ЛАБОРАТОРНАЯ РАБОТА №1

**По курсу «Интернет-программирование»**

**Цель работы.**

Ознакомиться с принципами работы ресурсов в сети Интернет и методами разработки в сети Интернет. Практически освоить развертывание инструментария Интернет-разработчика и работу с ним путем выполнения лабораторного задания.

**Приобретаемые компетенции:**

1. Понимание основных принципов работы ресурсов в сети Интернет.
2. Развертывание и настройка рабочего пространства Интернет-разработчика.
3. Верстка многоколоночного макета сайта с применением языков HTML и CSS.

# Принципы работы ресурсов в сети Интернет. Архитектура «клиент-сервер»

В общепринятом понимании сеть Интернет представляет собою сеть клиентских узлов (потребителей информации) и сеть серверов (источников информации). Клиентские узлы в сети Интернет в общем случае не связываются друг с другом напрямую. Все операции они осуществляют посредством серверов. Эта архитектура работы называется архитектурой «клиент-сервер». Сервера могут взаимодействовать друг с другом напрямую.

Первоначально сеть Интернет, понималась только как сеть сайтов. Но сейчас под термином сеть Интернет уже понимается практически все, что работает в архитектуре «клиент-сервер» и может быть доступно для удаленного взаимодействия т.е. по каналам связи.

Поэтому программным обеспечением, обеспечивающим работу клиентских узлов может быть: браузер (для сайтов), специальное приложение (например, Skype), мобильное приложение (для смартфонов).

Программное обеспечение, обеспечивающее работу сервера для Интернет-приложения типа сайт, как правило представлено:

1. Web-сервером (например Apache), обеспечивающим прием и распределение запросов клиентов между рабочими приложениями сервера.
2. Средой исполнения кода прикладного приложенияна сервере (например, PHP) исполняющая запрос и возвращающее результат web-серверу для отправки клиенту
3. Файлами кода прикладного приложения на языке среды исполнения кода (например, PHP)
4. Базой данных (например, MySQL) используемой для хранения данных прикладных приложений сервера
5. Средствами повышения производительности. Например, кеширующий сервер nginx, акселератор PHP Zend и т.д.
6. Средствами расширения функциональности. Например, библиотеки и модули для Apache и PHP
7. Средствами управления и контроля. Например, phpMyAdmin для визуального управления базой MySQL, обработчики логов web-сервера и т.д.

Само прикладное приложение может быть написано как на языке «скриптового типа», например: PHP, Perl (т.е. интерпретируемом), так и на «компилируемых» языках, например Java. При использовании интерпретируемых языков прикладное приложение содержится в файлах в исходном коде и компилируется интерпретатором в момент вызова web-сервером. При использовании компилируемых языков — приложение уже заранее откомпилировано в машинный или промежуточный код.

Скриптовые языки проще в сопровождении т. к. для изменения приложения достаточно изменить один файл, который будет откомпилирован непосредственно при запуске и, поэтому нет необходимости компилировать все приложение, которое может быть достаточно объемным и сложным. Но, очевидно, интерпретация существенно (в 20-30 раз) замедляет скорость работы прикладных приложений. Для преодоления этих недостатков для скриптовых языков, например PHP, разработаны разного рода «ускорители» и «прекомпиляторы»: eAccelerator, Zend Opimizer и другие.

Смысл работы этих «ускорителей» заключается в том, чтобы оптимизировать стадии исполнения сценариев скриптовых языков. Например, обработка сценария на PHP обычным интерпретатором выглядит следующим образом:

1. Чтение файла
2. Генерация байткода
3. Выполнение кода
4. Выдача результата

Процесс генерации байткода (т.е. компиляция в машинный код данного сервера) выполняется каждый раз и отнимает значительную часть времени обработки файла. Избежать пунктов 3 и 4 — нельзя, а пункты 1 и 2 возможны для оптимизации.  
  
