Создание Web приложения с использованием ООП и языка PHP

**Введение**

**Веб-приложение** — [клиент-серверное](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_%E2%80%94_%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80" \o "Клиент — сервер) приложение, в котором [клиент](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)" \o "Клиент (информатика)) взаимодействует с [веб-сервером](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80" \o "Веб-сервер) при помощи [браузера](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80" \o "Браузер). [Логика](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0" \o "Логика) веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по [сети](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C" \o "Компьютерная сеть). Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной [операционной системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0" \o "Операционная система) пользователя, поэтому веб-приложения являются межплатформенными службами.

Веб приложения стали широко использоваться в конце [1990-х](https://ru.wikipedia.org/wiki/1990-%D0%B5_%D0%B3%D0%BE%D0%B4%D1%8B" \o "1990-е годы) — начале [2000-х годов](https://ru.wikipedia.org/wiki/2000-%D0%B5_%D0%B3%D0%BE%D0%B4%D1%8B" \o "2000-е годы).

**Чем веб-приложения отличаются от сайтов**

Для меня сайт это в первую очередь что-то информационное и статичное: визитка компании, сайт рецептов, городской портал или вики. Набор подготовленных заранее HTML-файлов, которые лежат на удаленном сервере и отдаются браузеру по запросу.  
  
Сайты содержат различную статику, которая как и HTML-файл не генерируется на лету. Чаще всего это картинки, CSS-файлы, JS-скрипты, но могут быть и любые другие файлы: mp3, mov, csv, pdf.  
  
Блоги, визитки с формой для контакта, лендинги с кучей эффектов я тоже отношу для простоты к сайтам. Хотя в отличие от совсем статических сайтов, они уже включают в себя какую-то бизнес-логику.  
  
А веб-приложение — это что-то технически более сложное. Тут HTML-страницы генерируются на лету в зависимости от запроса пользователя. Почтовые клиенты, соцсети, поисковики, интернет-магазины, онлайн-программы для бизнеса, это все веб-приложения.

**Какие бывают веб-приложения**

Веб-приложения можно разделить на несколько типов, в зависимости от разных сочетаний его основных составляющих:

1. Backend (бэкенд или серверная часть приложения) работает на удаленном компьютере, который может находиться где угодно. Она может быть написана на разных языках программирования: PHP, Python, Ruby, C# и других. Если создавать приложение используя только серверную часть, то в результате любых переходов между разделами, отправок форм, обновления данных, сервером будет генерироваться новый HTML-файл и страница в браузере будет перезагружаться.
2. Frontend (фронтенд или клиентская часть приложения) выполняется в браузере пользователя. Эта часть написана на языке программирования Javascript. Приложение может состоять только из клиентской части, если не требуется хранить данные пользователя дольше одной сессии. Это могут быть, например, фоторедакторы или простые игрушки.
3. Single page application (SPA или одностраничное приложение). Более интересный вариант, когда используются и бэкенд и фронтенд. С помощью их взаимодействия можно создать приложение, которое будет работать совсем без перезагрузок страницы в браузере. Или в упрощенном варианте, когда переходы между разделами вызывают перезагрузки, но любые действия в разделе обходятся без них.

**Технические особенности**

Существенное преимущество построения веб-приложений для поддержки стандартных функций браузера заключается в том, что функции должны выполняться независимо от операционной системы данного клиента. Вместо того, чтобы писать различные версии для Microsoft Windows, Mac OS X, GNU/Linux и других операционных систем, приложение создаётся один раз для произвольно выбранной платформы и на ней разворачивается. Однако различная реализация , CSS, DOM и других спецификаций в браузерах может вызвать проблемы при разработке веб-приложений и последующей поддержке. Кроме того, возможность пользователя настраивать многие параметры браузера (например, размер шрифта, цвета, отключение поддержки сценариев) может препятствовать корректной работе приложения.

Другой (менее универсальный) подход заключается в использовании Adobe Flash, Silverlight или Java-апплетов для полной или частичной реализации пользовательского интерфейса. Поскольку большинство браузеров поддерживает эти технологии (как правило, с помощью плагинов), Flash- или Java-приложения могут выполняться с легкостью. Так как они предоставляют программисту больший контроль над интерфейсом, они способны обходить многие несовместимости в конфигурациях браузеров, хотя несовместимость между Java- или Flash-реализациями на стороне клиента может приводить к различным осложнениям.

