how the process works, web services provide efficient technology distribution throughout an entire network.

Artifacts of web Service:

- Each web service is made up of two parts:
  - **Service:** Where a web service is an interface described by a service description. It's implementation is service. It is software module deployed on network accessible platforms provided by service provider. It is invoked by or interacted with service requestor. It can also function as requestor, using other web services in its implementation.
  - **Service Description:** it contains the details of the interface and implementation of the service. It includes its data types, operations, binding information and network location.

5.2.1 What are the Different Types of Web Services?

- There are a few central types of web services: XML-RPC, UDDI, SOAP, and REST:
  - **XML-RPC** (Remote Procedure Call) is the most basic XML Protocol to exchange data between a wide variety of devices on a network. It uses HTTP to quickly and easily transfer data and communication other information from client to server.
  - **UDDI:** (Universal Description, Discovery, and Integration) is an XML-based standard for detailing, publishing, and discovering web services. It's basically an internet registry for businesses around the world. The goal is to streamline digital transactions and e-commerce among company systems.

- **SOAP:** It is an XML-based Web service protocol to exchange data and documents over HHTP or SMTP (Simple Mail Transfer Protocol). It allows independent processes operating on disparate systems to communicate using XML.
- **REST:** It provides communication and connectivity between devices and the internet for API-based tasks. Most RESTful services use HTTP as the supporting protocol.

- There are some well-known web services that use markup languages:
  - Web template
  - JSON-RPC
  - JSON-WSP
  - Web Services Description Language (WSDL)
  - Web Services Conversation Language (WSCL)
  - Web Services Flow Language (WSFL)
  - Web Services Metadata Exchange (WS-MetadataExchange)
  - XML Interface for Network Services (XINS)

RESTful Web Services:

- The acronym REST, or sometimes ReST, stands for Representational State Transfer and is an architectural style, meaning each unique URL represents an individual object of some sort. A REST web service uses HTTP and supports/repurposes several HTTP methods: GET, POST, PUT or DELETE. It also offers simple CRUD-oriented services. Fun fact:

SOAP Web Services:

- SOAP is defined as Simple Object Access Protocol. This web service protocol exchanges structured data using XML and generally HTTP and SMTP for transmission. SOAP also uses WSDL (Web Services Description Language) documents to distribute a web service description model. This describes how the SOAP requests (client-side) and responses (server-side) must appear. Additionally, SOAP web services have standards for security and addressing.

5.3 BASIC OPERATIONAL MODEL OF WEB SERVICES, TOOLS AND TECHNOLOGIES ENABLING WEB SERVICES

5.3.1 Basic Operational Model of Web Services

Components in Operational Model of Web Services:

1. **Service provider:**

- The provider creates the web service and makes it available to client application who want to use it. It is the owner of service. It deploys its service on to server.

2. **Service Consumer:**

- Service consumer is the one who is using web service. A consumer is nothing but the client application that needs to contact a web service. The client application can be a

.Net, Java, or any other language based application which looks for some sort of functionality via a web service.

3. **UDDI(Universal Description, Discovery,** and **Integration):**

- It is an XML based registry for businesses word wide to list themselves on the internet. It defines a set of services supporting the description and discovery of the business, organizations, or other web service providers. The UDDI makes the services available and the technical interfaces which may be used to access those services.
- The idea behind UDDI is to discover organizations and the services that organizations offer, much like using a telephone directory. It allows the business to list themselves by name, product, location, or the web service they offer.
- A UDDI works in the following manner:
  - A service provider registers its business with the UDDI registry.
  - A service provider registers each service separately with the UDDI registry.
  - The consumer looks up the business and service in the UDDI registry.
  - The consumer binds the service with the service provider and uses the service.

4. **WSDL (Web Service Description Language):**

- It is an XML based interface description language. It is used for describing the functionality offered by a web service. Sometimes it is also known as the WSDL file. It provides the machine-readable description of how the service can be called, what parameter it expects, and what data structure it returns.
- It describes service as a collection of network endpoint, or ports. It is often used in combination with SOAP and an XML schema to provide XML service over the distributed environment.
- The WSDL describes services as collections of network endpoints, or ports. The WSDL specification provides an XML Format for documents for this purpose.

Operational Model of Web Services:

- Service provider and service consumer communicates using web services. Here service provider publishes an interface for his web services that describes all attributes of web services this is an XML based interface which is called WSDL (Web Services Description Language).
- It is an XML based interface that is used to describe the functionalities of web services, service consumer can get this WSDL and then use the web services and all the request of web services.
- So if the service provider does not know the service consumer then how service consumer can get hold to WSDL document, for that purpose a web service provider publishes his web service through WSDL on an online directory from where consumer can search and find the web services. This online directory or registry is called as UDDI (Universal Description, Discovery and Integration).

- UDDI is an XML based standard for publishing and finding web services.
- Over UDDI service provider can publish his WSDL and service consumer can query and access WSDL.
- Here client or service consumer sends request to server or service provider which processes the request and provides the response but to enable the communication between these two we need to have medium so it is HTTP/Internet and the format will be XML (Extensible Markup Language) /JSON (Java Script Object Notation) through which web services communicate with each other.

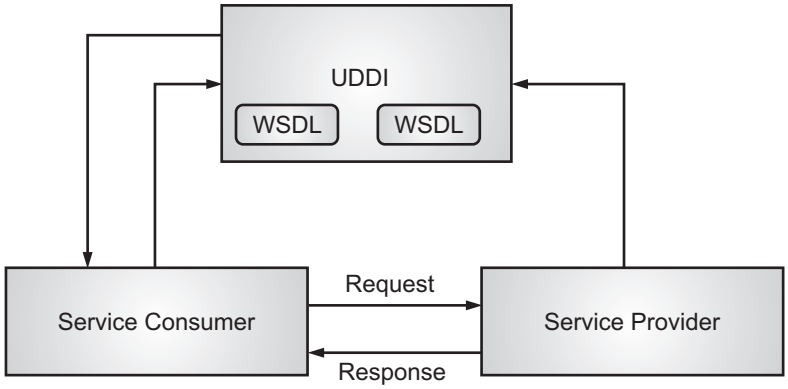


Fig. 5.1: Operational Model of Web Services

5.3.2 Tools and Technologies Enabling Web Services

Following are the most commonly used technologies for the current web services:

1. **XML:**

- XML stands for Extensible Markup Language and is a text-based markup language. It is derived from Standard Generalized Markup Language (SGML).
- XML is preferred and used for storing, organizing and transporting data, rather than specifying how to display it like HTML tags, which are used to display the data.
- The design of XML primarily focuses on documents and the language is widely used for the representation of arbitrary data structures such as those used in web services.
- XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message and then waits for a corresponding XML response. Web services are not tied to any one operating system or programming language, as all the communication is in XML.

2. **HTTP:**

- The HyperText Transfer Protocol or HTTP is a protocol for collaborative, distributed information system and it is used at the application-level. It has been the base for data communication in the World Wide Web (WWW) since the 90's.
- The HTTP communication is usually over TCP/IP connections. The HTTP protocol is based on the interaction of request/responses messages. It is initiated by a client

sending a request method to the server with an URI and a protocol version, followed by a MIME-like message containing additional information like client information, or some content. Once received, the server responds with a response message containing a status line, which includes the message protocol version and a status code, followed by a MIME-like message that contains additional server information.

3. SOAP:

- The Simple Object Access Protocol is standard protocol that provides a definition for XML-based information exchange by means of XML messages. The SOAP version 1.2 defines how structured and typed information can be exchanged between peers in a distributed, decentralized environment.
- SOAP provides a paradigm for allowing different programs running in different or the same operative system to communicate with each other using a transport protocol (mainly HTTP) and XML based structures.
- SOAP is a lightweight protocol that provides a message exchange pattern for structured information in a decentralized, distributed environment; it defines an extensible messaging framework based on XML to provide a message construct (SOAP messages) which can be exchanged over different underlying protocols. This framework is independent of any programming model and other implementation semantics.

4. WSDL:

- WSDL stands for Web Service Description Language, and it is used for describing the interface of any web service this interface describes how your application can call the existing web service, which parameters are required, which are optional, and it also defines how your application should process the responses it receives from a web service. We will be focusing at WSDL 2.0 standard.
- WSDL files are XML files that describe the interface of a web service. It contains the following elements:
    - **Description:** This element is the root of any WSDL 2.0 file, any other WSDL element will be nested within this element
    - **Types:** This element contains an specification of the data types that are to be exchanged between the provider of the service and the consumer.
    - **Interface:** This element describes the operations available in the web service and what messages should be exchanged between the provider and the consumer for each operation (request / response). This element is also used for describing any possible fault message.
    - **Binding:** This element describes how you can access the service over the network. This is usually binding the web service to the HTTP protocol.

- **Service:** This element describes where the service can be accessed over the network, it usually contains an URL leading to the service.
- **Documentation:** This element is optional and provides a humanly readable description of the web service.
- **Import:** This element is optional. It is used for importing XML schemas or even other WSDL files.

```
<definitions>
<types>
  Defines the (XML Schema) data types used by the web services.
</types>
<message>
  Defines the data elements for each operation
</message>
<portType>
  Defines the operations that can be performed and the messages involved.
</portType>
<binding>
  Defines the protocol and data format for each port type.
</binding>
</definitions>
```

5. UDDI:

- The function of the Universal Distribution, Discovery and Interoperability (UDDI) registry is to allow the publication of web services, and to allow anyone to find web services someone else may be offering and you may want to use.
- UDDI is very much like yellow pages for web services, it is a directory of Web Services. The service providers register themselves in UDDI proving contact details (name, phone, fax, etc), geographic information (where in the world they services could be used). Any service provider or service requestor could search in UDDI to find other service providers, which could also include a link to the WSDL that the service provider is offering.
- The service providers can use SOAP messages for registering themselves, or for discovering others.
- The information in UDDI could be divided in three categories:
  - **White pages:** Business name and address, contact information, etc.
  - **Yellow pages:** Type of business, location, industry type, etc.
  - **Green pages:** Technical information, link to the WSDL file or information how to connect with the services.

- Individually any one of these technology is evolutionary. Each technology provides standard for the next step in the advancement of web services.
- One of the big promise of web services is seamless, automatic business integration: a piece of software will discover, access, integrate and invoke new services from unknown companies dynamically without human intervention. Dynamic integration of this nature requires the combined involvement of SOAP, WSDL, UDDI to provide dynamic standard infrastructure for enabling dynamic business of tomorrow.

5.4 BENEFITS AND CHALLENGES OF USING WEB SERVICES

- **Web service** is a standardized medium to propagate communication between the client and server applications on the World Wide Web. A web service is a software module that is designed to perform a certain set of tasks.
- The web services can be searched for over the network and can also be invoked accordingly. When invoked, the web service would be able to provide the functionality to the client, which invokes that web service.

Benefits of Web Services:

- Integration of applications can be done in a faster way. Also it requires less cost.
- Webservices provides interoperability. Interoperability means that system is not specific to any language and any platforms. It also allows designers to combine business systems with all different kind of devices like cell phones, normal desktops, server-based applications etc. with service providers. It also supports traditional Web applications and mainframe systems.
- If the function exists, but only in a form that's hard to integrate, a simple adapter or wrapper function will quickly and cheaply enable other applications to use the existing function.
- It uses standards like XML, WSDL, UDDI, HTTP for implementation, so there is a growing choice of runtime and tooling products. Also customers avoid proprietary lock-in to a single vendor's technology, so vendor's pricing power is reduced.
- It reduces the time to integrate applications because it uses interface-based development, using self-describing service descriptions.
- Web services reduces cost for tooling, training costs for developers, testing costs, and consulting costs to some extend as many web services technologies are being layered into existing operating system and Web application platforms (such as Microsoft .NET and IBM WebSphere). Because the Web services approach provides a means to wrapper existing software, more software is reused. With higher reuse, you reduce new software development and therefore lower overall costs.
- As cost is less, it allows small and medium sized businesses to participate in Web services applications.

Challenges of using web services:

- **Availability:** It is the percentage of time that a service is operating. Every user or client who uses web services know that it is not available hundred percent all the time.
- **Security:** Properties include the existence and type of authentication mechanisms the service offers, confidentiality and data integrity of messages exchanged, no repudiation of requests or messages, and resilience to denial-of service attacks.
- **Response time:** It is the time a service takes to respond to various types of requests. It is a function of load intensity, which can be measured in terms of arrival rates (such as requests per second) or number of concurrent requests. Quality of service takes into account not only the average response time, but also the percentile (95^{th} percentile, for example) of the response time.
- **Throughput:** It is the rate at which a service can process requests. Quality of Service measures can include the maximum throughput or a function that describes how throughput varies with load intensity.

5.5 WEB SERVICES ARCHITECTURE AND ITS CHARACTERISTICS

- An architecture that uses a distributed, discovery-based execution environment to expose and manage a collection of service-oriented software assets. A software asset is a piece of business logic; it can be a component, a queue, or a single method that performs a useful function that you decide to expose to the outside world. The architecture represents principles for governing roles and responsibilities of its constituents.
- With Web services, the Service Oriented Architecture (SOA) is based upon the interactions between three roles: a provider, a registry (or broker) and a requestor. The interactions between these roles involve publishing information about a service, finding which services are available, and binding to those services.

Roles in Web Services Architecture:

1. Service Provider:

- A provider is considered as the owner of a service. From a composite computing perspective, it is a software asset that others regard as a network-accessible service. In most cases, this software asset is exposed as a web service, which contains:
  - Has an XML description.
  - Has a concrete implementation that encapsulates its behaviour.
- How the service is exposed is up to the provider; you can access it through SOAP over HTTP, through a JMS message queue, or via other technologies (such as SMTP); the service may implement a request/response protocol, or it may just receive messages and deliver asynchronous replies.

2. Service Requestor:

- A requestor is nothing but the client application that needs to contact a web service. A requestor is a business that discovers and invokes software assets provided by one or more providers. From a composite computing perspective, a requestor is an application that looks for and initiates an interaction with a provider.
- The requestor can be:
  - A person using a web browser.
  - Computational entities without a user interface, such as another web service.
- The client application can be a .Net, Java, or any other language-based application which looks for some sort of functionality via a web service.

3. Service Broker(Service Registry):

- A registry, or a broker, manages repositories of information on providers and their software assets. This information includes:
  - Business data such as name, description, and contact information ("white pages" data).
  - Data describing policies, business processes, and software bindings—in other words, information needed to make use of the service ("green pages" data).
- The broker is nothing but the application which provides access to the UDDI. The UDDI enables the client application to locate the web service.
- It offers intelligent search capabilities and business classification or taxonomy data (called "yellow pages" data). From a composite computing perspective, a broker represents a searchable registry of service descriptions, published by providers. During the development cycle for a web service, a programmer (or tool) can use the information in registries to create static bindings to services. At runtime, an application can tap into a registry (local or remote) to obtain service descriptions and create dynamic bindings to services.

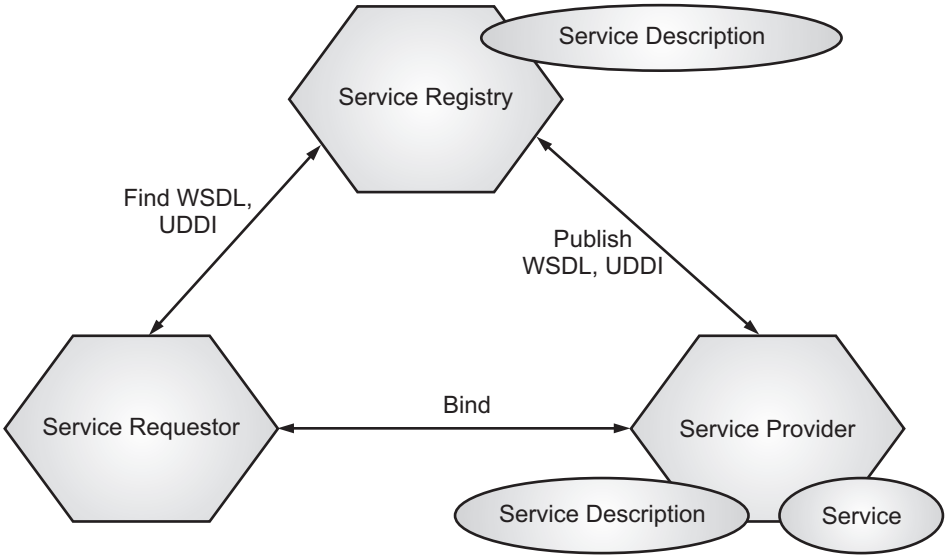


Fig. 5.2: The Service-Oriented Architecture

Operations/Interactions in Web Service Architecture:

1. **Publish:**

- A provider informs the broker (service registry) about the existence of the web service by using the broker's publish interface to make the service accessible to clients. These providers are usually standards organizations, software vendors, and developers.
- According to IBM's Web Services Conceptual Architecture document, several different mechanisms are used to publish service descriptions:
    - **Direct:** The service requestor retrieves the service description directly from the service provider, using email, FTP, or a distribution CD. Here, the service provider delivers the service description and simultaneously makes the service available to a requestor. There is no registry as such; the requestor is responsible for locating services and retrieving their descriptions.
    - **HTTP GET request:** This mechanism is currently used at `http://www.xmethods.com/`, a public repository of web services that developers can use to test their wares. The service request or retrieves the service description directly from the service provider by using an HTTP GET request. This model has a registry (the public web repository), though only in a limited sense.
    - **Dynamic discovery:** This mechanism uses local and public registries to store and retrieve service descriptions programmatically. In the web services world, the most frequently used registry is UDDI, though others exist (for example, ebXML R). Contextually, the service provider is an application that uses a specialized set of APIs to publish the service description.
- Dynamic discovery is the most interesting and versatile publishing model. UDDI and other protocols designed to support dynamic discovery are at the center of the web services landscape.

2. **Find:**

- In the find operation, the service requestor retrieves a service description directly or queries the service registry for the type of service required. The find operation can be involved in two different lifecycle phases for the service requestor: at design time to retrieve the services interface description for program development, and at runtime to retrieve the service's binding and location description for invocation.
- The requestor is an application that uses a specialized set of APIs to query a public or private registry for service descriptions. These queries are formatted in a well-defined, standard XML format and transmitted using an XML messaging format, such as SOAP or XML-RPC.

3. **Bind:**

- Eventually, a service needs to be invoked. In the bind operation the service requestor invokes or initiates an interaction with the service at runtime using the binding details in the service description to locate, contact and invoke the service.

- The binding interaction involves the requestor and provider and, optionally, the registry. In context, binding is what an application does when it uses the service description to create a message to be sent to the service provider. Web service description documents (WSDL documents) specify the network protocols (i.e., HTTP, MIME, SMTP, etc.) that a service supports, the APIs by which the service is accessed, and everything else that a requestor needs to use a service.

Characteristics of Web Services:

1. **XML-based:**

- By using XML as the data representation layer for all web services protocols and technologies that are created, these technologies can be interoperable at their core level. As a data transport, XML eliminates any networking, operating system, or platform binding that a protocol has.

2. **Loosely coupled:**

- A consumer of a web service is not tied to that web service directly. The web service interface can change over time without compromising the client's ability to interact with the service. A tightly coupled system implies that the client and server logic are closely tied to one another, implying that if one interface changes, the other must be updated. Adopting a loosely coupled architecture tends to make software systems more manageable and allows simpler integration between different systems.

3. **Coarse-grained:**

- Object-oriented technologies such as Java expose their services through individual methods. An individual method is too fine an operation to provide any useful capability at a corporate level. Building a Java program from scratch requires the creation of several fine-grained methods that are then composed into a coarse-grained service that is consumed by either a client or another service.
- Businesses and the interfaces that they expose should be coarse-grained. Web services technology provides a natural way of defining coarse-grained services that access the right amount of business logic.

4. **Ability to be synchronous or asynchronous:**

- Synchronicity refers to the binding of the client to the execution of the service. In synchronous invocations, the client blocks and waits for the service to complete its operation before continuing.
- Asynchronous operations allow a client to invoke a service and then execute other functions.
- Asynchronous clients retrieve their result at a later point in time, while synchronous clients receive their result when the service has completed. Asynchronous capability is a key factor in enabling loosely coupled systems.

