Курс:

«Создание web–приложений, исполняемых на стороне сервера при помощи языка программирования PHP, СУБД MySQL и технологии Ajax»

Модуль 01

ТЕМА: ОСНОВЫ РАЗРАБОТКИ НА СТОРОНЕ СЕРВЕРА

Что такое веб-сервер?

Процессы обработки файлов, выполнения запросов в базах данных в Интернете происходит при помощи веб-сервера. Понятие «веб-сервер» может относиться к аппаратной начинке и к программному обеспечению:

1. С точки зрения аппаратной части, веб-сервер — это компьютер, который хранит файлы сайта (стили, скрипты, картинки и другие) и доставляет их на клиентское устройство.
2. С точки зрения программной части, **веб-сервер** — это множество программных компонентов, которые контролируют доступ веб-пользователей к размещённым на сервере файлам. Например, **HTTP-сервер** — часть ПО, которая понимает URL-адреса и протокол обращения к страницам.

Для функционирования веб-сайта необходимо использовать либо статический, либо динамический веб-сервер:

Статический веб-сервер состоит из аппаратной части с серверным ПО — HTTP-сервером. Такой сервер отправляет размещённые файлы «как есть».

Динамический веб-сервер состоит из статического веб-сервера и дополнительного программного обеспечения, чаще всего сервера приложения и базы данных. Сервер приложений изменяет исходные файлы перед отправкой клиенту по протоколу HTTP. Обобщённо говоря, динамический веб-сервер может «выполнять код».

Как работает Apache

Часть сайтов в Интернете обеспечивается работой благодаря веб-серверу Apache. При запросе клиента на определённый адрес в Интернете отправляется запрос информации на сервер, а Apache возвращает ответ с необходимыми данными (страницей, текстом, мультимедиа и так далее).

Размещение файлов на сервере

Корневая папка сервера (`public_html`) — это ключевая директория, которая содержит любое содержимое ресурса, которое может быть доступным для клиента. Необходима из-за соображений безопасности и не должна включать важные скрипты, конфигурационные файлы и так далее.

Индексный файл — это файл главной страницы директории на сервере, то есть тот файл, который загружается, когда посетитель обращается напрямую к какому-либо каталогу.

Конфигурация веб-сервера

Изменить конфигурацию веб-сервера Apache можно разными способами, самыми распространёнными из которых является модификация **главного файла конфигурации** сервера Apache (`httpd.conf`) или же **файла уровня каталога** (`.htaccess`). Оба

файла можно изменять в любом текстовом редакторе, в том числе, консольном.

Они содержат набор инструкций, обуславливающих принципы работы Apache. Однако `.htaccess` покрывает рамки текущего каталога и дочерних папках без внесения изменений в общие настройки веб-сервера.

Директивы конфигурации — это наборы правил, которые определяют, как должно работать программное обеспечение (сервер) и другие настройки.

Файл конфигурации `httpd.conf`

Файл, который содержит конфигурацию сервера `httpd.conf`, содержит основное техническое описание работы «**демона**» — программного обеспечение, которое работает в фоновом режиме. Многим директивам присвоены некоторые значения по умолчанию, благодаря чему они требуют минимального изменения со стороны администратора. Управлять директивами основного конфигурационного файла могут только администраторы.

Важно: изменения в файле `httpd.conf` вступают в силу только после перезагрузки сервера.

Файл конфигурации `.htaccess`

Файл конфигурации каталога `.htaccess` отвечает за переопределение директив главного конфигурационного файла для отдельной директории (для локальных нужд).

Конфигурационный файл может находиться как в корневой директории хоста (`/public_html`), так и приложения (`/public_html/appName`). Настройки, сделанные в файле `.htaccess`, распространяются на родительский и дочерние

каталоги. Настройка файла позволяет вносить некоторые изменения в работу веб-сервера без получения прав администратора.

Локальный и виртуальный хост

Локальный хост — зарезервированное доменное имя для частных IP-адресов (в диапазоне от `127.0.0.1` до `127.255.255.255`). Для сети, состоящей из одного компьютера, используется `127.0.0.1`, именуемое также `localhost`.