Например, «ускоритель PHP» eAccelerator после выполнения пункта 2 сохраняет откомпилированную версию кода PHP и при последующих вызовах этого же кода уже может сразу запустить на исполнение этот код. При этом, понятно, что eAccelerator должен следить за изменением файла, чтобы если файл был изменен разработчиком, заново откомпилировать его. Поэтому пункт 1 сокращается только в части чтения содержимого файла, но проверяется дата изменения файла и/или контрольная сумма. Кроме того, понятно, что eAccelerator занимает оперативную память на хранение откомпилированных файлов, что при определенных условиях настройки хостинга и распределения нагрузки может привести не к ускорению, а наоборот, к замедлению работы. Если же памяти достаточно то ускорение в лучших условиях может составить до 10 раз. В целом же в Интернете можно встретить информацию о том, что при определенных условиях рост производительности с помощью «ускорителей» достигал 30-40 раз т.е. становился практически равным производительности заранее откомпилированного кода.

Существуют «ускорители» и для web-серверов, но механизм ускорения здесь достигается за счет экономии ресурсов. Например web-сервер Apache при выполнении запроса даже статического контента: изображений, файлов стилей (CSS) все равно «поднимает» интерпретатор PHP. Очевидно, что при загрузке картинки тратить ресурсы на загрузку интерпретатора PHP не нужно. Поэтому перед Web-сервером приложений ставят кеширующий сервер (например Nginx), который запросы на отдачу статического контента обрабатывает сам (и очень быстро), а запросы требующие исполнения скриптов пропускает на вход web-сервера, и потом также пропускает обратно готовый ответ.

## Понятие сессии

Весь процесс взаимодействия клиента и сервера со стороны сервера называется сессией (session). При обращении клиента к серверу сессия создается, ей присваевается уникальный идентификатор (session id), выделяются ресурсы под ее обработку, запрос исполняется и возвращается клиенту вместе с идентификатором его сессии на сервере. При этом на стороне клиента идентификатор тем или иным образом сохраняется и при последующих запросах сервер уже знает какой клиент к нему обращается. Сервер как правило, настроен так, чтобы не закрывать сессию клиента после первого запроса т.к. процесс создания сессии достаточно ресурсоемкий. Вместо этого сервер некоторое время ожидает следующих запросов от клиента. Это «некоторое время» называется таймаутом (timeout) сессии. Если клиент до истечения таймаута отправил еще один запрос, то сессия продлевается еще на величину таймаута. Если клиент направляет запрос после истечения таймаута сессии предыдущего запроса, то создается новая сессия, которая (без специальных действий) ничего не знает о предыдущей истекшей сессии данного клиента. Обычно таймаут web-сервера составляет от 3-х до 15 минут.

Кроме таймаута сессии (т.е. по сути таймаута web-сервера) таймаут существует у интерпретатора и базы данных. Назначение этих таймаутов — предотвратить ситуации зависания от неограниченного «пожирания» ресурсов. Например, ошибочно написанный код PHP может инициировать бесконечный цикл. Понятно, что такой зависший скрипт необходимо прервать.  
  
Все таймауты: web-сервера, интерпретатора PHP и СуБД MySQL можно настроить в соответствующих конфигурационных файлах.

Если адрес конкретной страницы не указан, то вызывается индексная страница папки. Для web-сервера Apache это index.\*, а для IIS это default.\*.

## Виды хостинга

Принципиально существует 3 вида хостинга: виртуальный хостинг, виртуальный сервер (VDS), выделенный сервер.  
  
В случае виртуального сервера несколько десятков (а то и сотен) сайтов находятся на одном физическом сервере и разделение ресурсов между ними производится программно. Это может привести к тому, что некоторый плохо настроенный или высоконагруженный сайт на некоторое время может «захватить» много ресурсов и в этот период мешать работе других сайтов.