5. Support Remote Procedure Calls (RPC):

- Web services allow clients to invoke procedures, functions, and methods on remote objects using an XML-based protocol. Remote procedures expose input and output parameters that a web service must support.
- Component development through Enterprise JavaBeans (EJBs) and .NET Components has increasingly become a part of architectures and enterprise deployments over the past couple of years. Both technologies are distributed and accessible through a variety of RPC mechanisms.
- A web service supports RPC by providing services of its own, equivalent to those of a traditional component, or by translating incoming invocations into an invocation of an EJB or a .NET component.

6. Supports Document Exchange:

- One of the key advantages of XML is its generic way of representing not only data, but also complex documents. These documents can be as simple as representing a current address, or they can be as complex as representing an entire book or Request for Quotation (RFQ). Web services support the transparent exchange of documents to facilitate business integration.

5.6 CORE BUILDING BLOCKS OF WEB SERVICES

- The Web Services Description Language (WSDL) and Universal Description, Discovery, and Integration (UDDI), along with SOAP, form the essential building blocks for Web Services. All these blocks together provide the foundation for the just-in-time, Service-Oriented Architecture.

A. WSDL (Web Services Description Language):

- The client invoking the web service should know what the web service actually does, so that it can invoke the right web service and where the web service actually resides. The WSDL file is again an XML-based file which basically tells the client application what the web service does. By using the WSDL document, the client application would be able to understand where the web service is located and how it can be utilized.
- It is an XML-based format for describing Web Services. It describes which operations Web Services can execute and the format of the messages Web Services can send and receive.
- **Major elements of WSDL:**

```
<definitions>
<import>*
<types>
   <schema></schema>*
</types>
```

```
<message>*
   <part></part>*
</message>
<PortType>*
   <operation>*
      <input></input>
      <output></output>
      <fault></fault>*
   </operation>
</PortType>
<binding>*
   <operation>*
      <input></input>
            <output></output>
   </operation>
</binding>
<service>*
   <port></port>*
</service>
</definitions>
```

- **Example of WSDL:**

```
<message name="getTermRequest">
   <part name="term" type="xs:string"/>
</message>
<message name="getTermResponse">
   <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
   <operation name="getTerm">
      <input message="getTermRequest"/>
      <output message="getTermResponse"/>
   </operation>
</portType>
```

```
<binding type="glossaryTerms" name="b1">
<soap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http" />
   <operation>
      <soap:operation soapAction="http://example.com/getTerm"/>
      <input>
          <soap:body use="literal"/>
      </input>
      <output>
          <soap:body use="literal"/>
      </output>
   </operation>
</binding>
```

- **definitions:** This element in a WSDL document acts as a container for the service description. It provides a place to do global declarations of namespaces that are intended to be visible throughout the rest of the document. The target Namespace attribute of the <definitions>element defines the namespace definitions that this document is creating.
- **import:** This element serves a purpose similar to the #include directive in the C/C++ programming language. It lets you separate the elements of a service definition into independent documents and include them in the main document, where appropriate. It promotes the modularization of WSDL documents and creates an environment of reuse that can create clear service definitions. WSDL documents structured in this way are easier to use and maintain, but require any WSDL parsing engine to perform additional I/O operations to import any externally referenced resource. You can have zero or more <import>elements.
- **types:** This element in a WSDL document acts as a container for defining the datatypes used in <message>elements. <message>elements define the format of messages interchanged between a client and a web service. Currently, XML Schema Definitions (XSD) is the most widely used data typing method, but WSDL allows the inclusion of other XML typing approaches.

  The <types>element has zero or more <schema>subelements, which must follow the rules for XMLSchema documents. The <schema>element of our WSDL document is surprisingly simple, but long.
- **message:** This element is used to model the data exchanged as part of a web service.

  <message>elements reference the types defined in the <types>section. The data contained within a <message>element typed by a <message>element is abstract. A message consists of one or more <part>subelements. A <part>subelement identifies the individual pieces of data that are part of this data message and their datatypes.

- **portType:** This element specifies a subset of operations supported for an endpoint of a web service. In a sense, a <portType>element provides a unique identifier to a group of actions that can be executed at a single endpoint.

  The <operation>element represents an operation. This element is an abstract definition of an action supported by a web service. A WSDL <operation>element is analogous to a Java method definition. A WSDL operation can have input and output messages as part of its action. The <operation>tag defines the name of the action by using a name attribute, defines the input message by the <input>subelement, and defines the output message by the <output>sub element. The <input>and <output>elements reference <message>elements defined in the same WSDL document or an imported one. A <message>element can represent a request, response, or a fault.
- **binding:** This element is a concrete protocol and data format specification for a <portType> element. It is where you would use one of the standard binding extensions—HTTP, SOAP, or MIME—or create one of your own. Each protocol has its own wire format. For example, HTTP has a simple header/body format. SOAP, which can exist inside of HTTP and other protocols, has its own header and body. ASOAP message can have attachments included as part of a message.
- **service: T**he binding never referenced the URL at which the web service is actually located. The <service>element typically appears at the end of a WSDL document and identifies a web service. Some WSDL documents do not contain a<service>definition. The primary purpose of a WSDL document is to describe the abstract interface. A <service>element is used only when describing the actual endpoint of a service.

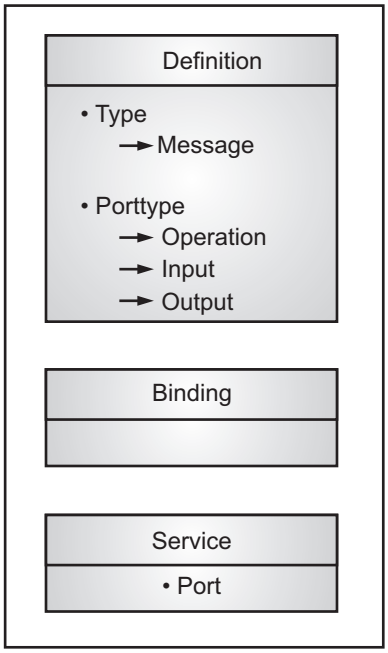


Fig. 5.3: Structure of WSDL

B. UDDI (Universal Description, Discovery, and Integration):

- UDDI is a standard for describing, publishing, and discovering the web services that are provided by a particular service provider. It provides a specification which helps in hosting the information on web services. It is a protocol that describes a standard way of setting up registries of Web Services, along with the methods of querying such registries for information about the Web Services they contain. Each UDDI registry's response to a query contains a WSDL message, which instructs the requester on how to interact with the desired Web Service.
- A business can register three types of information into a UDDI registry.
    - **White pages:** Basic contact information and identifiers about a company, including business name, address, contact information, and unique identifiers such as D-U-N-S numbers or taxIDs. This information allows others to discover your web service based upon your business identification.
    - **Yellow pages:** Information that describes a web service using different categorizations. This information allows others to discover your web service based upon its categorization (such as being in the manufacturing, car sales business, share market).
    - **Green pages:** Technical information that describes the behaviors and supported functions of a webservice hosted by your business. This information includes pointers to the grouping information of web services and where the web services are located.

UDDI Specifications:

- The UDDI project also defines a set of XML Schema definitions that describe the data formats used by the various specification APIs. The current version of all specification groups is Version 2.0.
- The specifications include:
    - **UDDI replication:** This document describes the data replication processes and interfaces to which a registry operator must conform to achieve data replication between sites.
    - **UDDI operators:** This document outlines the behavior and operational parameters required by UDDI node operators. This specification defines data management requirements to which operators must follow.
    - **UDDI Programmer's API:** This specification defines a set of functions that all UDDI registries support for inquiring about services hosted in a registry and for publishing information about a business or a service to a registry. This specification defines a series of SOAP messages containing XML documents that a UDDI registry accepts, parses, and responds to. This specification, along with the UDDI XML API schema and the UDDI Data Structure specification, makes up a complete programming interface to a UDDI registry.

- **UDDI data structures:** This specification covers the specifics of the XML structures contained within the SOAP messages defined by the UDDI Programmer's API. This specification defines five core data structures and their relationships to one another.

  The UDDI XML API schema is not contained in a specification; rather, it is stored as an XML Schema document that defines the structure and datatypes of the UDDI data structures.

Uses of UDDI:

- There are different users of UDDI. They use it to serve their purpose.
- **Business analyst:** For them UDDI is similar to an Internet search engine for business processes. Typical search engines, such as Ask Jeeves, organize and index URLs for web sites. A business exporting a web service needs to expose much more than a simple URL. A business analyst can browse one or more UDDI registries to view the different businesses that expose web services and the specifications of those services.
- **Business users:** They probably won't browse a UDDI registry directly, since the information stored within it is not necessarily reader friendly. A series of marketplaces and business search portals could crop up to provide business analysts with a more user-oriented approach to browsing the services and businesses hosted in a UDDI registry.
- **Software developers:** They use the UDDI Programmer's API to publish services (i.e., put information about them in the registry) and query the registry to discover services matching various criteria. It is conceivable that software will eventually discover a service dynamically and use it without requiring human interaction.

C. SOAP (Simple Object Access Protocol):

- SOAP is an XML-based protocol for accessing web services over HTTP. It has some specification which allows to use a it across all applications.
- SOAP is known as a transport-independent messaging protocol. It transfers XML data as SOAP Messages. Each message has something which is known as an XML document. The main advantage of Web services and SOAP is that they use HTTP, which is the standard web protocol.
- SOAP was developed as an intermediate language so that applications built on various programming languages could communicate to each other and avoid the extreme development effort. It is also capable of describing a remote procedure call or method invocation.
- A SOAP message used to:
  - Send and receive data transmissions across a network using either HTTP or SMTP1
  - Understand MIME encoding rules and base the means of constructing and deconstructing binary attachments on those rules

  - Construct and deconstruct XML documents that conform to the enveloping and encoding rules established by SOAP.
  - Perform the required action, if an action is indicated in the SOAP document
- SOAP messages are normally auto-generated by the web service when it is called.
- Whenever a client application calls a method in the web service, the web service will automatically generate a SOAP message which will have the necessary details of the data which will be sent from the web service to the client application.
- The SOAP message has the following components:
  - **SOAP Envelope**: This identifies the XML document as a SOAP message – This is used to encapsulate all the details in the SOAP message which are exchanged between the web service and the client application. This is the root element in the SOAP message.
  - **SOAP Header**: The header element can contain information such as authentication credentials which can be used by the calling application. It can also contain the definition of complex types which could be used in the SOAP message. By default, the SOAP message can contain parameters which could be of simple types such as strings and numbers, but can also be a complex object type.
  - **SOAP Body:** This contains call and response information – This element is what contains the actual data which needs to be sent between the web service and the calling application.

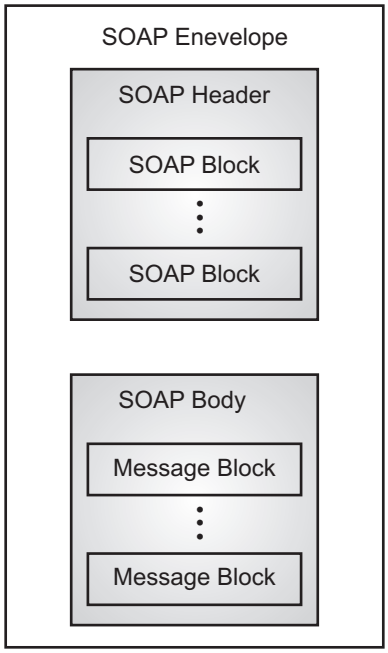


Fig. 5.4: Structure of SOAP Message

5.7 STANDARDS AND TECHNOLOGIES AVAILABLE FOR IMPLEMENTING WEB SERVICES

The Web Services Description Language (WSDL) and Universal Description, Discovery, and Integration (UDDI), SOAP and HTTP are standards and technologies are used in implementation of web service, which are explained in 5.3 and 5.6.

5.8 WEB SERVICES COMMUNICATION MODELS

- In Web services architecture, depending upon the functional requirements, it is possible to implement the models with RPC-based synchronous or messaging-based synchronous/asynchronous communication models. These communication models need to be understood before Web services are designed and implemented.

RPC Based Communication Model:

- The RPC-based communication model defines a request/response-based, synchronous communication. When the client sends a request, the client waits until a response is sent back from the server before continuing any operation.
- Typical to implementing CORBA or RMI communication, the RPC-based Web services are tightly coupled and are implemented with remote objects to the client application. Following diagram represents an RPC-based communication model in Web services architecture.
- The clients have the capability to provide parameters in method calls to the Web service provider. Then, clients invoke the Web services by sending parameter values to the Web service provider that executes the required methods, and then sends back the return values. Additionally, using RPC based communication, both the service provider and requester can register and discover services, respectively.

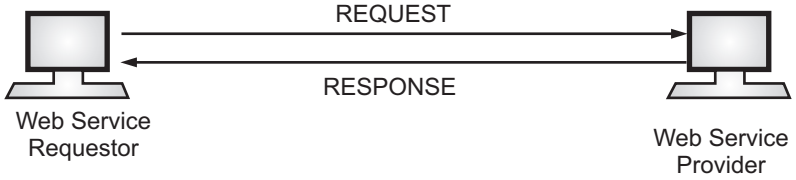


Fig. 5.5: RPC Based Communication Model

Message Passing Based Communication Model:

- The message passing based communication model defines a loosely coupled and document-driven communication. The service requestor invoking a messaging-based service provider does not wait for a response.
- With document style interaction, XML documents are transmitted between the client service requester and the services provider. The XML documents exchanged do not map to back-end method calls of the service provider or the requester.
- In this the client service requester sends a message that include the business document to the Web service provider rather than sending a set of parameters or

procedural calls. The service provider receives the document, processes it, and then may or may not return a message. The document style communication is typically used in conjunction with asynchronous messaging protocols to provide guaranteed and reliable message delivery and to support multi-hop communication based on intermediaries.

5.9 BASIC STEPS OF IMPLEMENTING WEB SERVICES

- The process of implementing Web services is quite similar to implementing any distributed application using CORBA or RMI. However, in web services, all the components are bound dynamically only at its runtime using standard protocols. Following figure illustrates the process of implementing Web services.
- As illustrated the basic steps of implementing Web services are as follows:
  1. The service provider creates the Web service typically as SOAP based service interfaces for exposed business applications. He provider then deploys them in a service container or using a SOAP runtime environment, and then makes them available for invocation over a network. The service provider also describes the Web service as a WSDL-based service description, which defines the clients and the service container with a consistent way of identifying the service location, operations, and its communication model.
  2. The service provider then registers the WSDL-based service description with a service broker, which is typically a UDDI registry.
  3. The UDDI registry then stores the service description as binding templates and URLs to WSDLs located in the service provider environment.
  4. The service requester then locates the required services by querying the UDDI registry. The service requester obtains the binding information and the URLs to identify the service provider.
  5. Using the binding information, the service requester then invokes the service provider and then retrieves the WSDL Service description for those registered services. Then, the service requester creates a client proxy application and establishes communication with the service provider using SOAP.
  6. Finally, the service requester communicates with the service provider and exchanges data or messages by invoking the available services in the service container. In the case of an ebXML based environment, the steps just shown are the same, except ebXML registry and repository, ebXML Messaging, and ebXML CPP/CPA are used instead of UDDI, SOAP, and WSDL, respectively. The basic steps just shown also do not include the implementation of security and quality of service (QoS) tasks.

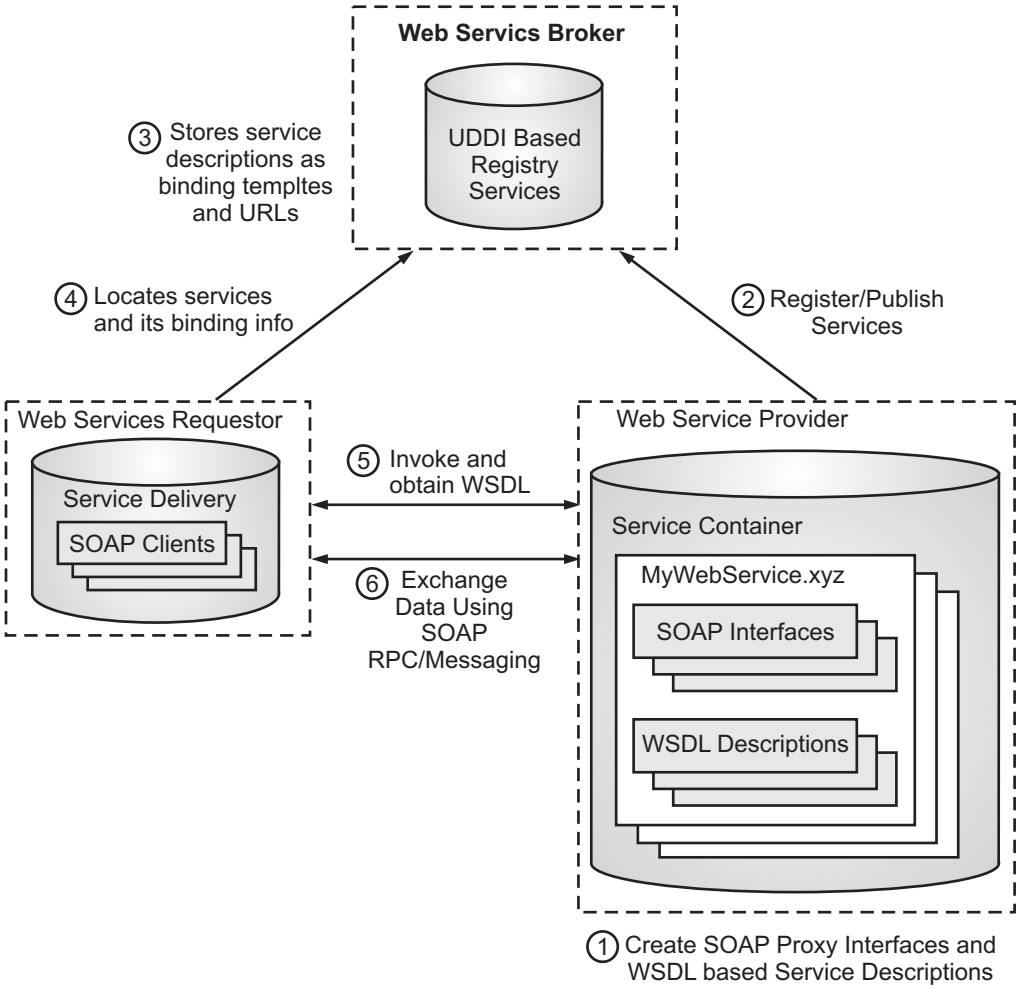


Fig. 5.6: Steps involved in Web services

Summary

- Web services enables to applications to communicate with each other these two applications can be different platforms.
- Web service is an interface that describes collection of operations that are network accessible through standardized XML messaging.
- It works in synchronous or asynchronous mode as per the need of the application.
- Web service is described using a standard, formal XML notion, called its service description. It covers all the details necessary to interact with service, including message formats, transport protocol and location. Web service is a standardized medium to propagate communication between the client and server applications on the World Wide Web. A web service is a software module that is designed to perform a certain set of tasks.

- There are different types of web services XML-RPC (Remote Procedure Call), UDDI-(Universal Description, Discovery, and Integration), SOAP, REST.
- In web services general model follows service provider defines its services over UDDI, service consumer searches and access services of the service provider. The standard medium of communicating between service provider and consumer is in the XML format.
- There are different technologies used in the web services like HTTP, SOAP, WSDL, UDDI.
- The Service Oriented Architecture (SOA) is based upon the interactions between three roles: a provider, a registry (or broker) and a requestor. The interactions between these roles involve publishing information about a service, finding which services are available, and binding to those services.
- Web Service core building blocks consist of WSDL, SOAP and UDDI.
- RPC-based and message passing (Document based) communication models are used with web services.

