**Лекция 2**

**Тема: Возможности PHP**

" ***PHP*** может все", – заявляют его создатели. В первую очередь *PHP* используется для создания *скриптов*, работающих на стороне *сервера*, для этого его, собственно, и придумали. *PHP* способен решать те же задачи, что и любые другие *CGI* - *скрипты*, в том числе обрабатывать данные html-форм, динамически генерировать html страницы и т.п. Но есть и другие области, где может использоваться *PHP* . Всего выделяют три основные области применения *PHP*.

* Первая область, как уже говорилось, – это создание приложений ( *скриптов* ), которые исполняются на стороне *сервера*. *PHP* наиболее широко используется именно для создания такого рода *скриптов*. Для того чтобы работать таким образом, понадобится *PHP* -*парсер* (т.е. обработчик *php* - *скриптов* ) и *web-сервер* для обработки *скрипта*, браузер для просмотра результатов работы *скрипта*, ну, и, конечно, какой-либо текстовый редактор для написания самого *php* -кода. *Парсер* *PHP* распространяется в виде *CGI* - *программы* или серверного модуля. Как *установить* его и *web-сервер* на свой компьютер, мы рассмотрим немного позднее. В этом курсе мы будем обсуждать, как правило, создание именно серверных приложений, как пример использования языка *PHP*.
* Вторая область – это создание *скриптов*, *выполняющихся в командной строке*. То есть с помощью *PHP* можно создавать такие *скрипты*, которые будут исполняться, вне зависимости от *web-сервера* и браузера, на конкретной машине. Для такой работы потребуется лишь *парсер* *PHP* (в этом случае его называют *интерпретатором* командной строки (*cli*, command line *interpreter*)). Этот способ работы подходит, например, для *скриптов*, которые должны выполняться регулярно с помощью различных *планировщиков задач* или для решения задач простой обработки текста.
* И последняя область – это создание *GUI* -приложений (графических интерфейсов), выполняющихся на стороне клиента. В принципе это не самый лучший способ использовать *PHP*, особенно для начинающих, но если вы уже досконально изучили *PHP*, то такие возможности языка могут оказаться весьма полезны. Для применения *PHP* в этой области потребуется специальный инструмент – *PHP-GTK*, который является расширением *PHP*.

Итак, область применения *PHP* достаточно обширна и разнообразна. Тем не менее существует множество других языков программирования, способных решать похожие задачи. Почему стоит изучать *PHP*? Что это нам дает? Во-первых, *PHP* очень прост в изучении. Достаточно ознакомиться лишь с основными правилами синтаксиса и принципами его работы, и можно начинать писать собственные *программы*, причем браться за такие задачи, решение которых на другом языке требовало бы серьезной подготовки.

Во-вторых, *PHP* поддерживается почти на всех известных платформах, почти во всех операционных системах и на самых разных *серверах*. Это тоже очень важно. Вряд ли кому-то захочется переходить, например, от работы под *Windows* к работе под Linux или от *сервера* *IIS* к *серверу**Apache* только для того, чтобы изучить еще один *язык программирования*.

В *PHP* сочетаются две самые популярные *парадигмы программирования* – объектная и процедурная. В *PHP4* более полно поддерживается *процедурное программирование*, но есть возможность писать *программы* и в объектном стиле. Уже в первых пробных версиях *PHP5* большинство недочетов в реализации объектно-ориентированной модели языка, существующих в *PHP4*, устранены. Таким образом, можно выбрать наиболее привычный стиль работы.

Если говорить о возможностях сегодняшнего *PHP*, то они выходят далеко за рамки тех, что были реализованы в его первых версиях. С помощью *PHP* можно создавать изображения, *PDF* -файлы, флэш-ролики, в него включена *поддержка* большого числа современных баз данных, встроены функции для работы с текстовыми данными любых форматов, включая *XML*, и функции для работы с файловой системой. *PHP* поддерживает взаимодействие с различными сервисами посредством соответствующих протоколов, таких как *протокол управления* доступом к директориям *LDAP*, протокол работы с сетевым оборудованием *SNMP*, протоколы передачи сообщений *IMAP*, *NNTP* и *POP3*, *протокол передачи гипертекста* *HTTP* и т.д.

Обращая внимание на взаимодействие между различными языками, следует упомянуть о поддержке объектов *Java* и возможности их использования в качестве объектов *PHP*. Для доступа к удаленным объектам можно использовать расширение *CORBA*.