В случае виртуального сервера, разделение ресурсов производится аппаратно-программно, что гарантирует, что каждый контракт (сайт) получит оговоренное количество ресурсов и не будет способен помешать другим сайтам.

В случае выделенного сервера все ресурсы сервера находятся в распоряжении заказчика. Кроме того, заказчик может полность настраивать ПО сервера по своим потребностям.

## Проблематика разработки и сопровождения ПО в архитектуре «Клиент-сервер»

В архитектуре «клиент-сервер» код исполняется на удаленном сервере, а изменяется, как правило, на компьютере разработчика. Это, очевидно, затрудняет тестирование и отладку кода Интернет-приложений.

Как правило работу с кодом разработчики осуществляют по протоколу FTP (SFTP). В этом случае экземпляр файла скачивается на компьютер разработчика и после правки закачивается обратно на сервер. Разработчик вызывает в броузере страницу с файлом и смотрит результат. Как правило, по умолчанию интерпертатор настроен так, чтобы возвращать на экран строку ошибки в случае ее возникновения.

Понятно, что такой метод отладки кода — неудобен. Как минимум синтаксические ошибки можно проверить и на стороне клиента. Поэтому появились редакторы кода, например, PHPExpertEditor. Подобный редактор можно связать с сайтом по протоколу FTP и при выборе любого файла, он загружается в редактор локально для правки, а после правки выгружается обратно на сайт. Данный инструментарий позволяет проверить синтаксические ошибки, но не помогает искать логические ошибки (отладка) и ошибки производительности.

Поэтому вслед за редакторами пришли интеллектуальные среды (системы) разработки — IDE (Intelege Development Environment), которые позволяет выполнять такие операции как: отладка, профилирование, рефакторинг сборка и выгрузка проекта в репозиторий, контроль версий.

Среди IDE для целей курса выбрана среда NetBeans, которая позволяет бесплатно работать с PHP, JavaScript, HTML, CSS, Java.

Данная среда позволяет создавать на машине клиента файловую структуру полностью или частично (для исключения лишнего) отражающая структуру файлов приложения на удаленном сервере. Работа с удаленным сайтом осуществляется по протоколам: FTP (порт 21) или SFTP (порт 22). Существуют инструменты синхронизации и защищенного удаления (проверяется, что файл не связан с другими файлами)

# Практическая часть

Целью лабораторной работы является приобретение понимания основных принципов работы Интернет-приложений путем практической установки и настройки локальной среды Интернет-разработчика, включающей локальную установку и настройку:

1. Web-сервер Apache
2. Язык сценариев PHP
3. Базу данных MySQL и средства администрирования этой БД — PhpMyAdmin
4. Среду разработки NetBeans
5. По выбору одну и систем управления контентом (CMS): Битрикс, WordPress или Joomla.

Все инструменты будут устанавливаться локально на виртуальную машину Windows8\_IPOVS, файл которой найти по адресу: **BACKUP(N:)/VM/Windows8\_IPOVS**. Параметры доступа к виртуальной машине: логин — user, пароль — 1q2w3e4r5t.

Все необходимые дистрибутивы для установке находятся на диске BACKUP(N:)

Первые три пункта представляют собою по сути локальную среду исполнения в которой будут работать Интернет-приложения данного курса. Для выполнения этих пунктов удобно воспользоваться готовым инструментов развертывания — OpenServer (<https://ospanel.io/>). OpenServer уже развернут на виртуальной машине, порядок запуска показан на рис.1.