Check Your Understanding

1. Web services can be discovered using …….
   (a) UDDII (b) UDDI
   (c) UDDDI (d) UDII
2. What is RPC?
   (a) Regional procedure calls (b) Registered procedure counselor
   (c) Remote Procedure Calls (d) Regional protection control
3. What is WSDL?
   (a) Web Services Detail Language (b) Web Services Description Language
   (c) Web Services Development Language (d) Web System Description Language
4. What is NOT a part of a SOAP Message ?
   (a) SOAP Body (b) SOAP Header
   (c) SOAP Envelope (d) SOAP footer
5. Which technologies are involved in web services?
   (a) SOAP (b) WSDL
   (c) HTTP (d) XML
   (e) All of the above
6. The basic Web Services platform is combination of …… and …….
   (a) CSS and Java (b) CSS and HTTP
   (c) XML and HTML (d) XML and HTTP

7. …… is the basis for Web services.
   (a) PHP (b) CSS
   (c) CGI (d) XML
8. Which of the following is correct about SOAP?
   (a) SOAP is an XML-based protocol for exchanging information between computers.
   (b) SOAP is a communication protocol.
   (c) SOAP is for communication between applications.
   (d) All of the above.
9. Which of the following component of Web service describes interfaces to web services?
   (a) UDDI (b) WSDL
   (c) SOAP (d) None of the above.
10. What SOAP stands for?
   (a) State Access Object Protocol (b) State Allied Object Protocol
   (c) Simple Access Object Protocol (d) Simple Allied Object Protocol

ANSWERS

1. (b)	2. (c)	3. (b)	4. (d)	5. (e)	6. (d)	7. (d)	8. (d)
9. (a)	10. (c)						

Practice Questions

Q. I Answer the following questions in short.

1. What is a web service?
2. Explain characteristics of web services
3. List all the technologies of web services.
4. Explain the role of service provider and service requestor.
5. How UDDI works?
6. List benefits of web services.
7. What are challenges with web services?
8. Why web services are widely used?
9. Explain operations or interactions in web service architecture.
10. Explain the importance of XML in web services.

Q. II Answer the following questions.

1. Explain with suitable diagram web services model
2. What are different technologies used in web services

3. Explain WSDL?
4. What are different building blocks of web services? Explain?
5. Discuss steps for implementing web services?
6. What is SOAP? Explain in detail?
7. Explain role of participants in Service-oriented architecture of web services?
8. Explain benefits and challenges of web services
9. Explain the structure of SOAP message.
10. Explain the structure of WSDL.

Q.III Define the terms:

1. RPC
2. XML
3. UDDI
4. HTTP
5. WSDL
5. SOAP
7. Service Requestor
6. Service provider
9. Service broker /Service registry

***

6…

PHP Framework (Joomla/Drupal)

Objectives…

- To understand PHP Framework.
- To understand MVC Architecture.
- To learn and Understand Drupal/Joomla.

6.1 INTRODUCTION TO PHP FRAMEWORK

- A framework or software framework, is a platform for building programs and developing software applications.
- A framework provides pool of APIs which can be reused for developing any software/applications.
- Undoubtedly, PHP is the world's most popular scripting language because of its exceptional features like flexibility and ease-of-use.
- Coding in any language often gets monotonous and repetitive that's why PHP framework is preferred which play an important role in developing easy and user friendly applications.
- PHP framework promotes Rapid Application Development (RAD) which reduces the repetitive code, saves time and helps to create stable and steady applications.
- PHP MVC is an application design pattern that separates the application data and business logic (model) from the presentation (view). MVC stands for Model, View & Controller.
- The controller acts as mediator between the models and views.

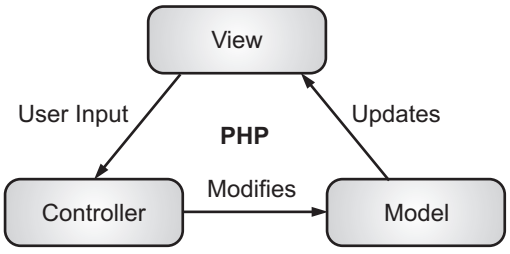


Fig. 6.1: MVC Model

MVC Model:

- **Model:** The model contains the server business logic. It represents the application data and deals with the data validation, data processing and storage.
- **View:** View deals with the presentation of data to the user in a specific format. It takes inputs from the model to display the preferred output to the user.
- **Controller:** Controller acts as a link between view and the model and directs the model to perform the suitable action as requested by the user.

6.1.1 Why use PHP MVC Framework?

- The PHP MVC Framework helps in simplification of working with complex technologies by:
  - Hiding the implementation details.
  - Observing and following professional coding standards.
  - Saving time and increasing developer's productivity as the base implementation of actions such as validating user inputs, database connectivity are already implemented partially.

6.1.2 Content Management System (CMS)

- CMS or Content Management System is a web application for managing digital content.
- A CMS is built on top of an underlying framework.
- It enables users to install themes, plugins, etc.
- In addition, users can download readymade themes for their website and accordingly add or customize contents.
- Using CMS contents can be created, managed, modified and published in a user friendly interface.
- CMSs are normally used for Enterprise Content Management or Web Content Management.
- Drupal and Joomla are the popular CMS.

6.1.3 Some popular PHP Frameworks

1. **Drupal:** Drupal is the #1 and most preferred platform for web content management among global organizations, government establishments, higher education institutions, and NGOs. It is multilingual, flexible and highly capable.
2. **Joomla:** It is a free and open source content management system for web content publishing. It follows the MVC application framework that helps to build influential web applications.
3. **Code Igniter:** It is the most favored by developers who prefer simple and elegant toolkit to create full-featured web applications.

4. **Wordpress:** It is the easiest and most flexible blogging and website content management system for beginners. It is designed for everyone, emphasizing accessibility, performance, security, and ease of use.
5. **CakePHP:** It is an open source and rapid development framework for PHP. It enables the developers to work in organized and rapid manner without loss of flexibility.
6. **Symfony:** It provides a pool of reusable PHP components and a PHP framework to build web applications, APIs, micro services and web services.
7. **Zend Framework:** It is a collection of professional PHP packages. It provides flexibility as it uses a range of languages and 100% object oriented code.
8. **Laravel:** This web application framework provides expressive and elegant syntax for developing web applications.

6.2 INTRODUCTION TO JOOMLA

- Joomla is a Content Management System using which powerful online applications and websites are built.
- One of world's most preferred and a popular software package, Joomla is used to build, organize, manage and publish content for small and large businesses and organizations worldwide.
- It is free and open source; hence its popularity is growing day by day.
- It is written in PHP. It uses object oriented programming technique, stores data in MYSQL database and is built on model view controller web application framework.

6.2.1 Features of Joomla

1. **Free and Open Source:** The major advantage of using Joomla is that it is available for free as it is open source. All source codes can be reused as per users requirements and they are available for free.
2. **Multilingual:** Joomla offers more than 70 languages, hence making it quite popular and widely supported open source Content Management System (CMS).
3. **Updations made Easy:** Joomla consists of an in-built updater that makes the updation process easy for the users, and it does not involve any professional skills. It contains the "One Click Version Update" feature, which is super easy to use.
4. **Banner Management:** Option for easily adding advertising and monetizing the website with the help of banner management is readily available for users. The banner management tool allows you to create clients, campaigns and option to add multiple banners, customizing codes and tracking clicks.
5. **Media Manager:** The media manager is a tool that manages, uploads and organizes media files and folders. With the help of configurable MIME settings, more type of files can be handled.
6. **Content Management:** Joomla is the most preferred Content management system which contains some excellent features that helps the users to organize and manage

contents efficiently. It is very easy to create the content using WYSIWYG (What You See Is What You Get) editor. It allows users to edit the content without any knowledge of code. Joomla has a powerful extensibility feature which offers over 7500 extensions to extend your website and broaden its functionality.

6.2.2 How Joomla works?

- Joomla can be called as segmented or modular content management system.
- It can be expanded, customized, and components can be swapped easily with a few clicks of the mouse.
- Joomla pulls the data from the database requested by the user by using PHP commands and scripts.
- The site's template or theme defines its look and feel, and also allows the user to interact directly with the site and directs about what to do.
- For carrying out operations requested by the user, Plugins and modules are helpful and essential too. All of these components work together to ensure that your site looks and operates the way it is supposed to and provides a consistent experience to the visitors of site.

6.2.3 The Platform Components, Modules and Plugins for Joomla

Components:

- A component is a kind of Joomla! extension. They are the core functional units of Joomla! which can be also termed as mini applications.
- Maximum components have two key parts: an administrator part and a site part.
- The site part is used to render pages of your site when they are requested by the site visitors during normal site operation.
- The administrator part provides an interface to configure and manage different aspects of the component and is accessible through the Joomla! administrator application.
- Joomla! come with a number of core components, like the content management system, contact forms and Web Links.

Modules:

- Modules are lightweight and flexible extensions used for page rendering.
- These modules are often “boxes” arranged around a component on a typical page.
- A popular example is the login module. Modules are assigned per menu item, so you can decide to show or hide (for example) the login module depending on which page (menu item) the user is currently on.
- Some modules are linked to components: the “latest news” module, for example, links to the content component (com_content) and displays links to the newest content items.
- Modules are managed in the Joomla! Administrator view by the Module Manager.

Plugins:

- A plugin is a kind of Joomla! extension.
- Plugins provide functions which are connected with trigger events.
- Joomla provides a set of core plugin events, but any extension can fire (custom) events.
- Whenever a particular event occurs, all plugin functions of the type associated with the event are executed in sequence. This is a powerful way of extending the functionality of Joomla.
- It also offers extension developers a way to allow other extensions to respond to their actions, making extensions extensible.
- The Joomla! plugin architecture follows the Observer design pattern.
- The JPlugin class provides the way to register custom plugin code with core or custom events.
- The JEventDispatcher class is an event handler which calls all plugins registered for a particular event, when that event is triggered.

Software Requirements for Joomla:

Browser Requirement:

- The latest release of each of the latest two supported major versions of:
- Desktop browsers:
  - Google Chrome
  - Firefox
  - Safari
  - Microsoft Edge
  - Opera

Web Server requirement:

- Joomla latest versions work on any web server with a version of PHP that meets the PHP version requirements. It works on Apache 2.4+, Nginx 1.8+ and Microsoft IIS 7.

Database requirement:

- PostgreSQL 9.1 or higher, MySQL 5.5.3 or higher and Microsoft SQL Server.

PHP requirement:

- Use PHP version 7.3 or higher for current release versions of Joomla.

6.2.4 Administering Joomla

- The Administrator plays the key role in Joomla's backend management interface.
- All type of management, supervision and control, configurations and settings to your site and content creation is handled by the Administration area.

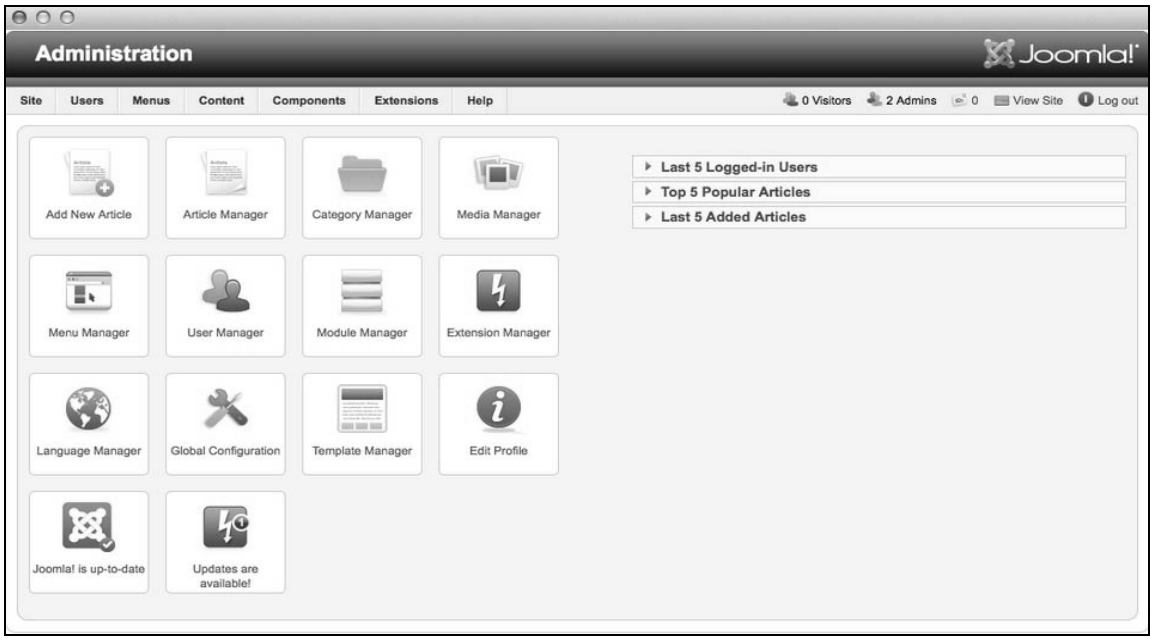


Fig. 6.2: The Administration Page

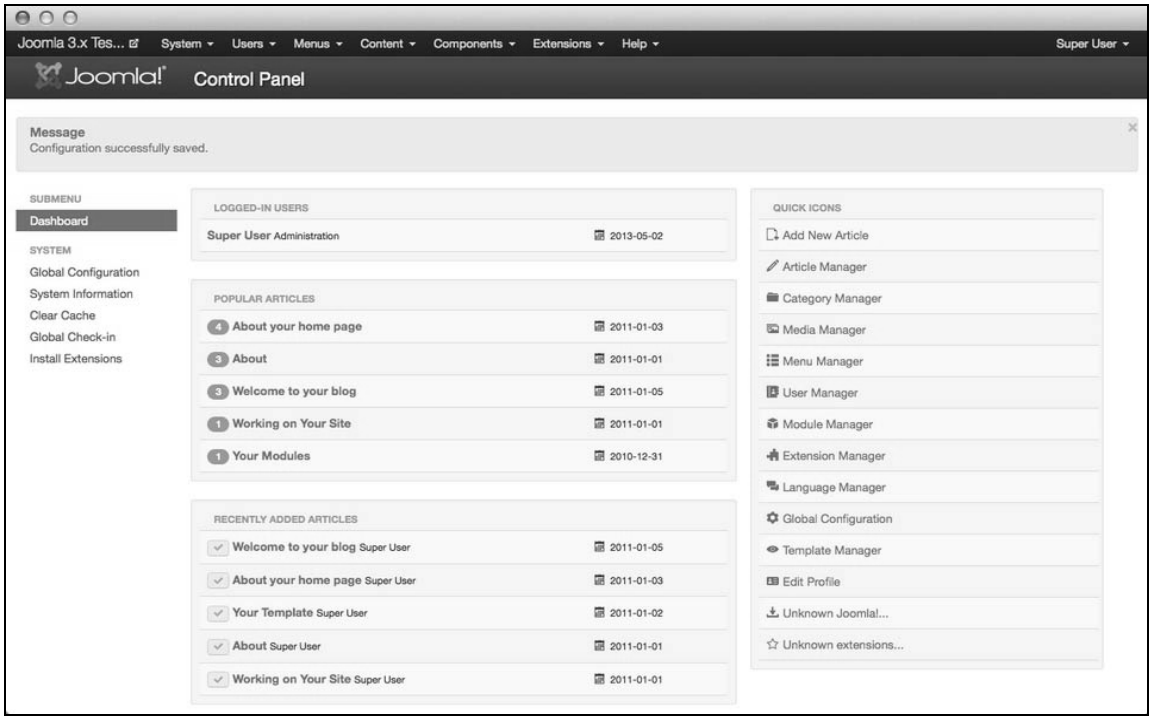


Fig. 6.3: The Joomla Administration Dashboard

- The Admin Menu is located at the top of administrator area which provides easy navigation to all sub menus and provides with Quick Icons on the main Control Panel. So, those users can go easily from one area of administrative site to other.

6.2.4.1 Presentation Administration for Joomla

- The appearance of a Joomla site is governed by various facets of content display.
- The primary decisive factor for the look and feel of a site is the template or templates.
- The graphics, colors, fonts of all site pages are all determined by the **template manager**.
- Some extensions, such as Simple Machines Forum (SMF), have their own theme settings. Therefore, the template manager is responsible for most of the site appearance.
- In addition to the template selection, the selected display language plays a significant role in determining the presentation of the site.
- The configuration for languages options is offered by the **Language Manager**.

A. Template Manager:

- The Template Manager provides the flexibility to the administrator to select the default template as well as provide editing options for both the main index file of the template and the style sheet file(s) which means that you don't need to use a text editor or FTP capabilities to make simple edits to a template.

Fig. 6.4: The Template Manager

- In the above figure, names of templates can be seen. Clicking the template name will display the template configuration screen.
- All of the basic details of the template are provided on the screen, including a short template description which can be found in template's XML descriptor file.
- The Parameters pane/section displays any parameters of the template available for configuration.
- A template may be assigned to particular menu items using the list box on the right side of the screen.
- A template can be assigned to be used for an individual menu item, multiple menu items, all unassigned articles, or none.

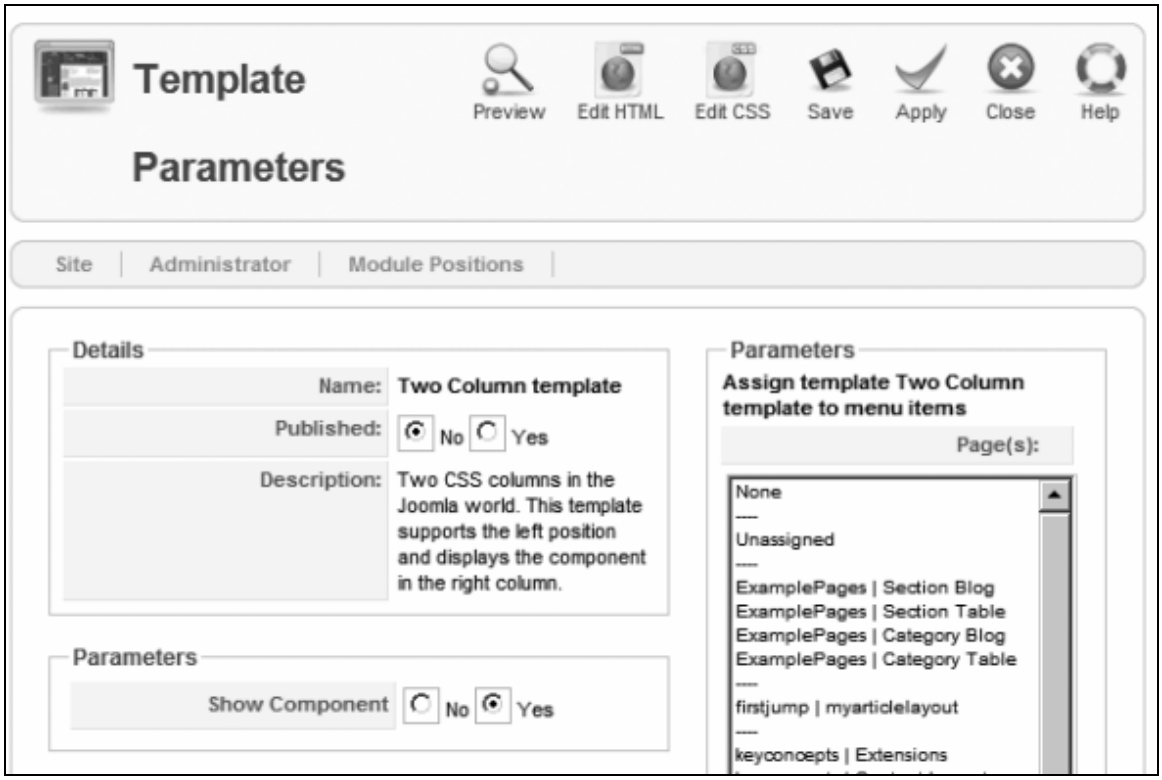


Fig. 6.5: The template configuration screen displays general settings as well as template-specific parameters

- A basic text editor is displayed after clicking the Edit HTML button (see figure 6.5).
- The text editor allows to perform minor changes or updates and touch-ups as frequent direct editing is not feasible and convenient. Also the text editor doesn't possess advanced features like search, replace and syntax highlighting.