Для работы с текстовой информацией *PHP* унаследовал (с небольшими изменениями) *механизмы* работы с регулярными выражениями из языка Perl и *UNIX*-систем. Для обработки *XML* -документов можно использовать как стандарты *DOM* и *SAX*, так и *API* для *XSLT* -трансформаций.

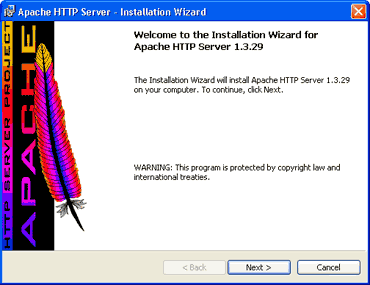
Для создания приложений *электронной коммерции* существует ряд полезных функций, таких как функции осуществления платежей *Cybercash*, *CyberMUT*, *VeriSign Payflow Pro* и *CCVS*.

### Установка и настройка ПО

Возможности языка мы обсудили, области применения рассмотрели, *историю* изучили. Теперь можно приступать к *установке* необходимого инструментария. Поскольку в качестве практической основы курса мы будем рассматривать задачи, решаемые с помощью технологии клиент- *сервер*, и *PHP* соответственно будет использоваться для создания *скриптов*, обрабатываемых *сервером*, нам нужно *установить**web-сервер* и *интерпретатор* *PHP*. В качестве *web-сервера* выберем, например, *Apache*, как наиболее популярный среди web-разработчиков. Для просмотра результатов работы *программ* нам понадобится web-*браузер*, например *Internet* Explorer.

#### Установка Apache 1.3.29 под Windows XP

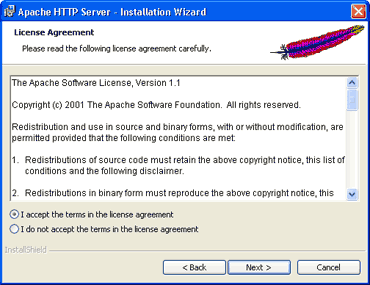
Чтобы что-нибудь *установить*, нужно для начала иметь соответствующее программное обеспечение (ПО). Скачать ПО для *установки* *Apache* можно, например, с его официального сайта [http://www.apache.org](http://www.apache.org/). Мы скачали файл apache\_1.3.29-*win32*-x86-no\_src.exe. Это автоматический установщик (иначе – wizard) *сервера**Apache* под Windows. Эта *программа* попытается почти самостоятельно (а точнее, с минимальными усилиями с вашей стороны) *установить* на компьютер какое-либо программное обеспечение, а в данном случае *сервер*. После запуска файла установщика появляется следующее окошко ([рис. 1.1](https://intuit.ru/studies/courses/42/42/lecture/27175?page=2#image.1.1)).



**Рис. 1.1.**Автоматическая установка сервера Apache

Чтобы *установить* HTTP- *сервер**Apache* версии 1.3.29 на свой компьютер, нужно нажать на кнопку Next. Кстати говоря, эта же *программа* позволит изменить или удалить уже *установленный**web-сервер*.

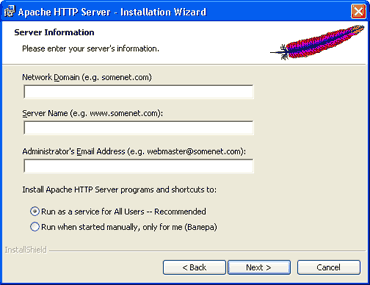
После нажатия кнопки Next *программа* предложит согласиться с условиями лицензии ([рис. 1.2](https://intuit.ru/studies/courses/42/42/lecture/27175?page=2#image.1.2)).



**Рис. 1.2.**Лицензионное соглашение

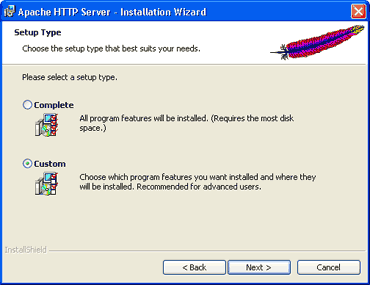
Следующий экран будет содержать информацию о *сервере**Apache*, и в частности о его Windows-реализации (его изображение не приводим).

На следующем шаге нужно ввести имя сетевого домена, имя *сервера* и e-mail администратора. *Программа* попытается автоматически определить ваш домен и хост по *настройкам* компьютера ([рис. 1.3](https://intuit.ru/studies/courses/42/42/lecture/27175?page=2#image.1.3)).