Виртуальный хост — выделенное место для сайта или приложения на физическом или программном сервере.

Настройка окружения

Для работы с PHP потребуется установить сборку веб-сервера XAMPP, которая включает в себя поддержку выполнения кода PHP, а также работу с базой данных MySQL.

Загрузить установочный пакет можно с официального сайта: https://www.apachefriends.org. Для установки сборки могут потребоваться права администратора.

Текстовые редакторы

В разработке скриптов PHP можно использовать различные текстовые редакторы. В некоторых редакторах предусмотрена расширенная поддержка работы с PHP, включая выделение ключевых терминов и интеграцию с инструментарием, среди основных редакторов с поддержкой PHP можно выделить:

- Visual Studio Code (бесплатный)
- Brackets (бесплатный)
- Atom (бесплатный)
- WebStorm (платный)
- Sublime Text (платный)

Введение в PHP

PHP (Hypertext Preprocessor) — это распространённый язык программирования общего назначения с открытым исходным кодом. PHP специально сконструирован для веб-разработок и его код может внедряться непосредственно в HTML.

Основная задача PHP — это «оживление» HTML страниц. PHP позволяет изменять веб-страницу на сервере непосредственно перед тем, как она будет отправлена браузеру. PHP умеет исполнять код — так называемые сценарии. В ходе исполнения PHP может изменить или динамически создать любой HTML-код, который и является результатом исполнения сценария. Затем сервер отправляет этот код браузеру. При этом браузеру не известно, как была сформирована данная страница — статично свёрстана верстальщиком, или динамически создана при участии PHP. Это не важно, так как браузер всегда работает только с тем, что получил от сервера.

PHP, как язык программирования, является:

1. **Типизированным** с **динамической нестрогой типизацией** — языком программирования, что имеет совокупность правил, назначающих свойства, именуемые типами, различных конструкций, составляющим программу — переменные, выражения, функции, модули. В типизированном языке программирования с ***динамической*** типизации все типы выясняются во время выполнения кода, что позволяет создавать более гибкие алгоритмы. Языки с ***нестрогой*** типизацией выполняют множество неявных преобразований автоматически, даже если может произойти потеря точности или преобразование неоднозначно.
2. **Интерпретируемым** — это язык программирования, которому для исполнения программы не нужен машинный код; вместо этого программу построчно исполняет интерпретатор. Такой язык может реализовывать конструкции, позволяющие динамические изменения на этапе времени выполнения. Эти конструкции затрудняют компиляцию и трансляцию на компилируемый язык.
3. **Объектно-ориентированным** — языком программирования, основанным на методологии программирования с представлением программы в виде совокупности объектов — сущностей, способных сохранять своё состояние (информацию) и обеспечивающая набор операций (поведение) для проверки и изменения этого состояния. Каждый из объектов является экземпляром определённого класса — шаблон для создания объектов, обеспечивающий начальные значения состояний.

Скрипты PHP могут быть описаны только в файле с расширением `.php` и обозначается `<?php` или `<?` (в случае, если предусмотрена соответствующая настройка в файле интерпретатора —

`php.ini`). Скрипт на PHP, как и на любом другом языке программирования, — это набор команд (инструкций). Обработчику программы необходимо как-то отличать одну команду от другой. Для этого используются специальные символы — **разделители**. В PHP инструкции разделяются при помощи точки с запятой.

Важным элементом каждого языка являются переменные, константы и операторы, применяемые к этим переменным и константам.

Переменная в PHP обозначается знаком доллара, за которым следует её имя. Например: `$variable`. Имя переменной чувствительно к регистру, то есть переменные `$variable` и `$Variable` различны. Благодаря динамической типизации возможно не указывать тип данных перед объявлением переменной.

Вывод информации происходит при помощи конструкции `echo`.

PHP предлагает как классический, так и альтернативный синтаксис для написания сценариев, при использовании которого фигурные скобки не ставятся. Смешивание синтаксиса в одном и том же блоке кода **не поддерживается**.