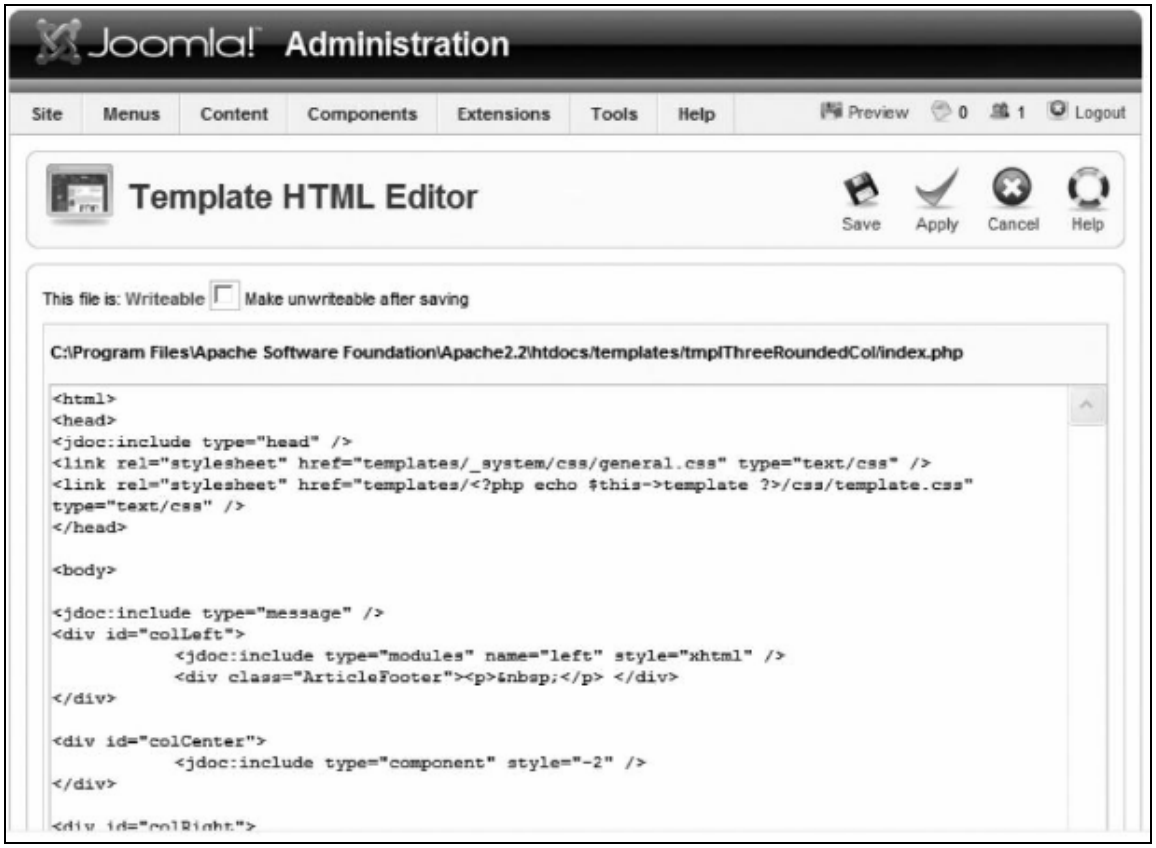


Fig. 6.6: You can edit the HTML of the template

from the Joomla Administrator interface

- The text editor with the style sheet file of the template is opened after clicking on the 'Edit CSS button' on the template screen.
- For a template there can be multiple style sheets, you can select which one to edit (See figure 6.7).
- All style sheet files located in the template's \css folder will be displayed it does not display only the files listed in the template's XML descriptor file.
- Therefore, the list may contain files that are not actually used by the template.

Fig. 6.7: When a template has multiple style sheet files, you can select a specific CSS file for editing

B. Language Manager:

- The Language Manager allows you to configure the available languages installed on the Joomla CMS.
- Joomla provides support in over 40 languages for interface capabilities and a single Joomla installation can support many languages at the same time.
- The default language selection at the time of installation is for the site is the language displayed to new users.
- Joomla gives the preference to each individual registered user to select a preferred language for presentation.
- The Language Manager helps the user to manage and administer a multilingual site, but new languages are installed via the Extension Manager.
- The language will appear in the Language Manager for configuration after installation. (See Figure 6.8).
- The pane under the Language Manager banner allows you to select either Site or Administrator to display the language selections for these interfaces.
- Joomla provides the flexibility of setting the site presentation in one language and administering it in another.

Fig. 6.8: The Language Manager allows selection and configuration of the site and Administrator interface languages

6.2.4.2 Content Administration for Joomla

- The core of Joomla lies in its content. It is the content for which the users visit the site and refer to.
- Here, you can access the **Article Manager**, **Category Manager**, **Featured Articles**, and **Media Manager** where you can manage any images, videos, or audio clips that have been uploaded.
- Content Management is the most basic function of Joomla.
- Majority of the contents are managed by the Article Manager.
- The managing of contents is typically done after the configuration of areas where the articles will be filled. This is done by the Section Manager and Category Manager.
- The FrontPage Manager provides an alternative and timesaving method of viewing all contents that are combined in the form of a "super category" display on the site's homepage.
- More than textual data images, videos, and audio clips help to portray data in a thoughtful and understandable way.
- The Media Manager comes here into picture which carries out the upload and management of media files within the articles.
- Finally, the Trash Manager as the name suggests works like a desktop trashcan which holds the "trashed" content before final deletion of data.

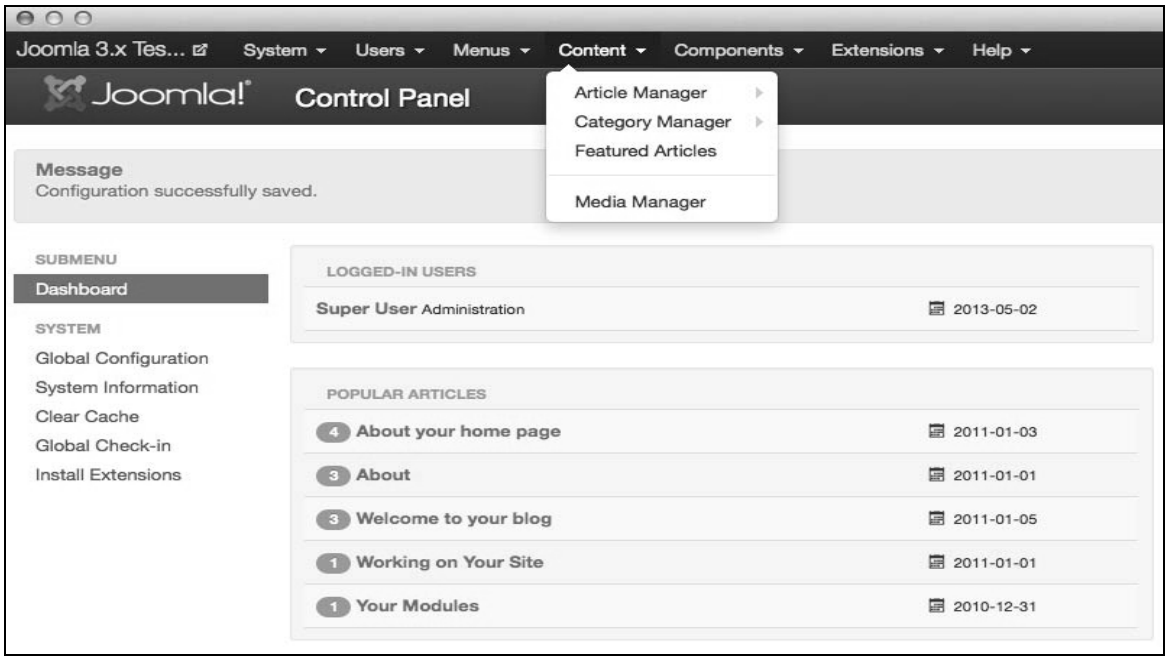


Fig. 6.9: Content Admin menu

A. Article Manager:

- Joomla Article is a piece of content consisting of text, links to other resources, image, media, etc.
- Articles are the basic units of information in the content system and the bottom level in the content hierarchy.
- Articles contain portions of information which should be placed on site in equilibrium, meaning overloading the site with abundance of articles will make the site's visitors confused and lost in the disordered information.
- The Joomla Administrators responsibility is to carefully crop the content on the site so that neither the visitor nor the administrator becomes lost in the confusion.
- Hence correct, required and exact content will make a Joomla site look user friendly and most preferred by users.
- Sometimes a site becomes dense and heavy due to preceding or earlier data.
- Joomla provides a mechanism to prevent the site from becoming overwhelmed with older content, hence pruning is done so that the site becomes smooth and easy to use.
- Pruning is accomplished by the use of Archive button.
- After archival, the data is not available for display on the site.
- In case, the pruned data needs to be viewed, it can be seen in the list of archived items.

- In case, if the user needs to return the article to the site, in the list of archived items, select the desired item and click the Unarchive button and the document will be restored to general publication.

Fig. 6.10: The Article Manager displays all of the published and unpublished articles on the system

- There are numerous global settings that can be applied to articles and which can be accessed by clicking the Preferences button in the Article Manager.
- The Edit configuration window consists of parameters such as linked titles, author names display, and so on as shown in following Figure 6.11.
- These options are already familiar to users as it is available from when the user creates site articles.
- When an article parameter is set to the Use Global option, the selection in this configuration window is the one that is used for that parameter.

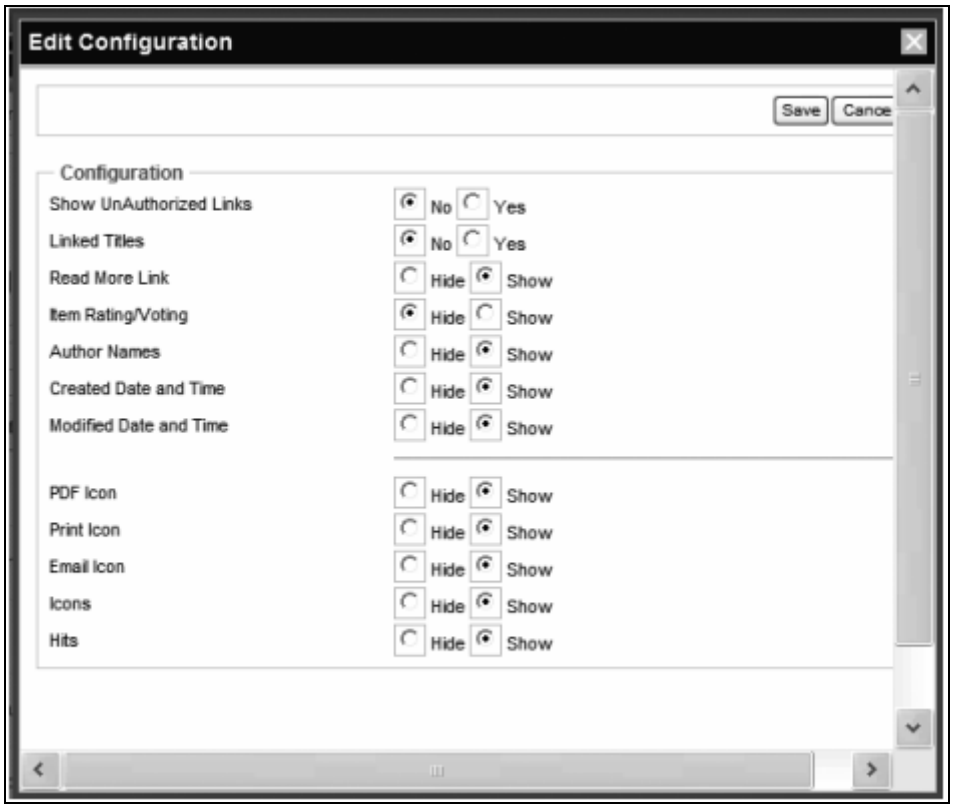


Fig. 6.11: The Article Manager configuration lets you configure the global article settings

B. Section and Category Managers:

- Articles are organized into sections and categories in Joomla.
- The Section and Category Manager provides multiple ways to build up and categorize a large number of different content, such as articles.
- It also helps you manage your content they way you want, which means categorizing it through different criteria.
- Content is always presented within the section or category where it was filed - except when it is set for Frontpage display.

C. Frontpage Manager:

- Menus assist to access sections and categories for displaying most of the site's content but the Frontpage is entirely unique as it is the home page of the site, and content from any section, category, or uncategorized article can be displayed there. Due to this reason, there is a dedicated manager for managing the Frontpage (See Fig. 6.12).
- The functions of Frontpage manager are very much similar to the Article Manager.
- It allows articles to be published, unpublished, reordered, and archived.
- The Fronpage Manager can also be thought of as shortcut for managing the contents that appear on the home page.

Fig. 6.12: The Frontpage Manager displays content from any section or category that is displayed on the home page

D. Media Manager:

- Although the Media Manager (shown in Figure 6.13) handles all types of media (including sound and video), but most Joomla administrators use it majorly for administering images.
- The Media Manager allows media files of several types to be uploaded, including files with the following extensions: .bmp, .csv, .doc, .epg, .gif, .ico, .jpg, .odg, .odp, .ods, .odt, .pdf, .png, .ppt, .swf, .txt, .xcf, and .xls.
- In the Legal Extensions parameter of the System tab in the Global Configuration Manager, the file types allowed for upload can be customized by adding or removing a file extension type.
- By default, newly uploaded files are placed into the \images directory.
- On a Linux server, the path to this directory will appear something like this:

  `/home/username/public_html/images/`
- On a Windows staging server, the path to this directory will appear something like this:

  `C:/Program Files/Apache Software Foundation/Apache2.2/htdocs/images/`

Fig. 6.13: The Media Manager

- Images inserted into an article are generally held in the \stories subdirectory.
- Therefore, the path to an image used in a story will have a path something like this:

```
C:/Program Files/Apache Software Foundation/Apache2.2/
htdocs/images/stories/ houseicon.png
```

- The Media Manager will allows the users to create a new folder by entering the name of the desired folder in the text box that appears to the right of the current path and clicking the New Folder button.
- Any of the media uploaded through this interface is accessible for insertion into article content.
- The Image button that appears at the bottom of the Joomla editor window allows to upload selected images.
- The relative path of the selected image will be stored with the article.
- Therefore, the HTML reference to use the previously mentioned image might look like this:

```
<img src=http://www.example.com/images/stores/houseicon.png>
```

E. Trash Manager:

- Like most desktop operating systems, deleted content is not immediately deleted from the Joomla system.
- When you delete an article from Article Manager, it doesn't get deleted entirely from your Joomla site. The item is relocated to the trash container.
- Joomla keeps these deleted articles just in case you need them again in the future.
- You can actually restore and publish the article again, or you can delete it completely from your site.
- Generally, the act of emptying the trash regularly is not performed by user, as the user tends to forget it, hence to be sure about it adding it to the administrative to do list makes the job easy.
- Evacuating the contents from the trash will free up space and available resources and is a better idea from security point of view.
- If there are huge contents in the trash, the administrator tends to simply empty the trash with examining the contents.
- Conversely, if there are few contents, the administrator tends to glance at the contents before deleting and restores contents which will be needed in future.

6.2.4.3 System Administration for Joomla

- The system administration in Joomla is responsible for the day-to-day operations and maintenance of the website.
- Activities might include ensuring that the site is properly backed up; managing user access; possibly installing extensions and maintaining the security of the website. The system administration has more settings than other managers.
- Smooth functioning of the Joomla site with maximum capacity is essential as millions of user's visit and access sites. Hence proper understanding of the configuration settings is critical and most importance.
- The Global Configuration Manager holds maximum of the global settings for the site, system, and server.
- The User Manager deals with the administration of the user accounts.
- The Menu Manager permits for creation and editing of menus as well as the menu items used by each menu for easy and fast access of various options.
- The Extension Manager supports installation and removal of new add-ons, extensions and languages for enhancing the capabilities of Joomla.
- The Module Manager, Plugin Manager and Template Manager provide management functionality for each of their specific add-on types.
- The Mass Mail Manager lets the administrator create a bulk mail communication to either selected user groups or all users of the system.

- The Control Panel acts as a home page for the administrator portion of the Joomla site and provides a good launching point for examining the system options.
- It provides the main overview page for navigation to site controls and content creation.

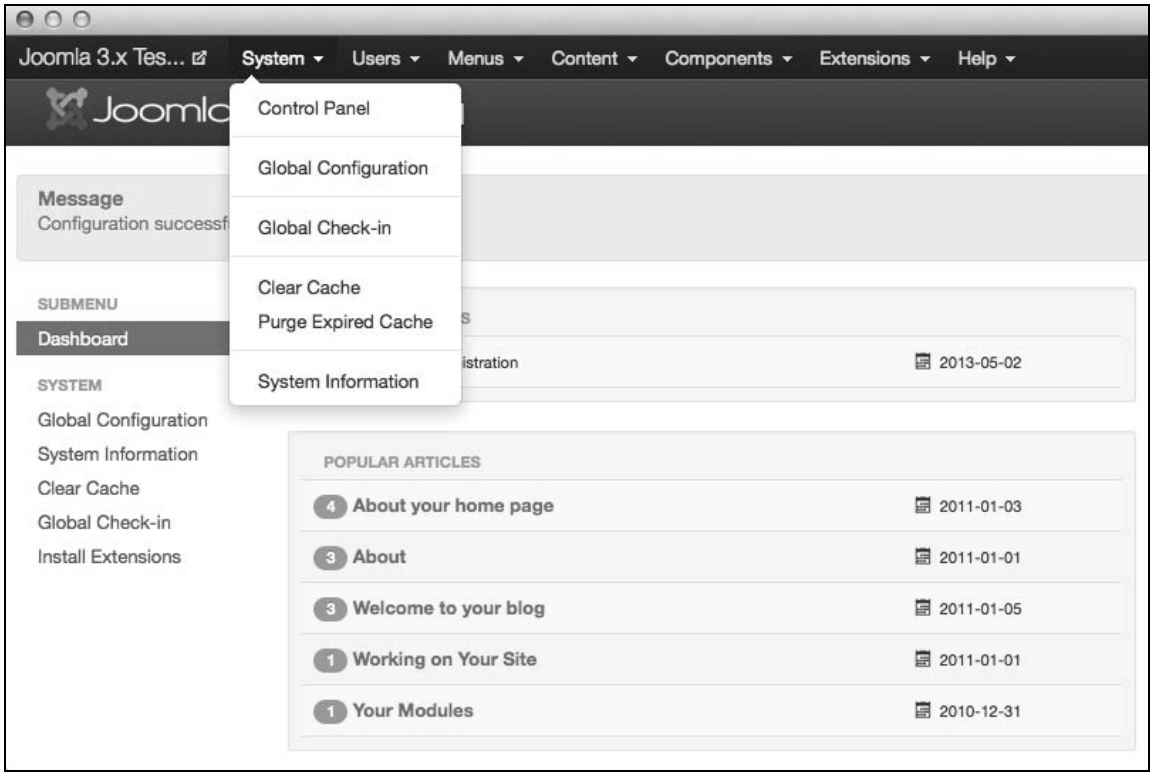


Fig. 6.14: The 'System' Admin Menu

A. Control Panel:

- The Control Panel is the home page of the Administrator interface and it provides a centralized panel where the administrator can jump to the most common parts of the site.
- The Control Panel provides access to many default Joomla! functions and features through a drop down menu bar, general notification area and clickable icons.
- A short list of some of the features from the Control Panel is:
  1. Create articles, categories, web links, modules and menus.
  2. Manage articles, categories, components, modules, templates, plugins, and menus.
  3. Install templates, components, plugins, and modules.
  4. Upload files such as images with the media manager.
  5. Update Joomla core software and installed templates, plugins, components and modules.

- Some of the direct links for quick access are to the Media, Menu, User, Module, Extension, and Language Managers as well as the Global Configuration.
- While at first glimpse the page may disclose only a number of navigation buttons, but there are three useful items on the right side of the Control Panel that are often unnoticed by Joomla webmasters - the Preview button, introductory text removal instructions, and the administrative panels.
- As you can see in Figure 6.15, the Preview button which is available on the toolbar provides a hyperlink to the Frontpage of the Joomla site for quick access.
- You can use this Preview hyperlink to open the home page in another window so that any changes made through the Administrator interface can be quickly evaluated.