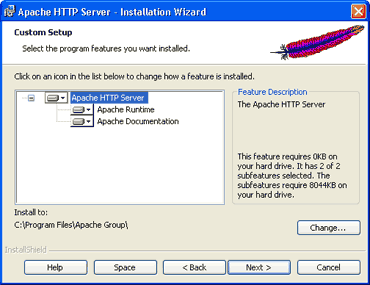
**Рис. 1.3.**Основная информация о сервере

После того как вы ввели данные в вышеприведенную форму, нужно выбрать тип *установки*: полная (устанавливаются все компоненты *сервера* ) или определяемая пользователем (можно выбрать компоненты для *установки* ) ([рис. 1.4.](https://intuit.ru/studies/courses/42/42/lecture/27175?page=2#image.1.4)).



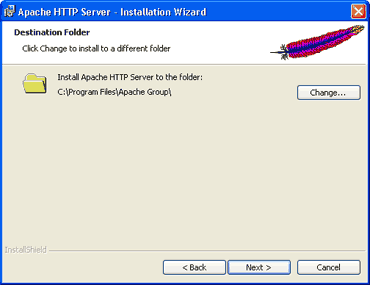
**Рис. 1.4.**Тип установки

Выбор компонентов *сервера* не очень большой – это инструменты, необходимые для работы *сервера* и документация к нему ([рис. 1.5](https://intuit.ru/studies/courses/42/42/lecture/27175?page=2#image.1.5)).



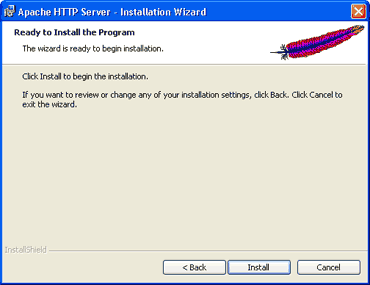
**Рис. 1.5.**Выбор компонент пользовательской установки

Мы выберем полную *установку*. Тогда на следующем шаге *программа* предложит выбрать папку, в которую будет *установлен* *сервер*. По умолчанию *сервер* устанавливается в папку c:\Program Files\Apache Group\ ([рис. 1.6](https://intuit.ru/studies/courses/42/42/lecture/27175?page=2#image.1.6)).



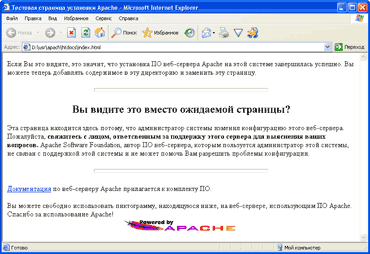
**Рис. 1.6.**Папка, в которую будет установлен сервер

На следующем экране ([рис. 1.7](https://intuit.ru/studies/courses/42/42/lecture/27175?page=2#image.1.7)) потребуется подтвердить правильность введенных данных и начать *установку*. Из любого окна *установки*, включая и это, можно вернуться назад и изменить введенные ранее данные.



**Рис. 1.7.**Начало установки

После нажатия кнопки Install начнется непосредственная *установка* *сервера*. От пользователя никаких дополнительных действий больше не требуется. С одной стороны, это хорошо, но есть у такой автоматизированной *установки* и некоторые недостатки. Например, *домашние директории* пользователей оказываются там же, где и *файлы настроек* *сервера* ( c:\Program Files\Apache Group\Apache\users\ ). Это небезопасно, если на компьютере работает несколько пользователей, не являющихся администраторами *сервера*. Но для начала можно ничего не менять. Допустим, мы запустили *установщик*, ввели все необходимые данные, он выполнил все операции успешно и говорит, что *сервер**установлен*. Как проверить, действительно ли *сервер**установлен*? Набираем в окне браузера http://localhost/. Если все *установилось* правильно, мы получим страничку приветствия *сервера**Apache* (см. [рис. 1.8](https://intuit.ru/studies/courses/42/42/lecture/27175?page=2#image.1.8)).