Альтернативный синтаксис возможно использовать при необходимости интеграции альтернативного кода между условиями PHP, например, языка HTML. Так как в этом случае используемые языки будут разделены, что упрощает разработку и восприятие страницы.

Типы данных

PHP поддерживает восемь простых типов данных:

1. `Boolean` — логический тип, переменные которого могут принимать два значения: `true` и `false`. Чаще всего логические значения используются в условных конструкциях.

2. `Integer` — представляет целое число со знаком размером в 32 бита.

3. `Double` — числа с плавающей запятой. Их размер, как правило, зависит от платформы.

4. `String` — строки, которые можно применять для работы с текстом. Строки бывают двух типов: в двойных кавычках и одинарных. От типа кавычек зависит обработка строк интерпретатором. Так, **переменные в двойных кавычках заменяются значениями**, а переменные в одинарных кавычках остаются неизменными. В качестве примера стоит объявить переменную `$world` со значением `World` и сделать вывод текста разными способами:

```
echo "Hello, $world!";
```

Выходное значение: `Hello, World!`

```
echo 'Hello, $world!';
```

Выходное значение: `Hello, $world!`

5. `Array` — массивы. Ассоциативный массив определяет набор элементов, каждый из которых представляет пару `ключ=>значение`. Объявление массива выглядит следующим образом:

```
$example = array(
    "key" => "value",
    "ключ" => "значение"
);
```

6. `Object` — объекты. Объект представляет собой переменную, экземпляр которой создаётся по специальному шаблону, называемому классом. Концепции объектов и классов являются неотъемлемой частью парадигмы объектно-ориентированного программирования. Объявление свойств для объекта (или добавление новых) имеет следующий синтаксис:

```
$переменная->ключ = значение.
```

Стоит упомянуть, что объект возможно создать без определения класса при помощи общего пустого класса `stdClass`. В таком случае конструкция создания объекта будет выглядеть так:

```
$object = new stdClass();
$object->key = "value";
```

7. `Resource` — ресурсы, специальный тип переменных, содержащих ссылки на внешние ресурсы, которыми могут быть файлы или подключения к базам данных.

8. `NULL` — специальное значение переменной, указывающее, что её значение не определено. Используется, когда необходимо указать, что переменная не имеет значения (пустая).

После отправки данных на сервер они, как правило, представлены в виде массива.

Подключение файлов

В PHP можно вызвать один сценарий из отдельного файла по его имени или по имени вызываемой функции. Такая способность называется **подключением файлов**. Причём таковым файлом может являться как PHP-сценарий, так и любой другой текстовый файл. Например, HTML-страница.

Для подключения файлов в PHP есть несколько конструкций:

1. **Конструкция включения файла до выполнения сценария** (`require`) — во время выполнения кода интерпретатор заменяет инструкцию на содержимое файла. Это можно использовать для подключения библиотек, HTML-страниц и так далее, что позволяет собрать единый сценарий из нескольких файлов разного типа. Данную конструкцию невозможно использовать в циклах для динамического подключения разных файлов. В случае отсутствия файла возвращается фатальная ошибка.
2. **Конструкция включения файла во время выполнения сценария** (`include`) — используется для включения файлов по время выполнения кода, что позволяет подключать разные файлы в циклах.

Конструкции однократного включения

В больших PHP сценариях инструкции `include` и `require` применяются очень часто. Поэтому становится довольно сложно контролировать, как бы случайно не включить один и тот же файл несколько раз, что чаще всего приводит к ошибке, которую сложно обнаружить.

В PHP предусмотрено решение данной проблемы. Используя конструкции однократного включения `require_once` и `include_once`, можно быть уверенным, что один файл не будет включён дважды. Работают конструкции однократного включения так же, как и `require` и `include` соответственно. Разница в их работе лишь в том, что перед включением файла интерпретатор проверяет, был ли включён ли указанный файл ранее или нет. Если да, то файл не будет включён вновь.

Операторы сравнения

Операторы сравнения позволяют сравнивать между собой два значения. Это уникальные операции, потому что независимо от типов своих аргументов они всегда возвращают одно из двух значений: `true` или `false`.