Fig. 6.15: The Preview button will take you out of the Administrator interface and to the site Frontpage

- At the bottom of the page, you'll see that there are instructions showing how to delete the introductory message (see Figure 6.16).
- The administrative panels (also shown in the figure 6.16) provide useful information such as the identities of logged-in users, the most popular articles on the site, a list of newly added articles, and general menu statistics (the number of items present on each menu).

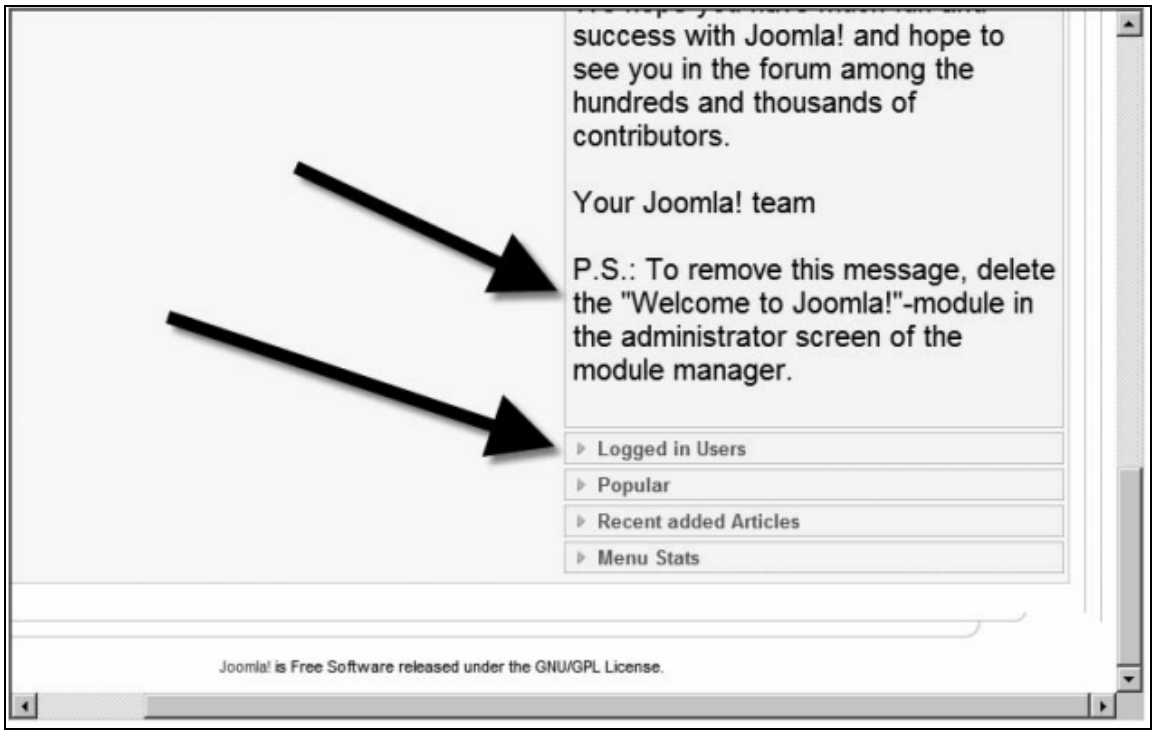


Fig. 6.16: Instructions on how to delete the Hello Message appears directly above the administrative panels

B. Global Configuration Manager:

- The Site menu provides the Global Configuration option under the Global Configuration Manager, embraces the general site wide settings.
- These settings help set up everything from the administrator password to the FTP upload capabilities.
- Global configuration is actually divided into three areas: Site, System, and Server.
- These panels are displayed by clicking the appropriate link under the Global Configuration banner.
- By default, the Site settings are displayed when the manager is initially presented.

1. Site Settings:

- Many of the options which are configured during the initial installation are available on the site screen.
- Further settings include Search Engine Optimization (SEO) settings, metadata for the site and feed settings (Fig. 6.17).
- Whenever the site is out of action, the Site Settings panel allows to take the server offline and set the message to the visiting browser.
- Whenever the database server is shut down for maintenance, this option is very useful as the message will ensure that your site hasn't vanished whenever visitors attempt to access it.

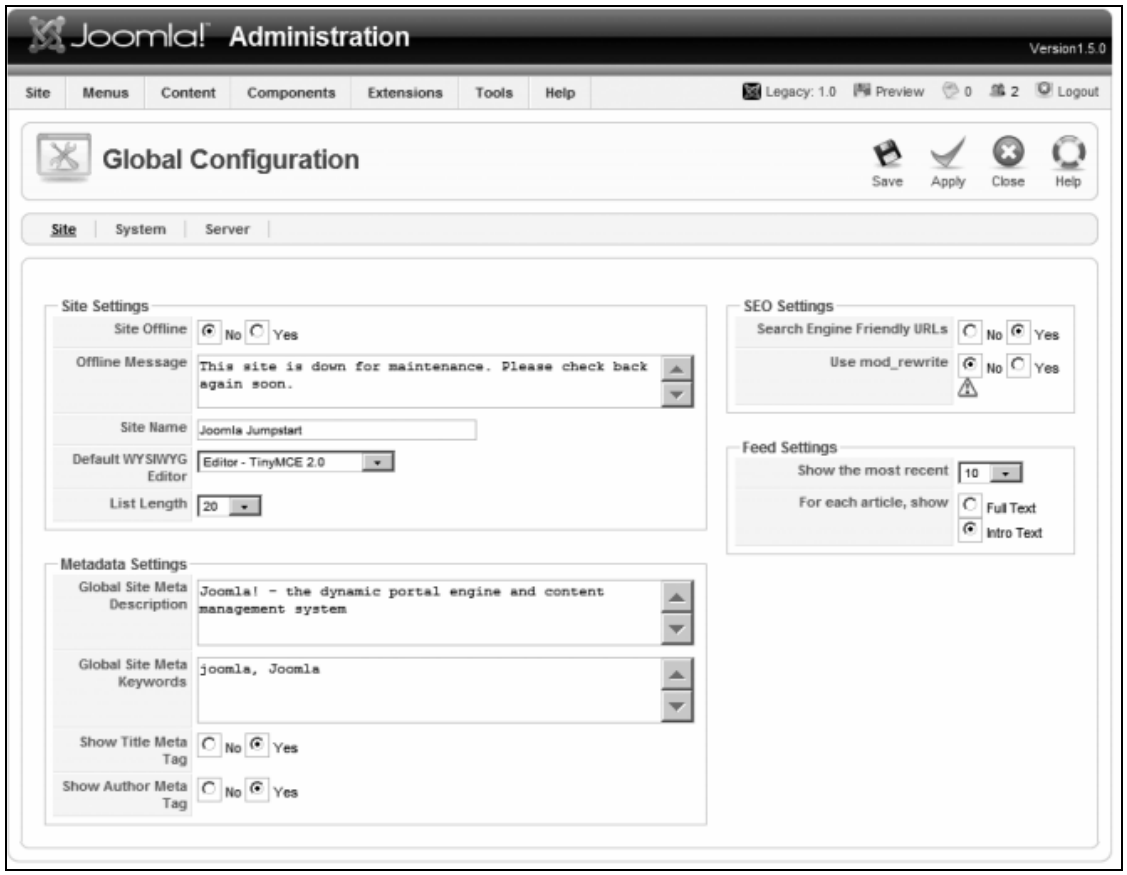


Fig. 6.17: The Site screen within the Global Configuration Manager

2. System Settings:

- The Site configuration determines how the site functions on the system, while the System configuration screen (see Figure 6.18) holds settings that affect the system itself.
- Many of these parameters affect performance, so the system should be monitored closely after any modification.
- A majority of the system settings are self-explanatory, but a few may be confusing to the Joomla beginners.
- The Debug settings have major performance and security consequences for a Joomla site and are rarely activated on a deployment server.
- The page cache can be turned on with the help of Cache settings.
- It simplifies the execution process so that the load of database is reduced by not constantly querying to generate the page to send to the user's browser.
- Caching can produce significant performance increases on a popular site.

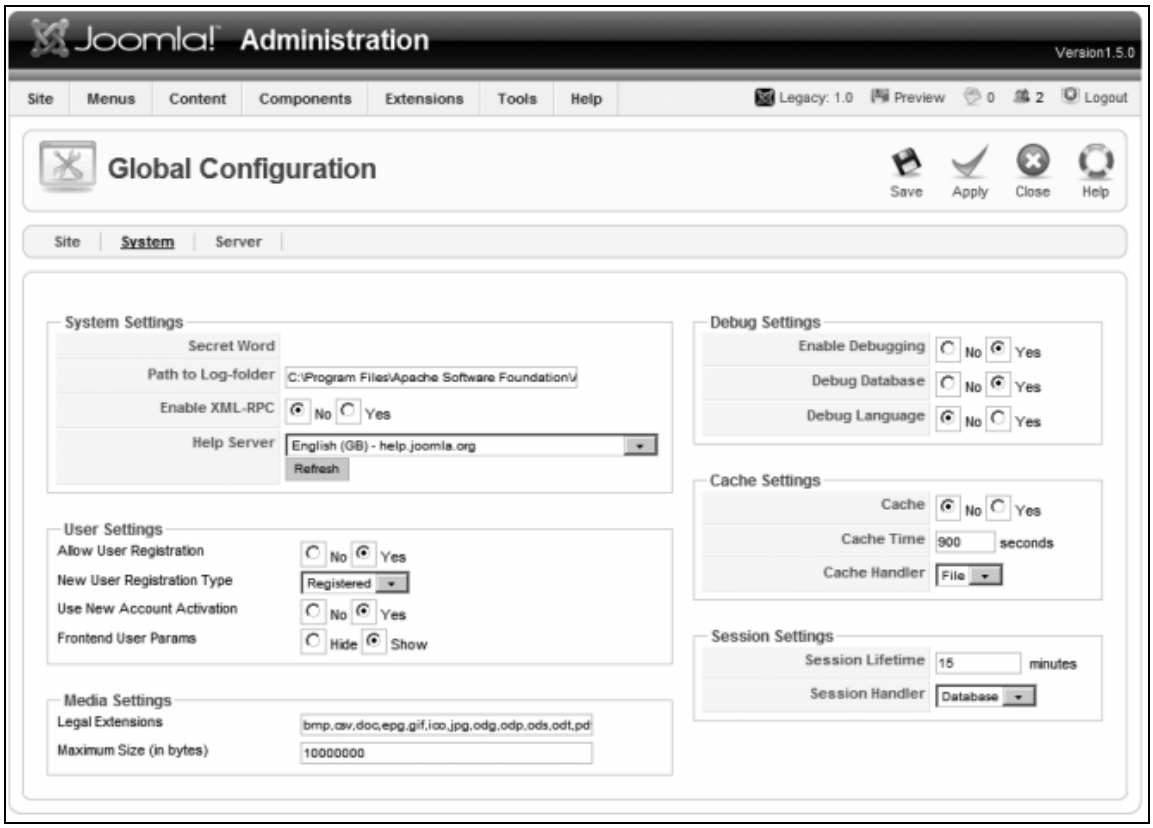


Fig. 6.18: The System Settings panel within the Global Configuration Manager

3. Server Settings:

- The last and final pane of the Global Configuration manager is responsible for the Server settings (Fig. 6.19).
- It is accountable for the configuration of the Joomla server functionalities and its association to the other servers.
- The GZIP function performs on-the-fly compression of the page requested by the browser and sends the file to browsers capable of decompressing it.
- The server settings activate GZIP page compression if the PHP server has the feature available.
- During Joomla installation, the installer checks for it and flags you if it isn't active.

Fig. 6.19: The Server Settings panel within the Global Configuration Manager

C. User Manager:

- The User Manager provides the ability to look at a list of users and sort them in diverse ways.
- Users, groups and access levels can be created and edited.
- The administrator can grant and revoke privileges for individual user accounts. (See Fig 6.20).
- Multiple users publish content on Joomla site; hence user security is configured as per the needs of online publication.
- Whenever a user registers with the system, a confirmation message is sent and after account validation the user is positioned in the registered user group.
- A registered user, after logging in to Joomla site is provided with two options: Edit Account Details and Submit Web Link.
- Users with Author security level and above are allowed to submit new content.
- The User Manager provides the following features: Registration Configuration, Login Security and Managing Lost Password.

Fig. 6.20: The User Manager maintains the user login accounts

1. **Registration Configuration**:

- The site panel of the Global Configuration Manager and the Configuration button in the User Manager allows users to register without administrator approval.
- The settings available for the registration system are displayed after clicking the Configuration button (Fig. 6.21).

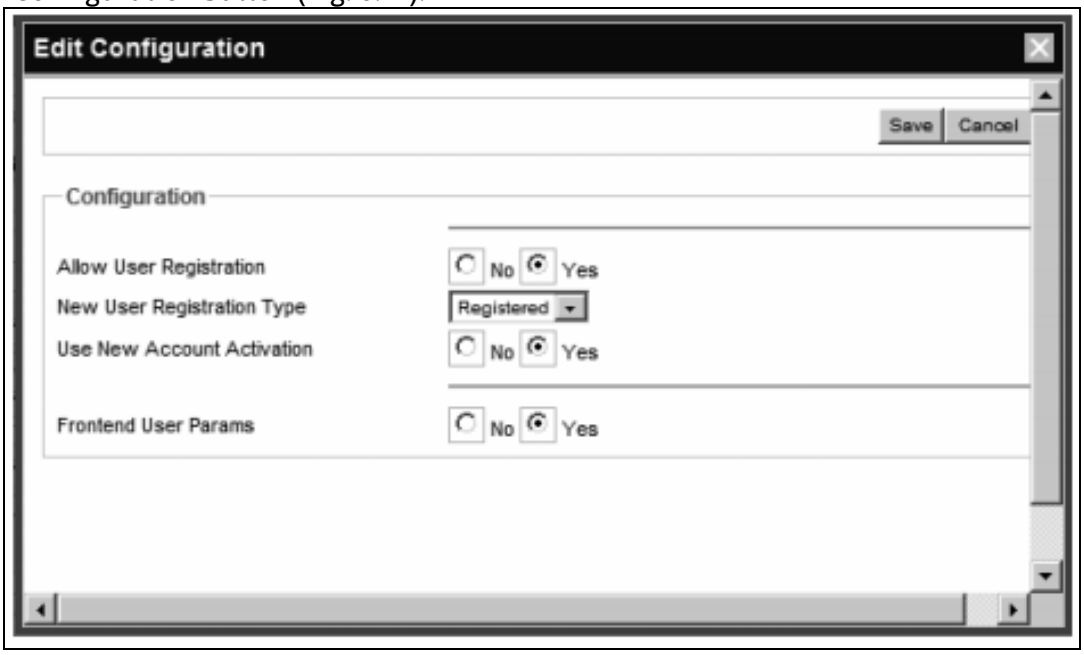


Fig. 6.21: In the User Manager, clicking the Configuration button will display the Edit Configuration window

- The administrator can easily alter all settings pertaining to the user account, as well as disable or delete the account if necessary.
- The Filter drop-down lists on the right side of the User Manager allows the user to separate out all but logged-in users or display the users based on the group to which they belong.
- Note that unlike other security systems, Joomla users may not belong to more than one group at a time.

2. Login Security:

- Anonymous access for all users and primary login security for registered users are provided by Joomla.
- The types of users who access the Joomla site are categorized into three basic groups.

1. Unregistered users:

- These users are the visitors to the web site who haven't logged in and may not be registered.
- They can also be termed as public front-end users.
- Many small and simple websites don't have a registration system or module, so all such users fall into this category.

2. Registered front-end users:

- These are the users or readers of your site who gain access to limited and restricted content by logging in.
- Generally, a registered user account is activated after filling a registration form, receiving confirmation e-mail, or they are manually confirmed by the administrator.
- Content on a Joomla site can be restricted to registered users.
- Some paid sites offer articles to only subscribed users who pay monthly, quarterly or yearly fee.
- Registered users may be authorized to donate or contribute content to the site, but adding new content is the limit of their permissions.

3. Registered back-end users:

- The system administrators, contributors, or moderators have the privilege to log in and alter core portions of the site.
- The permissions available to them determine their ability to make changes to the site.
- These users have access to the administrator back-end.
- When you edit a user account in the User Manager, as shown in Figure 6.22, you can see these three categories present in the Group list box.
- Two of these general categories (registered front-end and back-end users) have subcategories that further define the privileges of the user account.

Fig. 6.22: Editing a user record from the User Manager allows the administrator to assign the user to a group.

3. **Managing Lost Password**:

- If a user loses or forgets their password, the Joomla interface allows to place a request and a reminder is sent to the registered account's e-mail. Passwords are stored encrypted in MD5 format, so they cannot be recovered easily.
- Passwords are reset by administrator only, in case if they are lost.
- The new password will only be sent to the e-mail address that was registered with the account.
- If the user has closed down that account or is no longer able to access it, the System Administrator must be contacted to do a special individual reset.

6.2.5 Working with Joomla

6.2.5.1 Adding Articles

- Articles are information which consists of some text, images and links to different resources. These articles can be created in Joomla by two ways:

After logging in as Administrator:

- In the main Control Panel, click on the *New Article* menu item

  Or

- Click the **Contents** → **Articles** → **Add New Article** menu item

Or

- Click the **Contents** → **Articles** menu item for opening the Articles Page. Then click the *New* toolbar button.

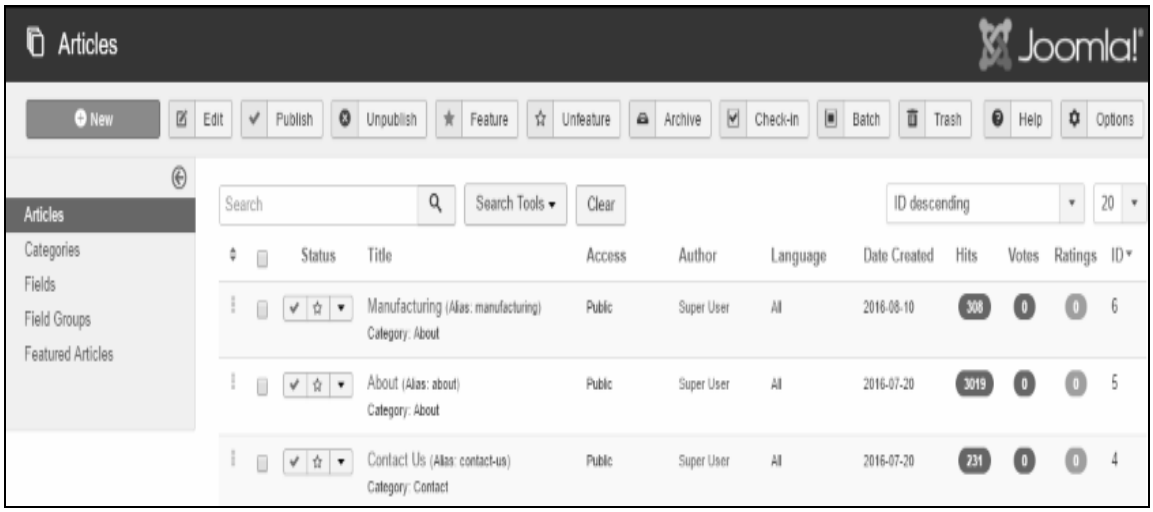


Fig. 6.23

The options for naming the article, categorizing, editing content and parameter selection are available in New Article screen.