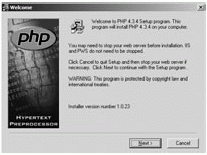


**Рис. 1.8.**Страница приветствия

Итак, *сервер**установлен*. Как теперь с ним работать? Откуда можно запускать *скрипты* и где должны находиться файлы пользователей? Файлы, которые должны быть обработаны *сервером*, можно сохранять либо в корне *сервера* (в нашем случае это c:\Program Files\Apache Group\Apache\htdocs ), либо в директориях пользователей (в нашем случае это c:\Program Files\Apache Group\Apache\users\ ). Местоположение корня *сервера* и директорий пользователей прописано в *настройках* *сервера*, а точнее, в файле конфигурации *httpd.conf* (найти его можно в c:\Program Files\Apache Group\Apache\conf ). Для изменения этих путей нужно изменить соответствующие переменные в файле конфигурации *сервера*. Переменная в файле конфигурации ServerRoot отвечает за корневую директорию *сервера*, а переменная UserDir – за директорию, где будут располагаться файлы пользователей *сервера* (для более безопасной работы советуем изменить переменную UserDir на что-нибудь типа c:\users\ ). Чтобы получить доступ к файлу test.html, находящемуся в корне *сервера*, нужно набрать в браузере http://localhost/test.html (т. е. *имя хоста*, имя файла). Если же файл test.html находится в директории пользователя user, то для его просмотра нужно набрать в браузере http://localhost/~user/test.html.

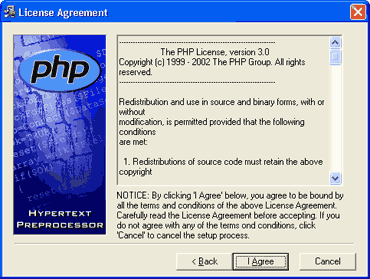
#### Установка PHP 4.3.4 под Windows

Перейдем к *установке* *PHP*. Скачать его *дистрибутив* можно с официального сайта *PHP* – [http://www.php.net](http://www.php.net/). Для удобства лучше выбрать автоматический инсталлятор, как и в случае с *сервером*. Самое первое окошко при такой *установке* *PHP* содержит приветствие и предупреждение о существовании авторских прав на этот продукт ([рис. 1.9](https://intuit.ru/studies/courses/42/42/lecture/27175?page=3#image.1.9)).



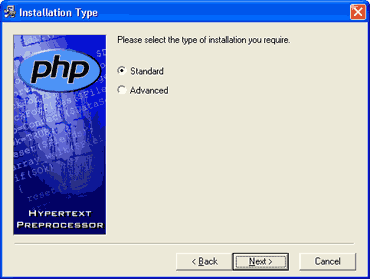
**Рис. 1.9.**Программа автоматической установки PHP

Далее нужно прочитать и согласиться (или не согласиться) с лицензией ([рис. 1.10](https://intuit.ru/studies/courses/42/42/lecture/27175?page=3#image.1.10)).



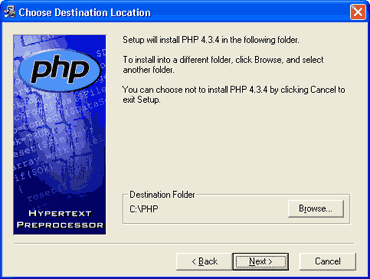
**Рис. 1.10.**Лицензионное соглашение

Затем выбираем тип *установки*: стандартный или для специалистов. Мы выберем стандартную *установку* ([рис. 1.11](https://intuit.ru/studies/courses/42/42/lecture/27175?page=3#image.1.11)).



**Рис. 1.11.**Тип установки

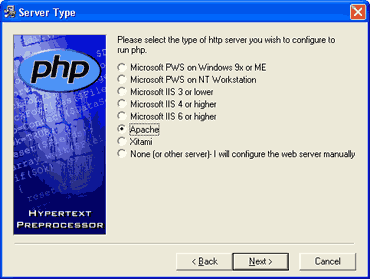
В следующем окне *программа* предложит выбрать директорию для *установки* *PHP*. Туда будут скопированы файлы библиотек, расширений, *интерпретатор* командной строки, модули и т.п. ([рис. 1.12](https://intuit.ru/studies/courses/42/42/lecture/27175?page=3#image.1.12)).



**Рис. 1.12.**Выбор папки, в которую будет установлен PHP

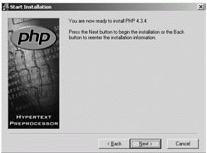
В следующем окошке предлагается ввести адрес SMTP- *сервера*, т.е. *сервера* отправки почтовых сообщений, и обратный адрес отправителя почты. Это окошко мы приводить не будем.