В PHP разрешается сравнивать только *скалярные* переменные. Массивы и объекты сравнивать нельзя, даже на равенство: при выполнении такой операции массивы преобразуются в слово `Array`, которое и будет сравниваться.

Имеющиеся операторы сравнения представлены в таблице 1.

Таблица 1 — Операторы сравнения

Пример	Название	Результат
`$a == $b`	Equal Равно	**`true`**, если `$a` равно `$b` после преобразования типов
`$a === $b`	Identical В точности равно	**`true`**, если `$a` равно `$b` и имеет тот же тип
`$a != $b` `$a <> $b`	Not equal Не равно	**`true`**, если `$a` не равно `$b` после преобразования типов
`$a !== $b`	Not identical В точности не равно	**`true`**, если `$a` не равно `$b` или они разных типов
`$a < $b`	Less than Меньше	**`true`**, если `$a` строго меньше `$b`
`$a > $b`	Greater than Больше	**`true`**, если `$a` строго больше `$b`
`$a <= $b`	Less than or equal to Меньше или равно	**`true`**, если `$a` меньше или равно `$b`

Пример	Название	Результат
`$a >= $b`	Greater than or equal to Больше или равно	**`true`**, если $a больше или равно $b
`$a <=> $b`	Spaceship Космический корабль	Число типа Integer меньше, больше или равное нулю, когда $a соответственно меньше, больше или равно $b

В случае сравнения числа со строкой или две строки, содержащие числа, каждая строка будет преобразована в число, и сравниваться они будут как числа. Преобразование типов не происходит при использовании операторов `===` и `!==`, так как в этом случае кроме самих значений сравниваются типы данных.

Ветвление

Ветвление в программировании — это операция, применяющаяся в случаях, когда выполнение кода должно зависеть от некоторого условия. Ветвление является одной из трёх (наряду с последовательным исполнением команд и циклами) базовых конструкций структурного программирования.

В языке PHP существует три основных формы реализации ветвления — *условный оператор*, *тернарный оператор* и *оператор множественного выбора*.

Условный оператор

Условный оператор начинается с ключевого слова `if` и реализует выполнение команд, когда логическое выражение (условие) принимает значение `true`. Встречаются следующие формы условного оператора:

1. **Единичный выбор** — если при выполнении условие возвращает `true`, то команды выполняются, иначе выполнение кода продолжается со следующей за условным оператором команды.
2. **Множественный выбор** — при возвращении `false` первого блока, может быть выполнено альтернативное условие — `elseif`.
3. **Двойной выбор** — при возвращении `true`, код выполняются из первого блока в фигурных скобках, при `false` — из второго. Используется в качестве последнего возможного альтернативного значения, который находится в самом конце. Обозначается как `else`.
4. **Конец оператора** — используется только в альтернативном синтаксисе в качестве завершающего элемента.

Формы записи условного оператора для классической и альтернативной форм записи представлены в таблице 2.

Таблица 2 — Формы записи условного оператора

Наименование	Классический синтаксис	Альтернативный синтаксис
1. **Единичный выбор**	`if (cond) {}`	`<? if (cond): ?>`
2. **Множественный выбор**	`elseif (cond) {}`	`<? elseif (cond): ?>`
3. **Двойной выбор**	`else (cond) {}`	`<? else: ?>`
4. **Конец оператора**	–	`<? endif; ?>`

Тернарный оператор

Также существует и тернарный оператор, который работает почти, как и условный оператор, но вместо ключевых слов требуется писать ? в случае значения `true` и `:` для `false`. Тернарный оператор имеет следующий синтаксис:

```
cond ? true : false;
```

Конструкции выбора

Часто вместо нескольких расположенных подряд инструкций условного оператора целесообразно воспользоваться конструкцией выбора. Она предназначена для выбора действий в зависимости от значения является аналогом условного оператора.

Конструкцию выбора можно использовать, если предполагаемых вариантов много и для каждого варианта нужно выполнить определённые действия. В таком случае использование условного

оператора с множественным выбором становится неудобным и менее производительным.

Принцип работы конструкции выбора:

1. Вычисляется значение выражения.
2. Поиск и выполнение кода из подходящего блока.
3. Если значение из набора блоков не совпадает со значением выражения, выполняется блок `default` (если он указан).