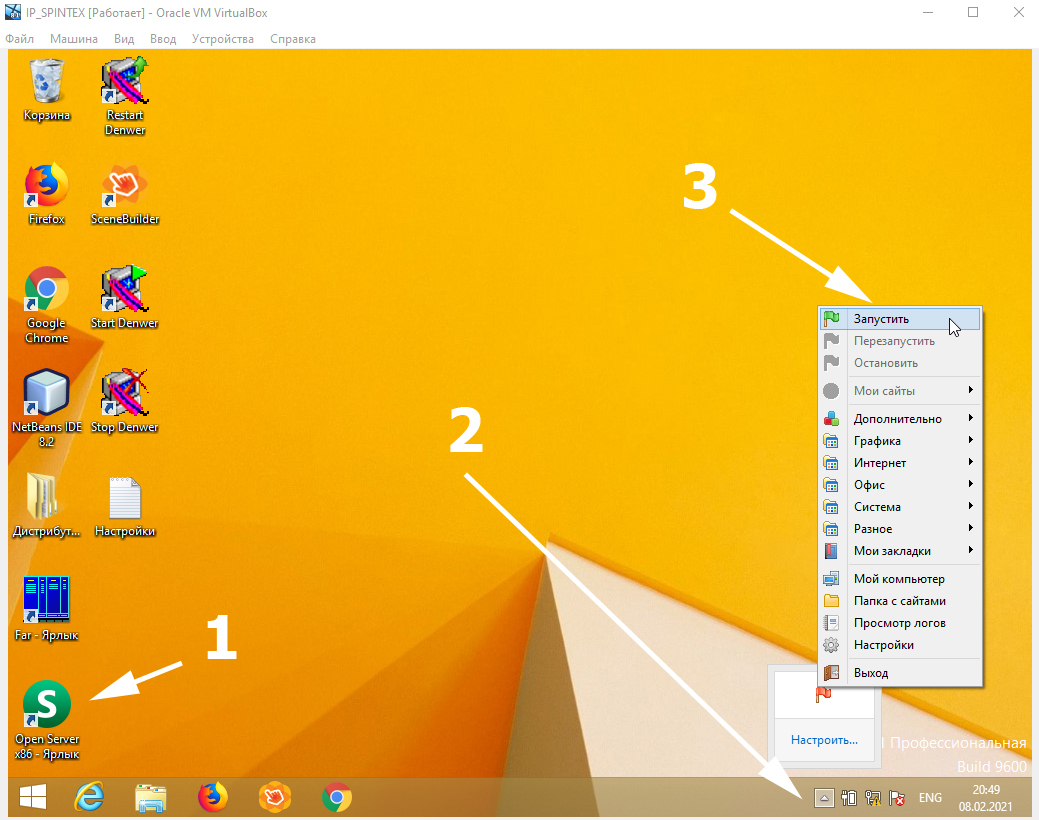


Рис.1. Порядок запуска OpenServer. 1. Размещение OpenServer списке скрытых значков. 2. Нажатие на красный флаг запускает меню управления OpenServer. 3. Нажатие на зеленый флаг в меню управления запускает среду OpenServer. В случае успешного запуска, флаг OpenServer в списке скрытых значков становится зеленым.

При запуске OpenServer OC Windows может потребовать разрешить OpenServer доступ к некоторым портам и файлу hosts. В случае успешного запуска по адресу [http://localhost](http://localhost/), должно отобразиться сообщение OpenServer, как показано на рис. 2.