На 2015 год технологию Adobe Flash не поддерживают Chrome, Safari, и другие популярные браузеры.

**Архитектура веб-приложений**

Веб-приложение состоит из клиентской и серверной частей, тем самым реализуя [технологию «клиент-сервер»](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80" \o "Клиент-сервер).

**«Клиент — сервер»** — вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически [клиент](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)" \o "Клиент (информатика)) и [сервер](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)" \o "Сервер (программное обеспечение)) — это [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5" \o "Программное обеспечение). Обычно эти программы расположены на разных [вычислительных машинах](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0" \o "Вычислительная машина) и взаимодействуют между собой через [вычислительную сеть](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C" \o "Вычислительная сеть) посредством [сетевых протоколов](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB" \o "Сетевой протокол), но они могут быть расположены также и на одной машине.

Преимущества

* Отсутствие дублирования кода программы-сервера программами-клиентами.
* Так как все вычисления выполняются на сервере, то требования к компьютерам, на которых установлен клиент, снижаются.
* Все данные хранятся на сервере, который, как правило, защищён гораздо лучше большинства клиентов. На сервере проще организовать контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа.

Недостатки

* Неработоспособность сервера может сделать неработоспособной всю вычислительную сеть. Неработоспособным сервером следует считать сервер, производительности которого не хватает на обслуживание всех клиентов, а также сервер, находящийся на ремонте, профилактике и т. п.
* Поддержка работы данной системы требует отдельного специалиста — [системного администратора](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D1%8B%D0%B9_%D0%B0%D0%B4%D0%BC%D0%B8%D0%BD%D0%B8%D1%81%D1%82%D1%80%D0%B0%D1%82%D0%BE%D1%80" \o "Системный администратор).
* Высокая стоимость оборудования.

Клиентская часть реализует [пользовательский интерфейс](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81" \o "Веб-интерфейс), формирует запросы к серверу и обрабатывает ответы от него.

**Веб-интерфе́йс** — [веб-страница](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0" \o "Веб-страница) или совокупность веб-страниц, предоставляющая [пользовательский интерфейс](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F" \o "Интерфейс пользователя) для взаимодействия с сервисом или устройством посредством протокола [HTTP](https://ru.wikipedia.org/wiki/HTTP" \o "HTTP) и [веб-браузера](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80" \o "Браузер).

На стороне клиента используется:

* Для реализации [GUI](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F)
  + [HTML](https://ru.wikipedia.org/wiki/HTML), [XHTML](https://ru.wikipedia.org/wiki/XHTML)
  + [CSS](https://ru.wikipedia.org/wiki/CSS)
* Для формирования запросов, создания интерактивного и независимого от браузера интерфейса:
  + [ActiveX](https://ru.wikipedia.org/wiki/ActiveX)
  + [Adobe Flash](https://ru.wikipedia.org/wiki/Adobe_Flash), [Adobe Flex](https://ru.wikipedia.org/wiki/Adobe_Flex)
  + [Java](https://ru.wikipedia.org/wiki/Java)
  + [JavaScript](https://ru.wikipedia.org/wiki/JavaScript)
  + [Silverlight](https://ru.wikipedia.org/wiki/Silverlight)

Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует [веб-страницу](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0" \o "Веб-страница) и отправляет её клиенту по сети с использованием [протокола](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB" \o "Сетевой протокол) [HTTP](https://ru.wikipedia.org/wiki/HTTP" \o "HTTP).

В настоящее время набирает популярность новый подход к разработке веб-приложений, называемый [Ajax](https://ru.wikipedia.org/wiki/Ajax" \o "Ajax). При использовании [Ajax](https://ru.wikipedia.org/wiki/Ajax" \o "Ajax) страницы веб-приложения не перезагружаются целиком, а лишь догружают необходимые данные с сервера, что делает их более интерактивными и производительными.