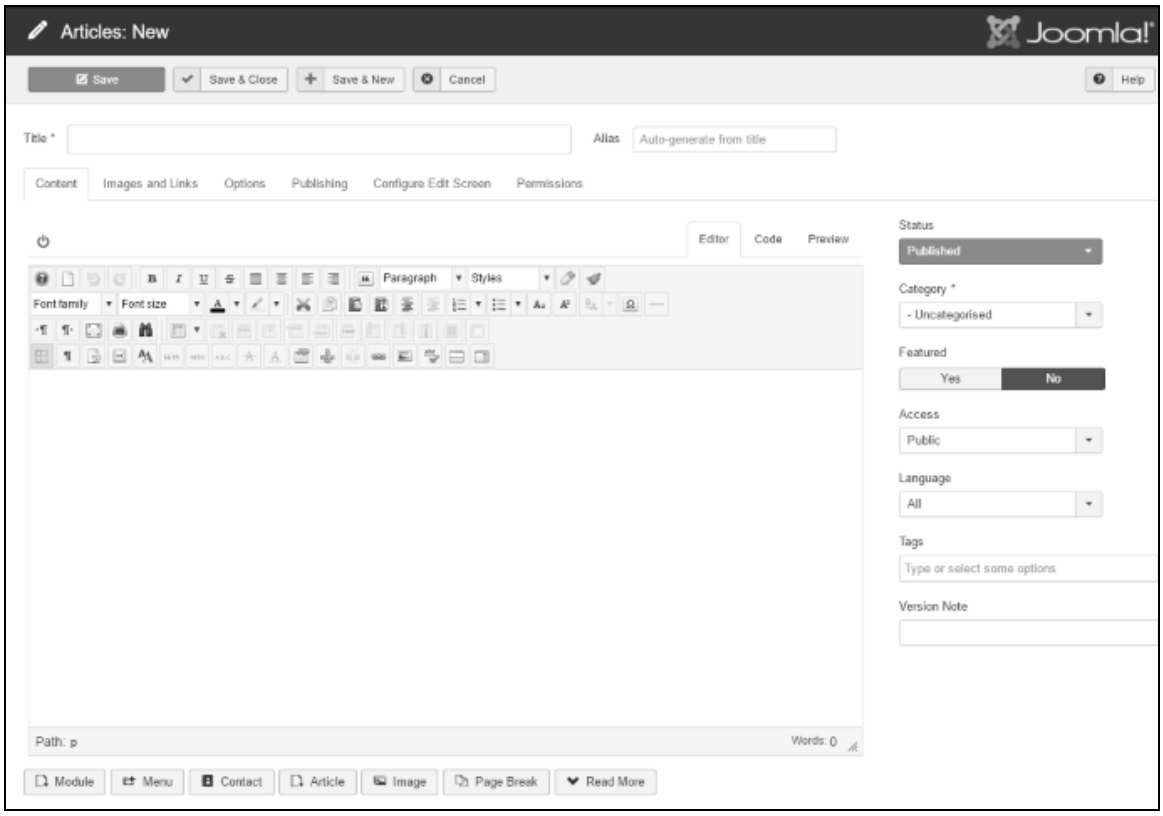


Fig. 6.24: 'New Article' page

For adding information to the article, follow the below mentioned steps:

1. Enter a title in the *Title* field. This is used to display the article's title.
2. Enter an alias in the *Alias* field. The alias as the name suggests is another name for title and is used to refer to the title if alias is not provided, Joomla generates it for the user.
3. The **Content** tab allows to set various options like :
   - Status: Published, Unpublished, Archived, or Trashed.
   - Category using the drop down menus.
   - Featured*:* whether or not to display the article on the home page.
   - Access restriction: Public, Registered, Super Users, etc.
   - Language the article is assigned to, if applicable.

   Additionally tags and version note can also be specified.
4. Various tabs are available to set several options or parameters as per user's requirements.
5. Finally, for saving the article multiple options are available :
   - **Save** toolbar button will save the changes, but remain on the Article edit screen.
   - **Save & Close** toolbar button will save the changes and return to the Article Manager screen.
   - **Save & New** toolbar button will save the changes and create another new Article.

6.2.5.2 Adding Menus to point to Content

- In Joomla!, a Menu is a set of menu items that aids easy website navigation. Each menu item specifies an URL to a page on the site, and hold settings that control the contents (articles, categories, modules, layouts etc.) of that page.
- For adding menus to point to already created article, follow the below mentioned steps:
  1. In the new browser window type in the URL, which will be similar to `http://www.your-site-name-here.com/administrator` or, if Joomla is installed on your local computer, then type `http://localhost/your-folder-name-here/administrator`.
  2. Login as administrator here and now you can carry out the process of creating a new item in the main menu that points to an article
  3. Log into the administration panel and from the menu bar select Menus.
  4. Choose the menu you wish to work with from the Menu item manager page or you can also select menu manager.
  5. A toolbar appears which will open Menu Item page after clicking on 'New'.
  6. From the Menu Item Page click on the Select list for **Menu Item Type**.

7. A list of articles will be displayed, from which user's desired article can be chosen and other required parameters can also be set.
8. At last, click the Save button which will save the menu item.

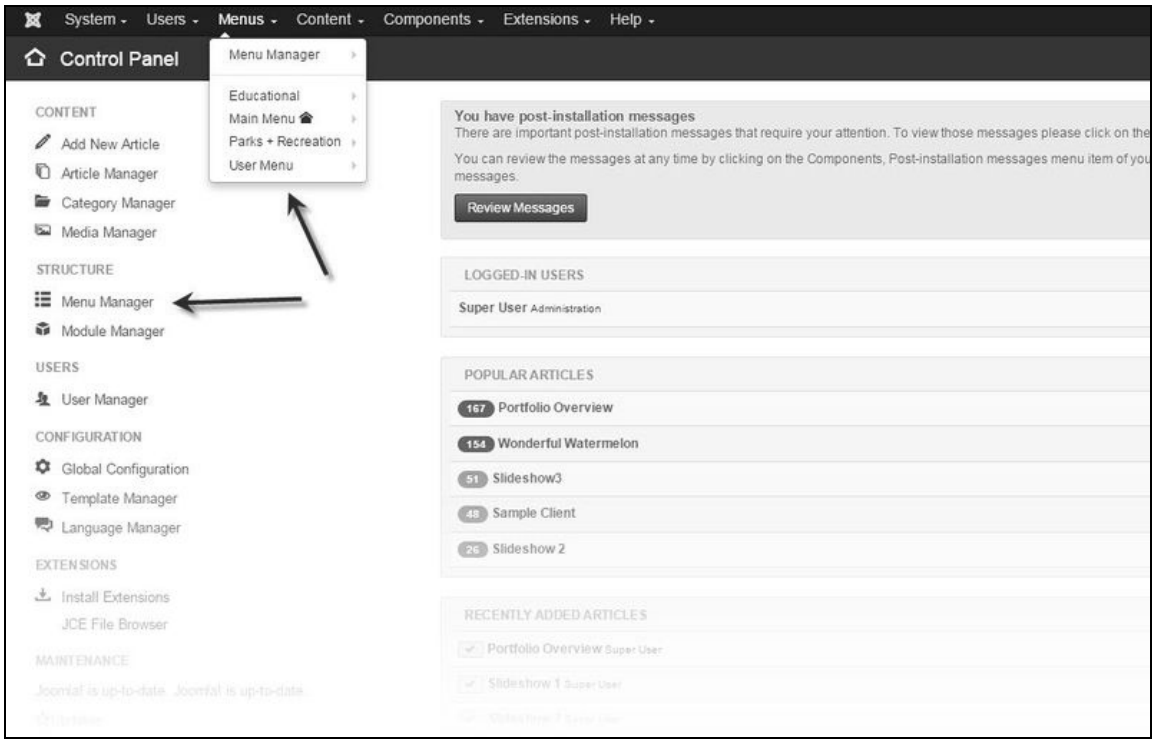


Fig. 6.25: 'Adding Menus' Page

6.2.5.3 Installing New Templates in Joomla

- A template controls the layout of a site. It is a type of Joomla! extension that changes the way your site looks.
- Steps for installing a template in Joomla is as follows :
  1. Login as administrator and first of all Click extensions, located on the top of the drop-down menu and then click "Manage" followed by clicking on "Install".
  2. Choose the option "Upload Package File", in the install page.
  3. Click on "Choose file" to upload your template installation.
  4. Then click the "Upload & Install" button after which the template is installed.
  5. To apply the default template for all the pages of your site, go to Extensions >> Templates >> Styles and here set your template to default by clicking on the star button.
  6. After installing the template, the pages and contents can be customized.

6.2.5.4 Creating Templates

- A template is responsible for managing the look and layout of a screen.

- It offers a framework which includes common elements, modules and components and cascading style sheet for a site.
- Following steps to be followed for creating templates in Joomla:
  1. **Create 'customtemplate' folder:** Go to Joomla installation folder → Templates → Create a new folder, for e.g. name it 'customtemplate'.
  2. **Create Folders for saving CSS and Images:** In the customtemplate folder → Create two folders viz.css and images. All the images used in template will be saved in the images folder only.
  3. **Create templateDetails.xml File:** A basic and mandatory file i.e. templatedetails.xml file is created whenever a Joomla template is created. It also corresponds to the basic template metadata. Custom position and options can be defined within the templateDetails.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<extension version="3.5" type="template">
<name>customtemplate</name>
<creationDate>2018-09-05</creationDate>
<author>TemplateToaster</author>
<authorEmail>example@templatetoaster.com</authorEmail>
<authorUrl>https://templatetoaster.com</authorUrl>
<copyright>Copyright© 2009-2018</copyright>
<license>GNU/GPL</license>
<version>1.0.2</version>
<description>My First Custom Template</description>
<files>
<filename>index.php</filename>
<filename>templateDetails.xml</filename>
<folder>images</folder>
<folder>css</folder>
</files>
<positions>
<position>breadcrumb</position>
<position>left</position>
<position>right</position>
<position>top</position>
<position>user1</position>
<position>user2</position>
```

```
<position>user3</position>
<position>user4</position>
<position>footer</position>
</positions>
</extension>
```

4. **Create index.php File:** In the newly created folder create 'index.php' file. Below is the basic code of Joomla template's index.php file.

```
<?php defined('_JEXEC') or die('Restricted access');?>
<!DOCTYPE html>
<html xml:lang="<?php echo $this->language; ?>" lang="<?php echo
$this->language; ?>" >
<head>
<jdoc:include type="head" />
<link rel="stylesheet" href="<?php echo $this->baseurl ?>
/templates/<?php echo $this->template ?>/css/template.css"
type="text/css" />
</head>
<body>
<jdoc:include type="modules" name="top" />
<jdoc:include type="component" />
<jdoc:include type="modules" name="footer" />
</body>
</html>
```

5. **Custom Images:** Any images to be added in the template can be done as follows :

```
<imgsrc="<?phpecho$this->baseurl; ?>/templates/<?phpecho$this->template;
?>/images/myimage.png" alt="Custom image" class="customImage" />
```

Here, the template variable will fill in the name of your template.

6. **Custom CSS:** Custom CSS can be added as follows :

```
<link rel="stylesheet" href="<?phpecho$this-
>baseurl?>/templates/<?phpecho$this->template;?>/css/styles.css"
type="text/css" />
```

Every file which is added must have a line in the templateDetails.xml file for the template, unless it resides in a sub-folder (which can be included in a <folder> element).

7. **Testing the template:** Find the template in the Template Manager, after selecting it, click on Default to make it the default template. In Joomla, the template can be

seen and accessed via Extensions -> Extension Manager -> Discover (i.e. the Discover tab). To find your template, Click Discover (i.e. the Discover button) and then install it after selecting. The template should show up in the Template Manager (Styles). Now you can create your template outside of Joomla and install it like any regular extension.

6.2.5.5 Adding a Module and Component

Adding a Module:

- Modules help to pull information from a website's database or from **Joomla** components. They can be placed in any predefined **module** position included with a **Joomla** template.
- Following are the steps for adding a module in Joomla:
  1. Login as administrator in Joomla.
  2. Go to Extensions->Modules.
  3. Click the 'New' button displayed in green color.
  4. A list of module types is displayed e.g. Action logs, Administrator, Articles, Custom etc.
  5. Choose the module to be added.
  6. Start creating the module as per your requirements.
  7. Save the module and the new module will now appear on the control panel.

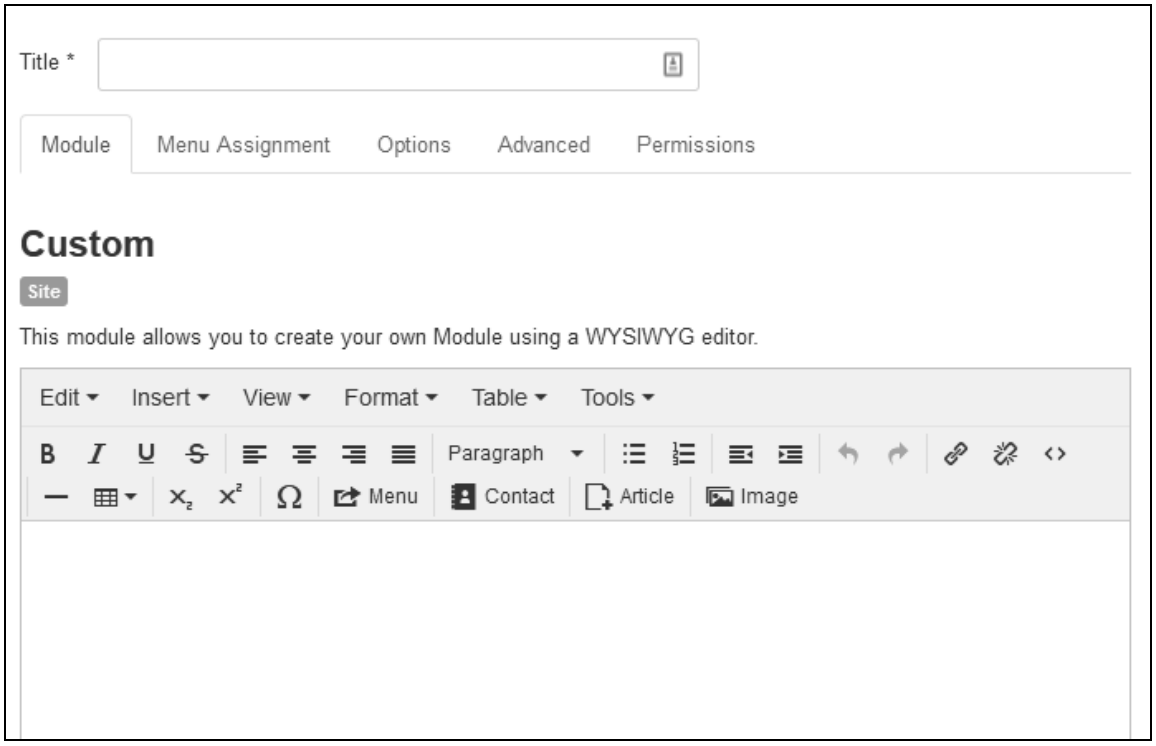


Fig. 6.26: 'Add Module' Page

Adding a Component:

- Components can be seen as Joomla extensions which are the core functional units of Joomla. They can also be termed as mini-applications. A compressed file containing all things which are needed for installing and uninstalling is used for installing extensions. For creating a Hello World! Component, follow the steps:
  1. Use your preferred file manager to create a directory for the Hello World! component.
  2. This directory can be place anywhere on your file system outside the Joomla installation directory. For this example, we will name the directory `com_helloworld`.
  3. Inside the directory some files need to be created using your preferred file manager. Using your preferred file manager, create the following files. Add the source code for each file as they are created.
     a. **`helloworld.xml:`** This is an XML(manifest) file that tells Joomla! how to install our component.
     b. **`site/helloworld.php:`** This is the site entry point to the Hello World! Component.
     c. **`site/index.html:`** Prevents web server from listing directory content.
     d. **`admin/index.html:`** Prevents web server from listing directory content.
     e. **`admin/helloworld.php:`** This is the administrator entry point to the Hello World! Component.
     f. **`admin/sql/index.html:`** Prevents web server from listing directory content.
     g. **`admin/sql/updates/index.html`:** Prevents web server from listing directory content.
     h. **`admin/sql/updates/mysql/index.html:`** Prevents web server from listing directory content.
     i. **`admin/sql/updates/mysql/0.0.1.sql:`** File allowing to initialise schema version of the `com_helloworld` component.
  4. For installing the Hello World! Component, create a .zip file of this directory using the preferred file manager. For this example, we will name the file `com_helloworld.zip`. After this the Hello World! Component has to be installed using the Extension Manager of Joomla!.
  5. Using your preferred web browser, navigate to the Administrator panel of your Joomla! site. The address would be `<yoursite>/joomla/administrator/index.php`. For this example we will navigate to `localhost/joomla/administrator/index.php`.
  6. Check Extensions → Manage → Install → Upload Package File → Choose File.
  7. Navigate and Select File.
  8. Click Upload & Install.
  9. A message will be displayed informing if the installation succeeded or failed.

- The basic function of the component can be tested by entering the Hello World! page for the site and administrator portions of your Joomla! website.
- Using your preferred web browser, navigate to the Hello World! component page located on the site portion of your website.
- The address would be `<yoursite>/joomla/index.php?option=com_helloworld`. For this example, we will navigate to `localhost/joomla/index.php?option=com_helloworld`.
- The Hello World! component is now visible in the administrator site of your Joomla installation under the *Components* menu.

6.2.5.6 Modifying the Existing Templates

- Templates are a collection of XML, PHP, HTML and image files which are stored in the *templates* directory of your site. These files can be edited directly or the Template Manager can be used.
- Following are the steps to edit/modify an existing template:
  1. Login as administrator in Joomla
  2. Click 'Extensions' and then click 'Template Manager' in the top menu and then select 'Templates' menu.
  3. A listing of installed templates will be displayed. Select the template you want to edit, and then click the *Template* Details and Files link and choose the file you wish to edit.
  4. You are now on the *Customise Template* page. Here you can find the Temp*late Master Files* and the *Stylesheets*. Click any of the files Template Master Files and Stylesheets .Edit the corresponding file. Below is a list of the files that can be edited:

     Edit main page template: index.php
     Edit error page template: error.php
     Edit print view template: component.php
     Edit css/template.css: css/template.css
  5. After editing the file, make the changes permanent by clicking on Save.

6.2.5.7 Creating Templates with Web Editors

- A web editor manages contents for a site. Many editors are available online for joomla which makes the management of contents easy and simple.
- Some of the popular editors for Joomla are :
  1. JCE editor
  2. ARK editor
  3. DropEditor
  4. NextGenEditor
  5. RokPad
  6. Jodit Editor

- Let us consider JCE editor for our example and see how a template can be created in this editor.
- The Template Manager available in the JCE editor is able to create new templates from existing content, that is, the contents of the article you are editing/creating when the Template Manager is open.
- A template can be created from a selection i.e. part or from the whole article.
- When a part of article is required, select the desired content before opening the Template Manager. This is not necessary if you are using the whole or full article.

Creating a Template

- Once the Template Manager plugin dialog is open, click the New Template icon. The Create Template dialog box will be opened.

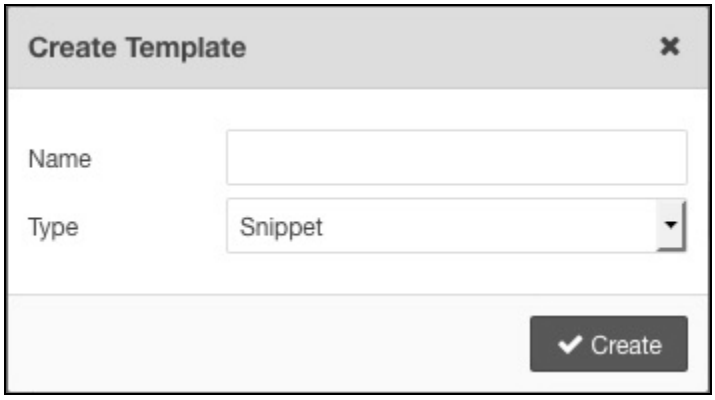


Fig. 6.27

- Provide the below mentioned details:
  - **Name** - The name of the template.
  - **Type** - The type of 'template' to create, either a Template or a Snippet.
- A Template wraps your content in a div with a unique id, so that it can be identified.
- Templates can be replaced by other Templates or Snippets using the Template Manager.
- Snippets are small piece of text which can be included in content.
- Templates can affect a selected or part of content but this is not applicable to snippets as they can only be inserted in article content at the current cursor position.
- The selected article content will be altered by whatever html code is contained in the Template that has a class included in the Content Classes' list in the Template Manager Configuration.

- For example, selected content will be given a yellow background by the following html in the Template:

```
<p class="selcontent" style="background-color: yellow;">Selected Content</p>
```

- Example of Template html ,