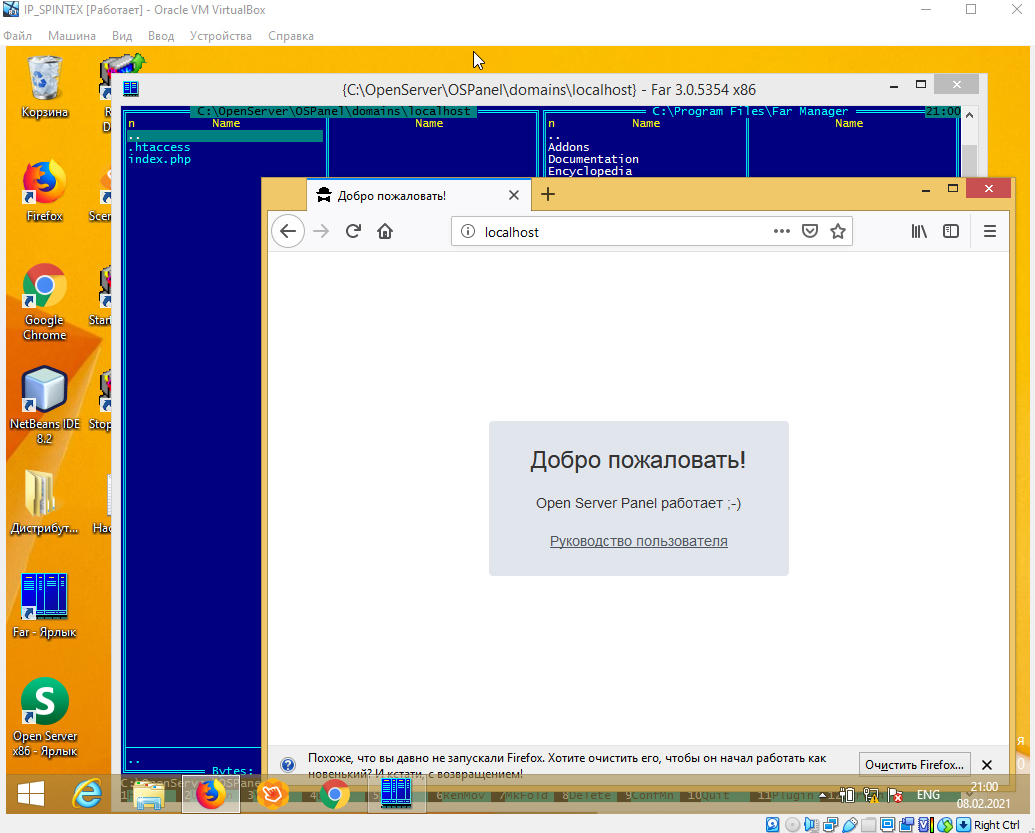


Рис.2. Страница по адресу localhost в случае успешного запуска OpenServer.

Не редко вместо этого экрана на компьютерах с Windows можно наблюдать окно, возвращаемое IIS (Internet Information Server). Чтобы web-сервер Apache из установки OpenServer смог перехватить обращения к localhost нужно отключить службу IIS по адресу: «Панель управления / Все элементы панели управления / Программы и компоненты / Службы IIS». После чего не забыть нажать «ОК», чтобы изменения вступили в силу.

После первого успешного запуска OpenServer создаст по адресу **C:\OpenServer\OSPanel\** папку domen в которой можно размещать папки сайтов. По умолчанию в папке **C:\OpenServer\OSPanel\domen\** автоматически создается папка localhost с файлом index.php, который вызывается по адресу [http://localhost](http://localhost/).

## Настройка локальных адресов.

Если запуск OpenServer на этом этапе прошел успешно, прежде установки и настройки IDE NetBeans, следует произвести некоторые настройки компонентов среды исполнения. В первую очередь нужно проверить, что успешно создаются новые домены.

Для этого в корне папки **C:\OpenServer\OSPanel\domen\** создается любая папка, например test.ru, а в ней папка www и индексный файл index.php, например, следующего содержания:

Листинг 1. Тестовая PHP страница.

<!DOCTYPE html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Проверка</title>

</head>

<body>

<center>Кодировка</center>

<? phpinfo(); ?>

</body>

</html>

После этого нужно перезапустить OpenServer, чтобы он «нашел» новый сайт, а затем ввести в адресной строке броузера адрес test.ru. Если все настроено правильно, то должно отобразиться русское слово «Кодировка» и таблица с действующими настройками интерпретатора PHP. Если вместо русских букв вы увидите «абракадабру», то необходимо настроить кодировку. Как это делается описано в параграфе «Настройка кодировки»

Соответствие символьных имен адресов папкам на диске компьютера задается в ОС Windows в файле – **C:\windows\System32\drivers\etc\hosts**. Локальный IP адрес адрес компьютера – 127.0.0.1 Символьное им, соответствующее этому адресу — localhost.