Также в последнее время набирает большую популярность технология [WebSocket](https://ru.wikipedia.org/wiki/WebSocket" \o "WebSocket), которая не требует постоянных запросов от клиента к серверу, а создает двунаправленное соединение, при котором сервер может отправлять данные клиенту без запроса от последнего. Таким образом появляется возможность динамически управлять контентом в режиме реального времени.

Для создания веб-приложений на стороне сервера используются разнообразные технологии и любые [языки программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), способные осуществлять вывод в стандартную [консоль](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D0%BE%D0%BB%D1%8C" \o "Консоль).

### ООП

**Объектно-ориентированное программирование** — парадигма программирования, в которой основными концепциями являются понятия объектов и классов.

Технология ООП обладает тремя главными преимуществами:

* она проста для понимания: ООП позволяет мыслить категориями повседневных объектов;
* повышенно надежна и проста для сопровождения — правильное проектирование обеспечивает простоту расширения и модификации объектно-ориентированных программ. Модульная структура позволяет вносить независимые изменения в разные части программы, сводя к минимуму риск ошибок программирования;
* ускоряет цикл разработки — модульность и здесь играет важную роль, поскольку различные компоненты объектно-ориентированных программ можно легко использовать в других программах, что уменьшает избыточность кода и снижает риск внесения ошибок при копировании.

Специфика ООП заметно повышает эффективность труда программистов и позволяет им создавать более мощные, масштабируемые и эффективные приложения.

Объектно-ориентированное программирование основано на:

* Инкапсуляции;
* Полиморфизме;
* Наследовании.

**Инкапсуляция**

Инкапсуляция - это механизм, объединяющий данные и обрабатывающий их код как единое целое.

Многие преимущества ООП обусловлены одним из его фундаментальных принципов — инкапсуляцией. Инкапсуляцией называется включение различных мелких элементов в более крупный объект, в результате чего программист работает непосредственно с этим объектом. Это приводит к упрощению программы, поскольку из нее исключаются второстепенные детали.

Инкапсуляцию можно сравнить с работой автомобиля с точки зрения типичного водителя. Многие водители не разбираются в подробностях внутреннего устройства машины, но при этом управляют ею именно так, как было задумано. Пусть они не знают, как устроен двигатель, тормоз или рулевое управление, — существует специальный интерфейс, который автоматизирует и упрощает эти сложные операции. Сказанное также относится к инкапсуляции и ООП — многие подробности "внутреннего устройства" скрываются от пользователя, что позволяет ему сосредоточиться на решении конкретных задач. В ООП эта возможность обеспечивается классами, объектами и различными средствами выражения иерархических связей между ними.

**Полиморфизм**

Полиморфизм позволяет использовать одни и те же имена для похожих, но технически разных задач. Главным в полиморфизме является то, что он позволяет манипулировать объектами путем создания стандартных интерфейсов для схожих действий. Полиморфизм значительно облегчает написание сложных программ.

**Наследование**

Наследование позволяет одному объекту приобретать свойства другого объекта, не путайте с копированием объектов. При копировании создается точная копия объекта, а при наследовании точная копия дополняется уникальными свойствами, которые характерны только для производного объекта.

**Классы и объекты в PHP**

**Класс** - это базовое понятие в объектно-ориентированном программировании (ООП). Классы образуют синтаксическую базу ООП. Их можно рассматривать как своего рода "контейнеры" для логически связанных данных и функций (обычно называемых методами).

Экземпляр класса - это **объект**. Объект - это совокупность данных (свойств) и функций (методов) для их обработки. Свойства и методы называются членами класса. Вообще, объектом является все то, что поддерживает инкапсуляцию.

Если класс можно рассматривать как тип данных, то объект — как переменную (по аналогии). Скрипт может одновременно работать с несколькими объектами одного класса, как с несколькими переменными.

Описание классов в PHP начинаются служебным словом **class**:

class Имя\_класса {  
// описание членов класса - свойств и методов для их обработки  
}

Для объявления объекта необходимо использовать оператор **new**:

Объект = new Имя\_класса;

**Конструкторы для классов PHP**

Конструктор - это специальный метод класса, который вызывается автоматически при инстанцинации объекта. В следующих разделах мы увидим, как инстанцировать объекты, но сейчас вам нужно просто знать, что конструктор используется для инициализации свойств объекта при создании объекта.