Далее предлагается выбрать из списка *сервер*, с которым будет работать *PHP*. Поскольку мы *установили* *Apache*, то выберем из списка именно его ([рис. 1.13](https://intuit.ru/studies/courses/42/42/lecture/27175?page=3#image.1.13)).



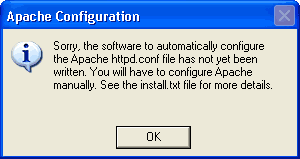
**Рис. 1.13.**Выбор сервера, на котором будет работать PHP

Следующее окошко начинает инсталляцию ([рис. 1.14](https://intuit.ru/studies/courses/42/42/lecture/27175?page=3#image.1.14)).



**Рис. 1.14.**Начало автоматической установки PHP

После копирования необходимых файлов мы получили следующее сообщение (см. [рис. 1.15](https://intuit.ru/studies/courses/42/42/lecture/27175?page=3#image.1.15)):



**Рис. 1.15.**Сообщение о невозможности автоматически настроить Apache для работы с PHP

Это значит, что нужно настраивать *сервер**Apache* для работы с *PHP* вручную.

Сначала следует выбрать, как мы хотим *установить* *PHP*, поскольку он поставляется в двух видах: *CGI-скрипт* ( php.exe ) или набор *SAPI-модулей* (например, php4isapi.dll ), используемых *сервером*. Последний вариант обладает новыми возможностями, однако из-за недостаточной проработанности может функционировать недостаточно надежно, особенно на платформах старше Windows 2000 (может появляться куча ошибок 500, могут возникать сбои в других *серверных* модулях, таких как ASP). Так что, если нужна абсолютная стабильность – надо выбирать *установку* *PHP* в виде *CGI* выполнимого приложения.

Если мы хотим *установить* *PHP* как серверный модуль, то в файле конфигурации *сервера* ( *httpd.conf* ) нужно написать:

LoadModule php4\_module

c:/php/sapi/php4apache.dll

AddType application/x-httpd-php .php .phtml

AddModule mod\_php4.c

Если мы устанавливаем *PHP* как *cgi* - *программу*, то в *httpd.conf* нужно написать:

ScriptAlias /php/ "c:/php/"

AddType application/x-httpd-php .php .phtml

Action application/x-httpd-php "/php/php.exe"

В этом случае могут возникнуть проблемы с безопасностью. Рекомендуется исправить директорию, где лежит исполняемый файл *интерпретатора* ( c:\php\ ), на что-нибудь менее очевидное (например, на c:\abc\_php\ ). Мы советуем устанавливать *PHP* как серверный модуль.

Еще нужно отредактировать файл *php.ini* (в папке c:\Windows ), чтобы *PHP* "знал", где находится корневая директория *сервера*, где пользовательские директории, а где его собственные библиотеки расширений. За это в файле *php.ini* отвечают соответственно переменные ***doc\_root*** , ***user\_dir*** и ***extension\_dir*** . Зададим их таким образом:

doc\_root = "c:\Program Files\Apache

Group\Apache\htdocs"

user\_dir = "c:\users"

extension\_dir = "c:\php\extensions"

Кроме того, можно выбрать расширения, которые будут загружаться при запуске *PHP*. В реализацию *PHP* под Windows изначально входит очень мало расширений. Чтобы загрузить расширение, нужно раскомментировать в *php.ini* соответствующую ему строчку 'extension=php\_\*.dll'. Например, чтобы загрузить расширение для работы с MSSQL, нужно раскомментировать строку 'extension=php\_mssql.dll'. Некоторые расширения требуют дополнительных библиотек. Поэтому рекомендуется скопировать дополнительные библиотеки в системную директорию (из папки c:\php\dlls ). При первой *установке* следует настроить и протестировать *PHP* без расширений.

Для того чтобы *настройки*, выполненные в конфигурационных файлах *сервера* и *PHP* вступили в силу, нужно перезапустить *сервер*.

Проверим, работает ли *PHP*. Для этого создадим тестовый файл ( 1.php ) в директории пользователя ( c:\users\nina ) со следующим содержанием:

<?php

echo"<h1>Привет всем!</h1>";

?>

Запустим этот файл через браузер, набрав <http://localhost/~nina/1.php>. Если что-то не так, то на экран будет выведен текст этого файла. Если все хорошо, то наш *скрипт* должен обработаться *сервером* и вывести большими буквами строку "Привет всем!".