При установке и перезапуске OpenServer проверяет список папок сайтов в папке web-сервера и соответствующим образом исправляет файл hosts.

Файл hosts, измененный OpenServer может выглядеть так:

127.0.0.1 localhost

127.0.0.1 www.nbproject.bitrixtest.ru

127.0.0.1 www.nbproject.Postyplenie

127.0.0.1 nbproject.bitrixtest.ru

127.0.0.1 www.subdomain.localhost

127.0.0.1 www.subdomain.test1.ru

Это означает, что все обращения по указанным символьным именам будут пересылаться на IP 127.0.0.1, который слушает web-сервер Apache, запущенный OpenServer и который будет запускать соответствующий прикладной код для каждого сайта (символьного имени) из hosts.

Рассмотрим далее некоторые полезные настройки сервисов: Apache, MySQL, PHP5

## Настройка web-сервера Apache

OpenServer предоставляет несколько конфигураций запуска web-сервера Apache. Чтобы узнать, какая конфигурация запущена нужно войти в мену управления OpenServer, выбрать пункт “Настройки”, в появившемся окне “Настройки” выбрать закладку “Модули”, а в ней раскрыть выпадающий список серверов под заголовком HTTP (см. рис. 3.)

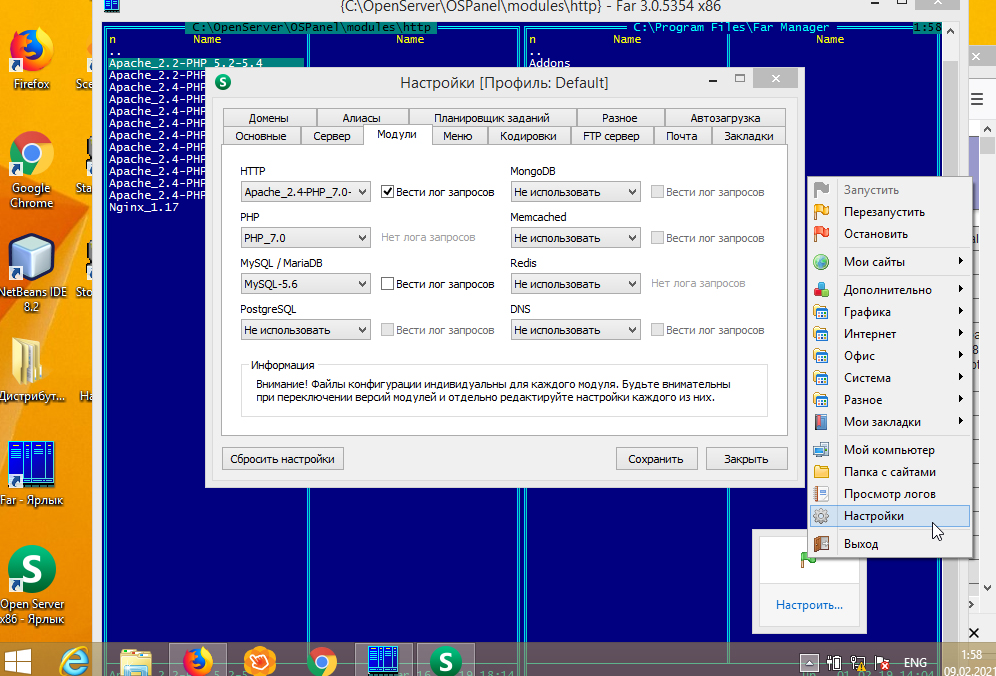


Рис. 3. Вызов окна настроек OpenServer.