Вы можете определить конструктор с помощью метода \_\_construct.

**Методы для классов PHP**

Давайте подумаем о методах класса как о функциях, которые выполняют определенные действия, связанные с объектами. В большинстве случаев они используются для доступа и управления свойствами объекта и выполнения связанных операций.

## **Что такое PHP?**

PHP – это широко используемый язык сценариев общего назначения с открытым исходным кодом.

Говоря проще, PHP это язык программирования, специально разработанный для написания web-приложений (сценариев), исполняющихся на Web-сервере.

Аббревиатура PHP означает “Hypertext Preprocessor (Препроцессор Гипертекста)". Синтаксис языка берет начало из C, Java и Perl. PHP достаточно прост для изучения. Преимуществом PHP является предоставление web-разработчикам возможности быстрого создания динамически генерируемых web-страниц.

**Возможности PHP**

Возможности PHP очень большие. Главным образом, область применения PHP сфокусирована на написание скриптов, работающих на стороне сервера; таким образом.

Существуют три основных области, где используется PHP.

* Создание скриптов для выполнения на стороне сервера. PHP наиболее широко используется именно таким образом. Все, что вам понадобится, это парсер PHP (в виде программы CGI или серверного модуля), вебсервер и браузер. Чтобы Вы могли просматривать результаты выполнения PHP-скриптов в браузере, вам нужен работающий вебсервер и установленный PHP.
* Создание скриптов для выполнения в командной строке. Вы можете создать PHP-скрипт, способный запускаться вне зависимости от вебсервера и браузера. Все, что вам потребуется - парсер PHP. Такой способ использования PHP идеально подходит для скриптов, которые должны выполняться регулярно, например, с помощью cron (на платформах \*nix или Linux) или с помощью планировщика задач (Task Scheduler) на платформах Windows.
* Создание приложений GUI, выполняющихся на стороне клиента. Возможно, PHP является не самым лучшим языком для создания подобных приложений, но, если вы очень хорошо знаете PHP и хотели бы использовать некоторые его возможности в своих клиент-приложениях, вы можете использовать PHP-GTK для создания таких приложений. Подобным образом Вы можете создавать и кросс-платформенные приложения.

PHP способен не только выдавать HTML. Возможности PHP включают формирование изображений, файлов PDF и даже роликов Flash (с использованием libswf и Ming), создаваемых "на лету". PHP также способен выдавать любые текстовые данные, такие, как XHTML и другие XML-файлы. PHP способен осуществлять автоматическую генерацию таких файлов и сохранять их в файловой системе вашего сервера, вместо того, чтобы отдавать клиенту, организуя, таким образом, кеш динамического содержания, расположенный на стороне сервера.

Одним из значительных преимуществ PHP является поддержка широкого круга [баз данных](http://www.php.su/phpdb/" \t "_blank). Создание скрипта, использующего базы данных, - очень просто. В настоящее время PHP поддерживает следующие базы данных:

|  |  |  |
| --- | --- | --- |
| Adabas D | Ingres | Oracle (OCI7 и OCI8) |
| dBase | InterBase | Ovrimos |
| Empress | FrontBase | PostgreSQL |
| FilePro (только чтение) | mSQL | Solid |
| Hyperwave | Direct MS-SQL | Sybase |
| IBM DB2 | MySQL | Velocis |
| Informix | ODBC | Unix dbm |

**Преимущества PHP**

Главным фактором языка РНР является практичность. РНР должен предоставить программисту средства для быстрого и эффективного решения поставленных задач. Практический характер РНР обусловлен пятью важными характеристиками:

* традиционностью;
* простотой;
* эффективностью;
* безопасностью;
* гибкостью.

Существует еще одна «характеристика», которая делает РНР особенно привлекательным: он распространяется бесплатно! Причем, с открытыми исходными кодами ( Open Source ).

**Традиционность**

Язык РНР будет казаться знакомым программистам, работающим в разных областях. Многие конструкции языка позаимствованы из Си, Perl.

Код РНР очень похож на тот, который встречается в типичных программах на С или Pascal. Это заметно снижает начальные усилия при изучении РНР. PHP — язык, сочетающий достоинства Perl и Си и специально нацеленный на работу в Интернете, язык с универсальным (правда, за некоторыми оговорками) и ясным синтаксисом.