Формы записи конструкции выбора для классической и альтернативной форм записи представлены в таблице 3.

Таблица 3 — Формы записи конструкции выбора

Наименование	Классический синтаксис	Альтернативный синтаксис
1. **Объявление конструкции**	`switch (var) {}`	`<? switch (var): ?>`

2. **Первое выражение**	`case value:`
	`  // Case code`
	`  break;`
		`case value: ?>`
		`  // Alt case code`
		`<? break; ?>`

3. **Последующие выражения**		`<? case value: ?>`
		`  // Alt case code`
		`<? break; ?>`

4. **Дополнительное выражение**	`default:`
	`  // Case code`
	`  break;`
		`<? default: ?>`
		`  // Alt case code`
		`<? break; ?>`

5. **Конец конструкции**	–	`<? endswitch: ?>`

В случае использования альтернативного синтаксиса важно обратить внимание на то, что **нельзя разрывать условие и первое выражение**.

Формы

В качестве простейшего примера пользовательского интерфейса и взаимодействия клиента и сервера можно выделить формы для заполнения данных.

Веб-формы позволяют ввести данные, которые отправляются на сервер для обработки и хранения или используются на клиента для обновления интерфейса (например, добавление элемента в список).

HTML-формы — состоят из нескольких элементов управления, которыми могут быть однострочные или многострочные текстовые поля, выпадающие списки, кнопки, и так далее.

Внутри форм используются следующие HTML-элементы:

1. `<form>` — контейнер формы;
2. `<label>` — подпись элемента;
3. `<input>` — поле для ввода;
4. `<textarea>` — многострочное поле для ввода (например, комментария);
5. `<button>` — кнопка.

Пример формы, состоящей из восьми элементов представлен на рисунке 1, нумерация элементов соответствует номерам из списка с перечнем.

Рисунок 1 — Простейшая форма для отправки данных

При нажатии на кнопку «Отправить сообщения» все введённые данные собираются в массив и отправляются на сервер.

Обработка данных формы

Одним из главных достоинств PHP является работа с HTML-формами. После отправка данных формы PHP-скрипту, информация из неё становится доступной.

Форма отправляется на сервер в виде запроса, который может передаваться в виде двух методов запросов — `GET` и `POST`. В соответствии с каждым методом, информация из запроса преобразуется в *суперглобальный массив*.

Суперглобальный массив или **суперглобальная переменная** — это встроенные переменные, которые всегда доступны во всех областях видимости. То есть, в любом месте скрипта и без использования синтаксиса `global $variable;` для доступа в других функциях и методах, среди них:

- `$_GET` — массив переменных, переданных скрипту через URL (строку запроса);
- `$_POST` — массив переменных, переданных в заголовке запроса HTTP.
- `$_COOKIE` — ассоциативный массив значений, переданных через файлы Cookies.
- `$_REQUEST` — массив, который содержит данные переменных `$_GET`, `$_POST` и `$_COOKIE`.

Массивы `$_GET` и `$_POST` будут существовать только при наличии запроса к серверу. Для проверки, существует ли данная переменная в программном коде, используется функция `isset()` и принимает искомую переменную в качестве аргумента.

GET-запрос при обработке формы подразумевает, что информация будет передана в адресной строке, как на примере ниже:

```
https://example.kz/index.php?key=value
```

Так внутри суперглобального массива будут объявлен ключа `key` со значением `value`. Этот метод используется для передачи серверу поисковых запросов, номера страницы и так далее.

POST-запрос же отличается от GET. Он используется для передачи данных с информацией в теле запроса, которые не видны пользователю. Этот метод обычно используется для передачи большего количества информации.

Существует заблуждение, что GET-запросы к серверу «менее безопасны», чем POST, так как передаются в открытом виде и могут быть перехвачены, но **это не так**. Без использования HTTPS *любые* данные будут переданы в открытом виде.

При использовании HTTPS полный URL-адрес виден только клиентскому устройству, а провайдер видит и другие устройства в сети видят только домен, на который отправляется запрос.