В примере на рис.3. выбрана конфигурация Apache\_2.4\_PHP\_7.0\_7.1. Конфигурационный файл web-сервера находится по адресу: **C:\OpenServer\OSPanel\modules\http\Apache\_2.4\_PHP\_7.0\_7.1\conf\httpd.conf**. После внесения изменений в конфигурационный файл web-сервер надо перезапустить. В случае использования OpenServer ручное редактирование данного файла бессмысленно т.к. он перезаписывается перед каждым запуском в соответствии с настройками в окне настроек OpenServer. Значения настроек подставляются в шаблоны конфигурационных файлов по адресу **C:\OpenServer\OSPanel\userdata\config**. Например, для сервера Apache\_2.4\_PHP\_7.0\_7.1, шаблоном конфигурационного файла является файл **C:\OpenServer\OSPanel\userdata\config\Apache\_2.4-PHP\_7.0-7.1\_server.conf**. Внутри этого файла шаблоны вида: %ssitedir%, при перезапуске OpenServer, заменяются на значения переменных из окна настроек OpenServer (например, %ssitedir% на c:/openserver/ospanel/domains) и готовый конфигурационный файл записывается в папку conf стартующего сервера. В нашем случае для перезапуска всех сервисов достаточно перезапустить OpenServer, который и перезапустит все сервисы.

В конфигурационном файле web-сервера настраиваются подключаемые модули, папки логирования, кодировка, возвращаемая сервером в ответе на запрос и другие параметры. Поскольку каждый подключаемый модуль увеличивает количество памяти, занимаемой сессией — рекомендуется отключать неиспользуемые модули.

Web-сервер ведет логирование запросов (если в панели настроек выбрана “галочка” у пункта “Вести лог запросов”). Обычно ведется два лога: ошибок - **C:\OpenServer\OSPanel\userdata\logs\ Apache\_2.4-PHP\_7.0-7.1\_errora.lo** и запросов - **C:\OpenServer\OSPanel\userdata\logs\Apache\_2.4-PHP\_7.0-7.1\_ queriesa.lo**. Эти логи являются «человекочитаемыми», однако их статистическую обработку с целями исследования посещаемости сервера и распределения нагрузки и ошибок удобно вести с помощью специального программного обеспечения, например — **Webalizer** (<http://www.webalizer.org/>) и т.д.

Кроме общей настройки веб-сервера по умолчанию, существует также возможность изменения этих настроек для каждого конкретного сайта с помощью специального файла **.htaccess**, который размещается в корневой папке сайта и содержит инструкции, аналогичные инструкциям в файле httpd.conf, а также может содержать специальные инструкции, например, для переопределения адресов с целью организации ЧПУ (“человекопонятного” URL), для паролирования доступа и т. д.

Из всех возможных инструкций наиболее часто в файле .htaccess используют инструкции разного рода перенаправлений (Redirect) посетителя между страницами сайта. Например, безусловное перенаправление любого посетителя на защищенный протокол https в файле htaccess выглядит так:

**RewriteEngine On**

**RewriteCond %{HTTPS} !on**

**RewriteRule (.\*) https://%{HTTP\_HOST}%{REQUEST\_URI}**

Первая строка включает работу редиректов. Вторая — проверка условия - «Если протокол не https». Третья выполняет правило - «Для любых запросов (задается регулярным выражением - (.\*)) перенаправить по протоколу https на хост HTTP\_HOST содержимое запроса REQUEST\_URI» Как видно в работе редиректов используется синтаксис регулярных выражений.

Использование файла .htacces – многообразно. Документацию по возможностям файла .htaccess находится по адресу: <http://www.htaccess.net.ru/>.

## Настройка PHP

Обработка файлов PHP запускается во всех конфигурациях серверов OpenServer, кроме “чистого” Nginx (т.к. запуск интерпретатора php выполняет сервер Apache). Какая версия PHP запускается можно посмотреть в окне настроек OpenServer в блоке “PHP” см. рис. 3. Также информацию о версиях и натройках PHP, Apache и БД MySQL можно получить, вызвав страницу с кодом Листинга 1, где функция PHP - phpinfo() выводит соответствующую информацию см. Рис.4.