И хотя PHP является довольно молодым языком, он обрел такую популярность среди web-программистов, что на данный момент является чуть ли не самым популярным языком для создания web-приложений (скриптов).

**Простота**

Сценарий РНР может состоять из 10 000 строк или из одной строки — все зависит от специфики вашей задачи. Вам не придется подгружать библиотеки, указывать специальные параметры компиляции или что-нибудь в этом роде. Механизм РНР просто начинает выполнять код после первой экранирующей последовательности (<?) и продолжает выполнение до того момента, когда он встретит парную экранирующую последовательность (?>). Если код имеет правильный синтаксис, он исполняется в точности так, как указал программист.

**Эффективность**

Эффективность является исключительно важным фактором при программировании для многопользовательских сред, к числу которых относится и web .

Очень важное преимущество PHP заключается в его «движке». «Движок» PHP не является ни компилятором, ни интерпретатором. Он является транслирующим интерпретатором. Такое устройство «движка» PHP позволяет обрабатывать сценарии с достаточно высокой скоростью.

По некоторым оценкам, большинство PHP-сценариев (особенно не очень больших размеров) обрабатываются быстрее аналогичных им программ, написанных на Perl. Однако, чтобы не делали разработчики PHP, откомпилированные исполняемые файлы будут работать значительно быстрее – в десятки, а иногда и в сотни раз. Но производительность PHP вполне достаточна для создания вполне серьезных web-приложений.

**Безопасность**

РНР предоставляет в распоряжение разработчиков и администраторов гибкие и эффективные средства безопасности, которые условно делятся на две категории: средства системного уровня и средства уровня приложения.

1. *Средства безопасности системного уровня*

В РНР реализованы механизмы безопасности, находящиеся под управлением администраторов; при правильной настройке РНР это обеспечивает максимальную свободу действий и безопасность. РНР может работать в так называемом безопасном режиме (safe mode), который ограничивает возможности применения РНР пользователями по ряду важных показателей. Например, можно ограничить максимальное время выполнения и использование памяти (неконтролируемый расход памяти отрицательно влияет на быстродействие сервера). По аналогии с cgi-bin администратор также может устанавливать ограничения на каталоги, в которых пользователь может просматривать и исполнять сценарии РНР, а также использовать сценарии РНР для просмотра конфиденциальной информации на сервере (например, файла passwd).

2*. Средства безопасности уровня приложения*

В стандартный набор функций РНР входит ряд надежных механизмов шифрования. РНР также совместим с многими приложениями независимых фирм, что позволяет легко интегрировать его с защищенными технологиями электронной коммерции (e-commerce). Другое преимущество заключается в том, что исходный текст сценариев РНР нельзя просмотреть в браузере, поскольку сценарий компилируется до его отправки по запросу пользователя. Реализация РНР на стороне сервера предотвращает похищение нетривиальных сценариев пользователями, знаний которых хватает хотя бы для выполнения команды View Source.

**Гибкость**

Поскольку РНР является встраиваемым (embedded) языком, он отличается исключительной гибкостью по отношению к потребностям разработчика. Хотя РНР обычно рекомендуется использовать в сочетании с HTML, он с таким же успехом интегрируется и в JavaScript, WML, XML и другие языки.

Нет проблем и с зависимостью от браузеров, поскольку перед отправкой клиенту сценарии РНР полностью компилируются на стороне сервера. В сущности, сценарии РНР могут передаваться любым устройствам с браузерами, включая сотовые телефоны, электронные записные книжки, пейджеры и портативные компьютеры, не говоря уже о традиционных ПК. Программисты, занимающиеся вспомогательными утилитами, могут запускать РНР в режиме командной строки.

Наконец, средства РНР позволяют программисту работать с внешними компонентами, такими как Enterprise Java Beans или СОМ-объекты Win32. Благодаря этим новым возможностям РНР занимает достойное место среди современных технологий и обеспечивает масштабирование проектов до необходимых пределов.