```
<div class="mceTmpl">
<p class="selcontent" style="background-color: yellow;">Selected Content</p>
<p style="padding: 3px; color: #666; border: 1px solid #CCC;"><br /><strong>Editors Comment</strong> (<span class="cdate"></span>)<br /><br />[Insert your comment here]</p>
</div>
```

6.2.5.8 Creating Real Templates

- In the previous section (Creating templates in Joomla) we learnt the basics of creating a template from scratch.
- Most templates are built from an existing base or prototype template.
- It is preferable and much easier to use a template with basic and essential utilities and then later augment features as required by users rather than customizing a full-fledged template.
- For most templates though three primary files define how the site will appear: `templateDetails.xml, index.php` and `template.css` (discussed above)
- Other than the basic three files, most templates also include the following:
  - **Thumbnail graphic file:** This file is situated in the root of the template directory; this file is a 140-by-90 image with a filename of `template_thumbnail.png`. This file is displayed by Joomla as a template preview when a mouse-over event occurs over the template name in the Template Manager of the Administrator interface.
  - **CSS directory:** Style sheets used by the template are stored in a separate directory named \css. The main style sheet of the template typically has a filename of template.css.
  - **Images directory:** A separate directory named \images contains any graphics used by the template.

Steps for Template Creation

- A Joomla template is an amalgamation of three key elements: graphics, PHP/HTML code, and one or more style sheets. By methodically working through the process of creating a template, you will have to follow below mentioned steps when you want to make a new template for your future needs.
- You can produce a new template by following these steps:
  1. Choose a color scheme for the site.
  2. Create style sheets that match the primary color scheme.
  3. Choose a font scheme that flatters the content.

4. Create the banner graphic.
5. Create the `index.php` file.
6. Create the `templateDetails.xml` file.

- Once you have implemented your basic template, you can easily upgrade it in the future.
- Since all web pages in Joomla are generated dynamically, changes you make to the template will be reflected instantly on every web page of your site.
- Within a few well-defined boundaries, any changes can be made to the template, and the web site will still function properly.

6.3 INTRODUCTION TO DRUPAL

- Drupal is an open source and free content management system used for building and maintaining websites.
- Many top businesses and organizations use Drupal. It has inherent benefits - cost, flexibility, freedom, security, and accountability - that are unmatched by proprietary software.
- Drupal is freely available to download and anyone can modify and extend the platform.
- It can easily and smoothly handle high traffic; therefore, it is widely used for enterprises and world's busiest websites.

6.3.1 Drupal Features

1. **Usage of HTML5:** HTML5, the most preferred markup language for designing web pages, is now available natively in Drupal 8, giving access to various input fields. It provides more functionality and compatibility with mobile and handheld devices.
2. **Multilingual:** Drupal supports multiple languages and possesses extensive multilingual features. The admin interface provides built-in translations and the pages can be created with language-based views filtering and block visibility.
3. **Configuration Management:** Carrying out configuration elements (like fields, views, content type, etc.) from local development to the server in Drupal has become an easy task. A version-control system can be used to keep track of configuration changes. Configuration data is stored in files, separate from the site database(s).
4. **Easy Editing:** With the WYSIWYG editor, the content editor has gained extraordinary power using which editing and content creation has become super easy. Site and content creators or editors can edit text on any page without having to switch to the full edit form, also drafts are now much easier to create.
5. **Better Support for Accessibility:** Rich Internet applications possessing better font sizes, tweaked color contrasts, jQuery UI's autocomplete, and modal dialogs which are as per industry standards are well supported by Drupal.

6.3.2 How Drupal works?

- Drupal is compatible for enterprises that anticipate growth in traffic volume and functionality hence it provides flexibility for customization.
- The architecture of Drupal majorly consists of interlinked set of components which comprises of APIs, themes and modules and these can be used and customized by different developers for different purposes.
- A module contains code and files that extend Drupal's functionality.
- Modules are event-driven, which means they look or listen for events which are generated by APIs or other modules and accordingly they trigger action.
- Drupal is a set of API services that modules use to interact with content and with other modules.
- A module consists of code and files that extend Drupal's functionality.
- Modules are event-driven: they 'listen' for hooks, or events, generated by APIs and other modules which will trigger them into action. .

6.3.3 The platformComponents, Modules and Plugins for Drupal

Components:

- Stand-alone Rules configuration which can be typically called as Components are that can be called from response or reaction rules, other modules and also programmatically from custom code.
- Components may be useful in the following cases:
  - Use of modules that make use of components, such as Views Bulk Operations or Rules Scheduler.
  - Occurrence of Some complex actions or conditions used in multiple places in your configuration, and want to manage them all in a single place.
  - Writing a module that uses configurable actions or conditions, and you don't want to write the user interface from scratch.
  - You are exporting Rules configuration, and you want fine-grained control over the exports.
  - You have complex actions which you want to run manually every once in a while.

Modules:

- A Drupal module is a compilation of files which consists of some functionality and is written in PHP.
- Because the module code executes within the domain of the site, it is allowed to use all the functions and access all variables and structures of Drupal core.
- In fact, a module is very much similar to a regular PHP file that can be independently created and tested and then used to drive multiple functionalities.

- Few popular Drupal modules are Chaos tool suite, Views, Toke, Libraries API, Entity API and many more.

Plugins:

- Plugins are very much similar to PHP native interfaces.
- The plugin system has the ability to discover every implementation of an interface (the default is magic namespacing), deals with metadata and provides a factory for the plugin classes.
- The Drupal Plugin API allows a module or subsystem to provide functionality (plugin instances) in an object-oriented way.
- Plugins that perform similar functionality are of the same plugin type.
- Module developers write plugins to extend various systems like blocks, field widgets, and image effects and add new options for administrators to choose from.

Software Requirements for Drupal:

Browser Requirement: The latest release of each of the latest two supported major versions of:

Desktop browsers:

- Google Chrome
- Firefox
- Safari
- Microsoft Edge
- Opera

Mobile browsers: Safari for iOS.

Web Server requirement: Drupal 8 and later versions work on any web server with a version of PHP that meets the PHP version requirements. It works on Apache, Nginx and other web servers.

Database requirement: PostgreSQL 9.1.2 or higher, MySQL 5.5.3 or higher, SQLite 3.6.8 or higher, Microsoft SQL Server and MongoDB are also supported by additional modules.

PHP requirement: Use PHP version 7.3 or higher for current release versions of Drupal 7, 8, and 9.

6.3.4 Administering Drupal

- Once Drupal is installed, you can perform administrative task such as configuring modules, working with content types, setting up site navigation, manage users, add additional functionality etc.
- This administrative account is automatically given all privileges for managing content and administering the site.
- Once you have logged into your site, set of toolbars appear at the top of the site which guide to perform various tasks.

- The toolbar helps to easily navigate between home icon, dashboard, content, structure, appearance, modules etc.

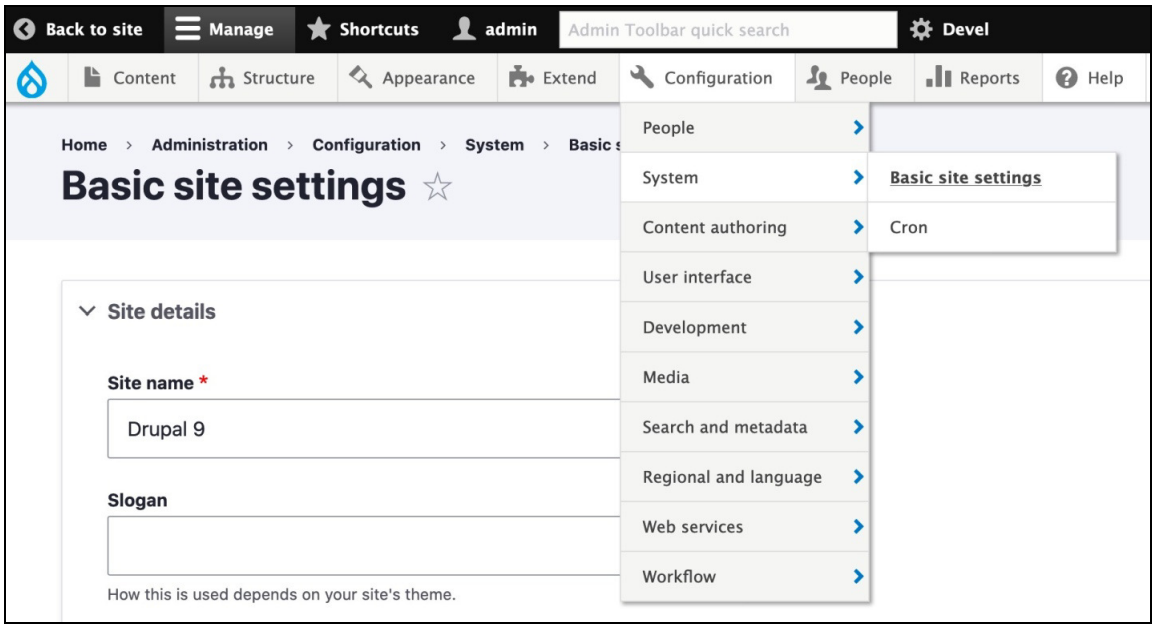


Fig. 6.28: The 'Drupal Administrative Toolbar' Page

- The **dashboard** on the Administration page gives administrators a customizable overview of important site information.
- You can add and remove items from the dashboard, or you can disable the dashboard completely.
- It provides a summary of what's happening on your website.
- The **Administrative module** provides a theme-independent administration interface which helps novice users .providing user friendly and easy access to site.
- It is a time-saver for site administrators, and useful for developers and site builders.
- **Administrative links** are displayed as menu at the top on all pages of your site.
- It consists of menu items for carrying out numerous tasks and actions which enables fast access to any administrative resource in your Drupal site.
- The **Admin toolbar** which is placed at the top of the site provides fast access to all administration pages.
- This toolbar gives access to the Content, structure, appearance and configuration of the site.
- Hence, the overall presentation of site, contents of the site and system settings can be controlled from this administrative area.

6.3.4.1 Presentation Administration

- The presentation administration deals with the look and feel and layout of a site. The color schemes, placement of various HTML controls, sections and structures in the page are all dealt in the Presentation Administration. This administration deals with the following concerns:

Customizing site appearance:

- Site appearance refers to how you make the website look.
- It deals with the colors, navigation, themes, etc.
- Any existing site or a new site can be customized for enhancing the appearance of a site.
- On the *Appearance* page via *Menu -> Appearance* (admin/ appearance) a site's appearance can be changed by installing new themes or editing theme settings.
- Drupal by default provides a small number of themes.

Managing site structure:

- Site structure refers to how you organize your website's content.
- A website often consists of contents on a variety of related topics, presented on posts and pages.
- Site structure deals with how this variety of contents are grouped, linked and presented to the visitor.
- For altering the arrangement of a site, open the *Structure* page via *Menu -> Structure* (admin/structure) to change the Structure of the site's content types, Blocks layout, Menus, Taxonomy, Views, Display modes, and Contact form by managing appropriate changes in the site.
- Drupal by default provides some structured components of sites.

6.3.4.2 Content Administration

- This module provides a dynamic interface for content administration.
- It is intended to supplement Drupal's built-in content management page, and features these improvements over the standard interface.
- Uses AJAX to perform searches, apply filters, and retrieve the results without reloading the entire page.
- Adds filters by author and text content (using Drupal's built-in search).
- Shows how many nodes exist within a given set of filters, and allows direct navigation to any page of results.
- The 'administer nodes' access permission is required for the built-in content management, which grants access to a broad range of content-related actions. This module's page is accessible to users who have received permission for only that purpose. Through this page, standard content permissions are enforced, so content can be viewed, modified, or deleted only by users who have these permissions.

- JavaScript is required for most of these features. Also, this module requires PHP version 5.2.

Installation:

- Download the package and extract the files to your modules directory.
- Enable the module on the Modules configuration page.
- Grant the appropriate users permission to 'access content administration' on the Permissions page.

Usage:

- Go to the Content Administration page provided by this module under the Administer -> Content Management menu.
- You should see a set of characteristics by which to filter nodes, and a list of nodes with icons to view, edit, delete.
- Clicking on these buttons will display the node rendered, an edit form for the node, or a confirmation form to delete the node.

Working with Search:

- The search module allows users to search for precise and specific content on your site.
- Search can be done for both users and for particular or selected words.
- When you are on the "content" tab of Search, you will be able to search for words appearing in the default rendering of node content on your site, Location fields, Taxonomy, etc., as well as comments.
- When you are on the "users" tab of Search, you will be able to search the user's names of registered users on your site, and if you have sufficient permissions and can also view their email addresses.

6.3.4.3 System Administration

- The system administration deals with managing users and their accounts, configuring the site's information, adding functionalities by enabling modules and many more.

Configure site information:

- The site information comprises of sites name, contact details, slogan of website, default time zone, versions of software used, content type etc.
- For carrying out the site configuration, open the *Site information settings* via *Menu -> Configuration*

  -> System -> Basic site settings (admin/config/system/site-information) for carrying out basic settings, such as the site name, slogan, e-mail address or the default front-page path (Fig. 6.29).

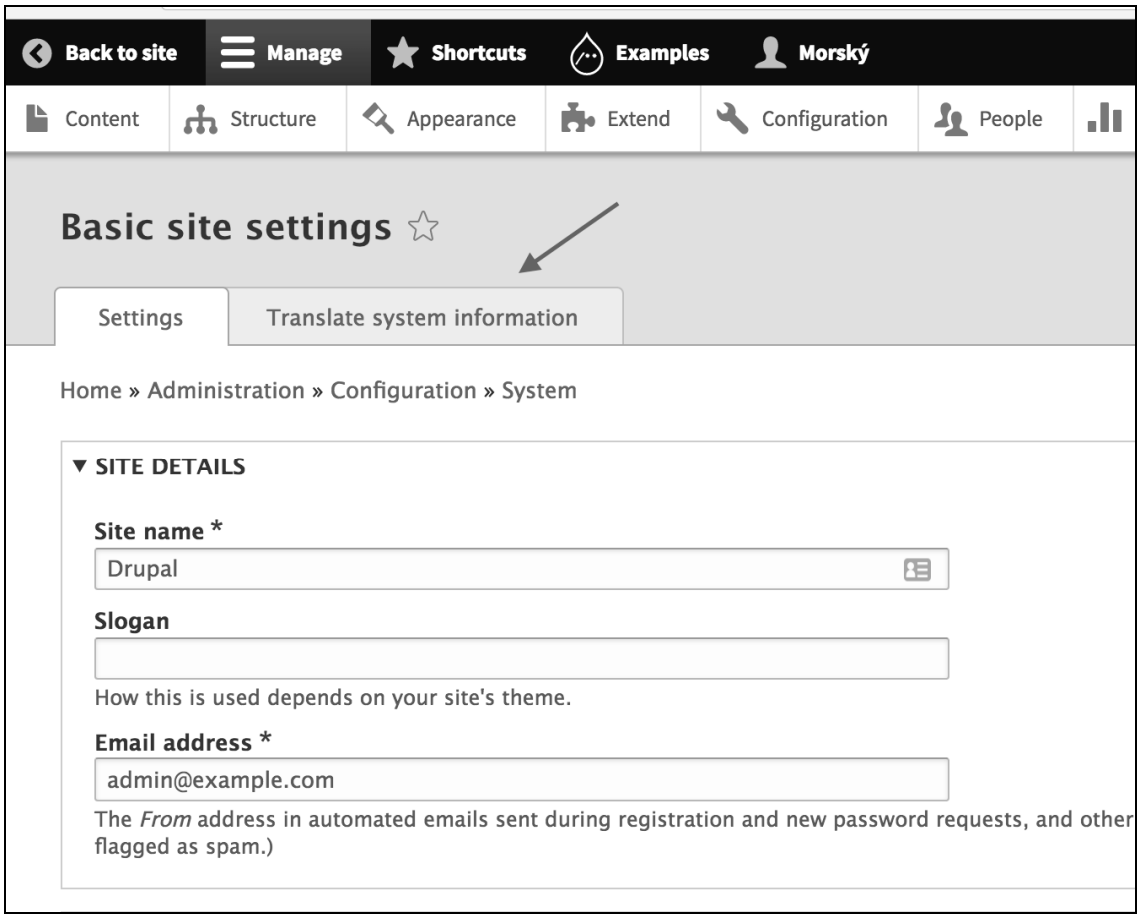


Fig. 6.29: Configuring Site Information

Manage users:

- User management is a significant task which helps to identify the different users and their privileges.
- It manages the information of the user which allows creating or deleting the user, changing passwords, time and roles.
- Open the *People* page via *Menu > People* (admin/people) to add new users or manage existing users.
- You can manage user roles and permissions by clicking on the "Permissions" tab on this page.
- To change the process by which users apply for accounts, visit the "People and Permissions" page via *Menu > Configuration > People > Account settings* (admin/config/people/accounts).
- The Permission relates to posting content permission. It contains the following types of permission – Block, Comment, Menu, Image, Filter, Node, Path, Toolbar, User etc.

- The roles allow setting permissions for group of users by defining their roles. Here you can create the roles and edit permissions for each user.
- The roles of user can be :
  1. **Anonymous user:** Allows user to access your website without asking them for the username or password.
  2. **Authenticated user:** Allows only those users to access your website, who are authenticated to use it.
  3. **Administrator:** User who is responsible to manage the complete website and allow users to create or view by his permission.

Add additional functionality:

- Drupal's functionality can be extended by enabling modules.
- The standard Drupal installation comes with a number of modules that are ready to be enabled.
- In addition, you can download community-contributed ("contrib") modules.
- Open the *Extend* page via *Menu > Extend* (admin/modules) to administer modules.

Configuration Management:

- In Drupal, configuration is the collection of admin settings that determine how the site functions, as opposed to the content of the site.
- Configuration typically includes things such as the site name, the content types and fields, taxonomy vocabularies, views and so on.
- Drupal stores site configuration data in a consistent manner, everything from the list of enabled modules, content types, taxonomy vocabularies, fields, and views.
- Making configuration changes on a live site is not recommended. The system is designed to make it easy to take the live configuration, test changes locally, export them to files, and deploy to production.
- Your site's configuration can be stored as part of your codebase and integrated with version control.
- By default, the "active" configuration is stored in the database ("config" table) for performance and security reasons. This is the complete configuration for the entire site at that moment.
- Configuration can be exported and imported as YAML files, either in its entirety, or a single piece of configuration, using **Drush** and/or **Drupal Console** config commands or the **Configuration Manager.**
- Exporting and importing configuration changes between a Drupal installation in different environments, such as Development, Staging, and Production, allows you to make and verify your changes with a comfortable distance from your site's live environment.

- This allows you to deploy a configuration from one environment to another (as a precaution, Drupal checks the site is the same before importing, by comparing its UUID).

Module and Theme configuration files:

- The default configuration transported with modules, distributions, and themes is imported into the active configuration store when the extensions are enabled. An extension's default configuration is found in the config/install directory.

How to import, export and synchronize?:

- With the Configuration Manager core module, you can import, export, and synchronize site configuration via Manage -> Configuration -> Development -> Configuration Synchronization (admin/config/development/configuration). Change can be reviewed before importing them.
- Either a single object can be imported or exported using a copy/paste workflow. This is useful in case, you wanted to just move a newly created view from one environment to another. Or the full site configuration can also be dumped as YAML files to a tar.gz file. This only works if you're moving configuration between two copies of the same site (e.g. dev and production) and for that reason, the sites UUIDs must match.

Internal Page Cache:

- Drupal provides an Internal Page Cache module that is recommended for small to medium-sized websites.
- This core module, which is enabled by default, caches pages for anonymous users. It can be found at **core/modules/page_cache**.
- This feature improves performance because it speeds up the site.
- Pages requested by anonymous users are stored the first time they are requested and then reused; depending on your site configuration, the performance improvement may be significant.

6.3.5 Working with Drupal

6.3.5.1 Adding Articles

- Articles can be thought of small stories or personal blogs.
- Articles may include information about new events or summary of activities which are already happened.
- To create a new article, content type article is used.
- To create an article, following steps are carried out:
  - Make sure you have logged in as user who has the privileges to create content.
  - Either from the administrator or navigation menu Select *Add content*.
  - A screen appears, prompting what type of content has to be added. Drupal comes with two content types Basic Page and Article.

- Choose Article. After which a form is popped which asks information of the Article.
- Enter the details viz, title, body, summary. You can also upload image (if required).
- At the end, Click on *Save* which provides you access to publish your content or keep that unpublished.

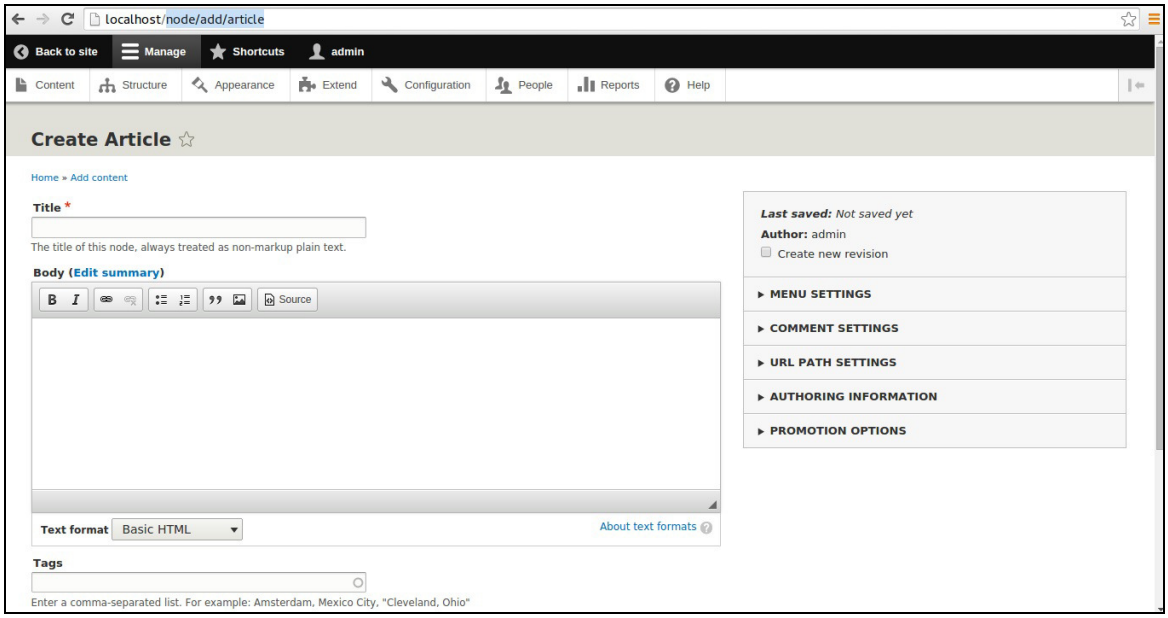


Fig. 6.30: Create Article Page

6.3.5.2 Adding Menus to point to Content

- For effective and easy navigation, menus provide links to various web pages.
- Menus provide a reference point to the content types like articles, blocks etc.
- There are five menus in which you can place menu items.
  1. **Main navigation:** They are created by site administrators. Usually they provide links to major sections of site.
  2. **Administration:** This menu consists of links for administrative tasks.
  3. **User account menu:** Links related to users account i.e. Log out and My Account etc.
  4. **Footer:** Created by administrators. They usually provide links to important pages within the site.
  5. **Tools:** They provide link to site visitors for necessary tasks.

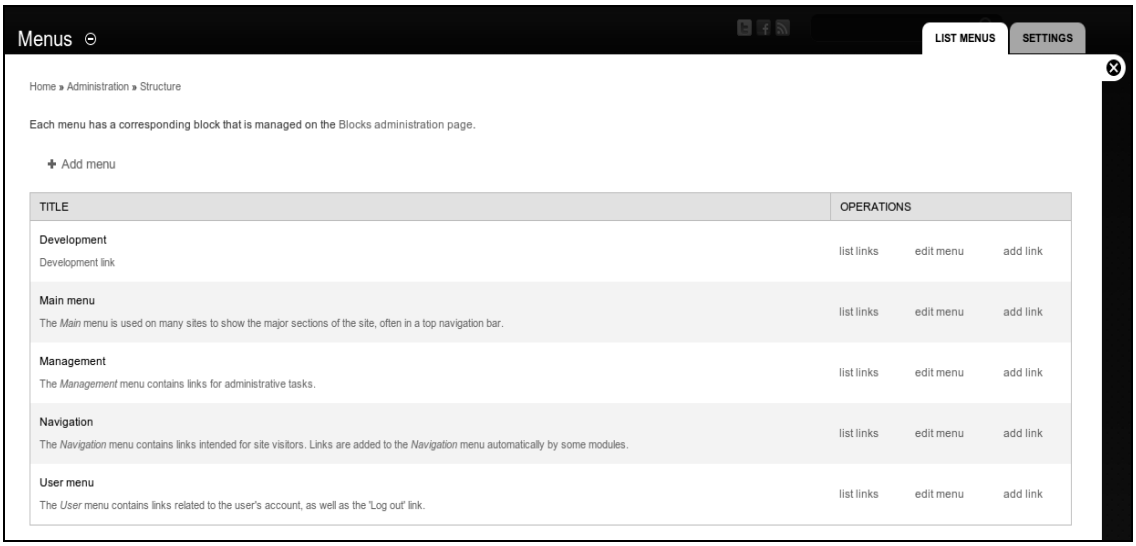


Fig. 6.31: The 'Adding menus' Page

6.3.5.3 Installing New Templates/Themes

- **Templates are often referred to as themes in Drupal.**
- The latest version of Drupal places all core themes under a directory named /core/themes and all contrib or custom themes under a directory named /themes (in the webroot).
- Below are the steps for carrying out installation of themes:

1. **Download the theme:**
   - Numerous themes are available in external sites, but it's important to ensure the version of the theme matches your version of Drupal.
   - The downloaded theme will appear in a compressed file format such as 'tar.gz' or 'zip'.
   - You need to extract the compressed file then you will get a list of files extracted into a folder.
2. **Upload the folder:**
   - Copy the files of downloaded theme to the desired themes folder in your Drupal installation.
   - Store all core themes under a directory named/ core/themes and all contrib or custom themes under a directory named /themes (in the webroot).
3. **Enable and make it the active, default theme:**
   - In the main Administration menu of your site, go to 'Appearance' (/admin/appearance).
   - Click on 'Install' to install the theme and use "set as default' to enable the theme for your website.
4. **Finally, Click the 'Save Configuration' button and the installation is complete.**

6.3.5.4 Creating Templates/Themes

- A template is referred to as theme in Drupal.
- A sites appearance, layout, styling and presentation can be controlled by Themes in drupal.

- Themes give a new look and feel to any website.
- A theme is a collection of files that define the layout and style information for Drupal content.
- To configure themes, the steps are as follows:

1. **Create the Theme folder:**
   - Creating a theme folder is the first and foremost step while creating themes. In this folder .info.yml file and libraries file will reside.
   - Create a folder in the /sites/all/themes directory.
   - Name your folder themename, all lowercase.
2. **Create the .info.yml file:**
   - Create a file named `themename.info.yml`, inside '`themename`' folder.
   - The `themename.info.yml` file can also be referred to as the configuration file in which the description of the site information can be stored(as below) using text editor like Notepad.