**«Движок» PHP**

Языки программирования бывают двух видов: интерпретируемые и компилируемые. А каким языком является PHP? Для того, чтобы ответить на этот вопрос, нам необходимо разобраться в терминологии.

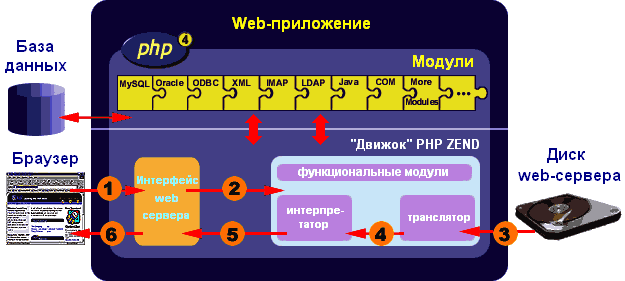
Программа, переводящая код, написанный на одном языке программирования, на другой называется транслятором. Компилятор – это тоже транслятор. Он переводит код, написанный на языке высокого уровня, в машинный код. В результате процесса компиляции создается двоичный исполняемый файл, который уже можно запускать без компилятора.

Интерпретатор – это совершенно другая категория. Интерпретатор не переводит код, а исполняет его. Интерпретатор анализирует код программы и исполняет каждую его строку. Каждый раз при исполнении такого кода, необходимо пользоваться интерпретатором.

По производительности интерпретаторы значительно уступают компиляторам, поскольку двоичный код выполняется намного быстрее. Зато интерпретаторы позволяют полностью контролировать программу во время ее исполнения.

Что касается PHP, то он не является ни компилятором, ни интерпретатором. PHP представляет собой нечто среднее, между компилятором и интерпретатором. Попробуем в этом разобраться и рассмотрим, как PHP обрабатывает код.

Рассмотрим рисунок:



Технология работы движка Zend API PHP

PHP составлен из двух почти независимых блоков — транслятора и интерпретатора.

На вход PHP подается сценарий. Он переводит его (транслирует) , проверяя синтаксис, в специальный байт-код (внутреннее представление). Затем PHP выполняет байт-код, при этом он не создает исполняемый файл.

Получается, что PHP является транслирующим интерпретатором.

Использование интерпретатора (а значит и PHP) имеет свои неоспоримые преимущества:

* Нет необходимости заботится об освобождении выделенной памяти, не нужно закрывать файлы по окончании работы с ними – всю рутинную работу сделает интерпретатор, поскольку программа выполняется под его бдительным контролем;
* Не нужно думать о типах переменных, а также не нужно объявлять переменную до его первого использования;
* Отладка программ и обнаружение ошибок существенно упрощаются – интерпретатор полностью контролирует этот процесс;
* В контексте web-приложений, интерпретатор также имеет еще очень важное преимущество – нет опасности «зависания» сервера при неправильной работе программы.

**PHP и базы данных**

MySQL - это популярная система управления базами данных (СУБД), очень часто применяемая в сочетании с PHP.

MySQL - это система управления реляционными базами данных. В реляционной базе данных данные хранятся не все скопом, а в отдельных таблицах, благодаря чему достигается выигрыш в скорости и гибкости.

MySQL - это ПО с открытым кодом. Применять его и модифицировать может любой желающий. Такое ПО можно получать по Internet и использовать бесплатно.

Технические возможности СУБД MySQL MySQL является системой клиент-сервер, которая содержит многопоточный SQL-сервер, обеспечивающий поддержку различных вычислительных машин баз данных, а также несколько различных клиентских программ и библиотек, средства администрирования и широкий спектр программных интерфейсов (API).

**Устройство MySQL**

MySQL состоит из двух частей: серверной и клиентской.

Сервер MySQL постоянно работает на компьютере. Клиентские программы посылают серверу MySQL SQL-запросы через механизм, сервер их обрабатывает и запоминает результат. То есть скрипт (клиент) указывает, какую информацию он хочет получить от сервера баз данных. Затем сервер баз данных посылает ответ клиенту.

Cтруктура MySQL трехуровневая: базы данных — таблицы — записи. Базы данных и таблицы MySQL физически представляются файлами с расширениями frm, MYD, MYI. Логически - таблица представляет собой совокупность записей. А записи - это совокупность полей разного типа.