```
name: themename
type: theme
base theme: false
Core: 8.x
libraries:
  - themename/global-styling-and-scripts
regions:
header: Header
primary_menu: 'Primary menu'
secondary_menu: 'Secondary menu'
page_top: 'Page top'
page_bottom: 'Page bottom'
highlighted: Highlighted
featured_top: 'Featured top'
breadcrumb: Breadcrumb
content: 'Content'
```

3. **Create the `.libraries.yml` file:**
   - Create a file named `themename.libraries.yml`, inside '`themename`' folder.
   - All the styling and js code are stored here.
   - Define all the css and js to be used.

```
global-styling-and-scripts:
version: VERSION
css:
theme:
css/bootstrap.css: {}
css/style.css: {}
dependencies:
- core/jquery
- core/jquery.ui.effects.core
js:
js/bootstrap.min.js: {}
js/Customjs.js: {}
```

*dependencies: Add dependencies only if required.

4. Creating Stylesheets:

- Add the CSS and Js files defined in `themename.libraries.yml`. Here, you can see an example style.css. You can design it in your own unique way.

```
/*header layout styling*/
.path-frontpage header
{
  background:
  #2A93D3;
  padding: 20px 0;
}
.region.region-header
{
  max-width: 1100px;
  margin: auto;
  width: 100%;
}
/* search form*/
.search-block-form h2
{
  color:
  #FFF;
  font-size: 16px;
  margin: 0;
  font-family: arial;
  font-weight: normal;
  display: inline-block;
}
#search-block-form
{
  display: inline-block;
}
#search-block-form #edit-submit
{
  background:
  #DDD;
  border: none;
  padding: 5px 15px;
  font-size: 14px;
}
#block-dummy-search
{
  text-align: right;
}
/* site-name */
#block-dummy-branding
```

```
{
  margin: 20px 0;
}
.site-name
{
  display: inline-block;
}
.site-name a
{
  color:
  #FFF;
  font-size: 18px;
  text-decoration: navajowhite;
}
.site-logo
{
  font-size: 14px;
  color: #FFF;
  margin-right: 10px;
}
/*main-menu*/
#block-dummy-main-menu
{
  max-width: 1100px;
  margin: auto;
  width: 100%;
}
ul.menu a.is-active
{
  color: #000;
  background: #FFF;
  padding: 10px;
  font-size: 14px;
  text-decoration: navajowhite;
  display: inline-block;
}
.menu li a
{
  color: #FFF;
  font-size: 14px;
  text-decoration: none;
  padding: 10px;
}
.navigation li
{
```

```
  display: inline-block;
}
.navigation .menu
{
  margin: 0;
}
/*main content */
.path-frontpage main
{
  max-width: 1100px;
  margin: auto;
  width: 100%;
}
```

- The above code simply sets the content background color, width, and margin of the navigation bar and so on.
- It also specifies information about styling the text of sidebars, footer, and navigation bar.
- You can give a unique touch to different elements of content and format the layout of your page through a stylesheet.
- You can also add more stylesheets as per your needs.

5. Design:

- If you want, you can view the core PHP template files you're using in your theming.
- The main files you might want to view include:
  - page.tpl.php (for the overall page layout)
  - node.tpl.php (for all the 'nodes' or main content sections of a page)
  - block.tpl.php (for the blocks, in whichever regions you place them)
  - comment.tpl.php (for all comments on your site)
- These PHP template files can be copied to your theme folder, and then modified as per your requirement, if you desire to change the way the html is structured.

6.3.5.5 Adding Modules and Component

Adding Modules :

- Drupal provides module which is a collection of PHP files having some functionality.
- Drupal also provides built-in modules.
- By installing a module, features and functionalities can be turned on and vice-versa.
- Modules in Drupal can be customized according to the user's requirements.
- The core download of Drupal provides modules for following functionalities :
  - Managing user accounts.
  - Managing basic content and fields managing navigation menus (the core Menu UI module).
  - Making lists, grids, and blocks from existing content.

- A module can be installed through the administrative interface.

1. Check the '*Manage*' administrative menu and navigate to '*Extend*' (*admin/modules*). All the available modules in the site are displayed on the *Extend* page.
2. Whichever modules have to be installed, accordingly check the boxes.

Adding Components:

- A component is a block plugin called `ComponentBlock` which is an extension of the block entity in Drupal.
- Interaction with components is just like interaction with normal blocks.
- The `component.yml` file informs Drupal about the component information like what this component is called, location of the assets, configuration of the blocks etc.
- Library definition for each component is created by the module which loads any other library dependencies, and renders the default html to the page.
- Drupal make the component available in the library so that it can be shared by other components too.

6.3.5.6 Modifying the Existing Templates/Theme

- It's quite simple and easy to duplicate, rename and customize an existing Drupal theme and use it to build your customized theme than to build one from scratch.
- The first step is to install the theme you want to modify.
- Later create a copy of the theme directory on your computer.
- Use this local copy to make your changes, and then upload the modified copy to your 'site/all/themes' directory on your web host.
- To edit the files in a theme, use a text editor:

To create your own version of customized theme, follow these steps:

1. Rename the folder to a unique name of your choice. The name may only contain lowercase letters, numbers, and underscores. For example, 'drupalfordummies'.

2. Open this folder and locate the .info file. Rename this file with your new theme name. For this example, 'drupalfordummies.info' is used.
Make sure you rename the info file. There should be only one 'info' file in your theme directory, and the name should match the name of your theme.

3. Open the .info file in a text editor.

4. Change the first line of this file to simply `;$Id$`

5. Change the name = line to use your theme name.
In the example, this line is changed to name = Drupal For Dummies.
This will be the name that shows up on the module selection page. It can contain spaces.

6. Change the description text to explain and describe your theme.
The example changes the description to 'description = Yellow and Black Theme'.

7. Now save this file using the new name which you have given to your theme.
   The example is named drupalfordummies.info. You can delete the original .info file.
   Your .info file should now look similar to the below figure. (Fig 6.34)
8. Open the template.php file in a text editor, use FIND and REPLACE to find the name "garland" and replace it with your theme's name (for example, *drupalfordummies*). Save your file.
9. Repeat Step 8 for the theme-settings.php file.

6.3.5.7 Creating Templates/Themes with Web Editors

- Most of the times, we often run into the problem of the content editor being locked in to predetermined layouts.
- This is where the WYSIWYG Template module becomes a super-handy addition to the WYSIWYG editor.
- Using this editor makes the task of the user easy by offering simple and customized options for creating templates.
- The WYSIWYG Template module allows the user to create and build any code template which can be inserted into the WYSIWYG editor by any content manager.
- This is of great help when a layout-specific code has to be inserted without any knowledge of HTML.
- Firstly the WYSIWYG editor module has to be installed in Drupal Website.
- Once the module is installed and enabled, ensure to enable the WYSIWYG Template button inside the WYSIWYG editor.
- Once enabled, switch over to the template configuration page located at "`/admin/config/content/wysiwyg-templates.`"
- From here users can build their own templates. Anything that is added in a template can be inserted inside of any body text.

6.3.5.8 Creating Real Templates / Themes

- As mentioned before (Creating real templates in Joomla), real templates refer to augmenting features according to users requirement and customize a full-fledged template.
- In the previous section (Creating templates in Drupal) we learnt the basics of creating a template from scratch. Most templates are built from an existing base or prototype template.
- A user can use themes contributed by others or create own to share with the community.
- Contributed themes are not part of any official release and may not have optimized code/functionality for user's purposes.
- In Drupal, you can create sub themes which is just like any other theme with just a single difference i.e. they inherit the resources of parent theme.

- A sub-theme can be a child of another sub-theme, and it can be branched and organized however you need it.
- This is what gives sub-themes great potential and make them significant.
- For creating sub-themes, define it like any other theme and declare its base theme with the "base theme" key. (Not to forget that that this key has no underscore.)
- Following is the folder structure after implementing the following files:

```
themes/
└──  fluffiness/
     ├── fluffiness.info.yml
     └── fluffiness.libraries.yml
```

```
The info file is named fluffiness.info.yml:
name: Fluffiness
type: theme
description: This is a fluffy sub theme of Classy
core: 8.x
# Defines the base theme
base theme: classy
# Defines libraries group in which we can add css/js.
libraries:
  - fluffiness/global-styling
# Regions
regions:
  header: Header
  featured: Featured
  content: Content
  sidebar_first: First sidebar
  sidebar_second: Second sidebar
  footer: Footer
```

Include `fluffiness.libraries.yml` file to add css/js in a global-styling group, defined above in the libraries: key.

```
global-styling:
  css:
    component:
      css/style.css: {}
```

If the user wants to use a different name instead of "fluffiness", replace all occurrences of "fluffiness" with your own name ("all occurrences" includes the folder name too).

For example:

```
themes/
└── my_custom_theme/
    ├── my_custom_theme.info.yml
    └── my_custom_theme.libraries.yml
```

The text you put in the "name:" line of the `info.yml` file is free-form and doesn't need to exactly match your sub-theme's filename. For example, it could be like this:

```
name: My Custom Theme
# (all the other lines omitted for brevity)
```

Summary

- PHP is one of the popular scripting languages which is platform independent and provides integration with several databases.
- MVC framework helps developers build robust applications suited to all types of users and areas.
- The MVC architecture is of great advantage to developers as it reduces their burden by providing presentable and standard coding pattern.
- Drupal is both a framework and content management system which is free and open source and packed with various built in functionalities.
- Joomla is also a free and open source content management system widely used for publishing web content.
- Drupal and Joomla provide various options to create and customize articles, pages, contents, themes etc.
- Plugins are add-on software, which are used to enhance the capabilities of a program.
- Components are the main functional units and can also be termed as Joomla/Drupal extensions.
- Articles are stories which can also be treated as personal blog or used as news article.
- Menus provide navigation to various web pages on a Drupal site.
- Menus also provide a reference point to the content types like articles, basic pages, contents etc.
- Themes can also be referred to as templates which are used for controlling a site's appearance and presentation.
- A module is a collection of files comprising of some functionalities written in PHP.

Check Your Understanding

1. _____ is an application design pattern that separates the application data and business logic from the presentation.
   (a) CMS (b) MVC
   (c) Framework (d) PHP
2. In Joomla _____ provides the flexibility to the administrator to select the default template as well as provide editing options for both the main index file of the template and the style sheet file.
   (a) Template Manager (b) Article Manager
   (c) Module Manager (d) Language Manager

3. The _____ helps the user to manage and administer a multilingual site.
   (a) Category Manager (b) Language Manager
   (c) Section Manager (d) Frontpage Manager
4. The Control Panel is a centralized panel where the administrator can jump to the most common parts of the site.
   (a) Menus (b) Articles
   (c) Control Panel (d) Modules
5. A template controls the _____ of a site.
   (a) settings (b) layout
   (c) various managers (d) None
6. CSS is _____.
   (a) Custom Style Sheet (b) Cascading Simple Sheet
   (c) Cascading Style Sheet (d) Category Style Sheet
7. Components can also be termed as _____.
   (a) simple applications (b) mini applications
   (c) complex applications (d) None
8. Articles can also be termed as _____.
   (a) documentation (b) book
   (c) short stories (d) None
9. For effective and easy navigation, _____ provide links to various web pages.
   (a) Menus (b) Articles
   (c) Modules (d) Templates
10. A sites appearance and presentation can be controlled by _____.
   (a) Themes (b) Components
   (c) Menus (d) Modules

ANSWERS

1. (b)	2. (a)	3. (b)	4. (a)	5. (c)	6. (d)	7. (a)	8. (b)
9. (a)	10. (d)						

Practice Questions

Q 1. Answer the following questions in short

1. What is PHP framework?
2. What is Drupal?
3. What is Joomla?
4. What is the responsibility of Article Manager?

5. List PHP framework names.
6. What does the Language Manager in Joomla do?
7. What is the functionality of Trash Manager in Joomla?
8. How do you test a template in Joomla?
9. How do you add components in Drupal?
10. What does the WYSIWYG Template module do?
11. What are the functionalities provided by Drupal Modules?
12. What is a Content Management System?

Q II. Answer the following questions.

1. Describe MVC architecture with diagram.
2. List and explain features of Drupal.
3. Explain features of Joomla
4. Explain menus in drupal.
5. What are the features of Administrative toolbar in Drupal?
6. Explain Presentation Administration in Joomla and Drupal.
7. How are articles created in Drupal?
8. How are templates created in Joomla?
9. Explain Content Administration in Joomla and Drupal.
10. How are modules and components added in Drupal?
11. Explain System Administration in Joomla and Drupal.

Q III. Define the terms

1. Framework
2. MVC
3. Content Management System
4. Module
5. Component
6. Plugin
7. Template
8. Article
9. Dashboard
10. Model
11. View
12. Controller

***

Bibliography

- Beginning Joomla! From novice to Professional by Dan Rahmel, Apress, 2008.
- https://www.drupal.org/docs
- https://docs.joomla.org/Main_Page
- https://www.dummies.com/web-design-development/drupal/customizing-drupal-themes/

***