Конфигурационный файл PHP находится по адресу **C:\OpenServer\OSPanel\modules\php\***имя\_версии\_PHP***\php.ini**. При классическом запуске PHP под Apache для настройки PHP нужно редактировать именно его, а затем перезапустить сервер Apache. В OpenServer этот конфигурационный файл, так-же как и конфигурационный файл web-сервера перезаписывается при каждом запуске/перезапуске OpenServer на основе шаблона конфигурационного файла по адресу **C:\OpenServer\OSPanel\userdata\config\***имя\_версии\_PHP***\_php.ini** и значений настроек из окна настроек OpenServer.

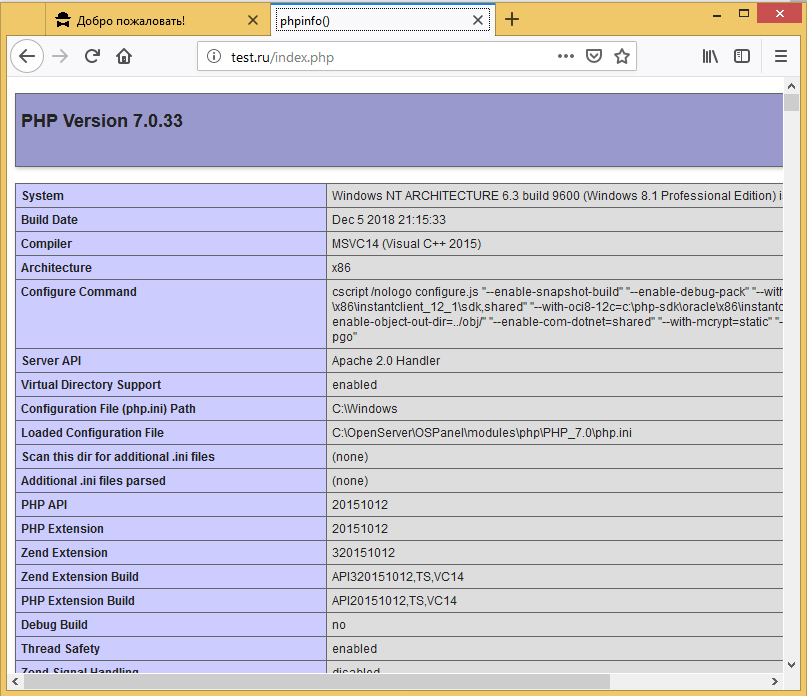


Рис. 4. Вид вывод информации о настройках сайта с помощью функции phpinfo()

Функциональность языка PHP расширяется с помощью библиотек. Эти библиотеки могут предоставлять дополнительные возможности по графике, работе с базами данных, работе с почтой, ускорения работы сценариев PHP и т.д. Файлы библиотек размещаются в папке **C:\OpenServer\OSPanel\modules\php\***имя\_версии\_PHP***\ext\** и подключаются в блоке **Dynamic Extensions** конфигурационного файла php.ini. Например, за работу PHP с базой данных (под Windows) отвечает библиотека расширения - **php\_mysql.dll**, которая подключается в конфигурационном файле строкой: **extension=php\_mysql.dll**. В случае OpenServer в окне настроек нет возможности выбирать модули для PHP. Поэтому такой выбор необходимо делать с помощью правки конфигурационного файла **C:\OpenServer\OSPanel\userdata\config\***имя\_версии\_PHP***\_php.ini**. Если такая необходимость возникла рекомендуется перед редактированием создать резервную копию конфигурационного файла.

Кодировка, в которой работает интерпретатор PHP задается директивой **mbstring.internal\_encoding = UTF-8**.

## Настройка базы данных MySQL

Взаимодействие с БД МуSQL по умолчанию осуществляется по порту 3306. В OpenServer возможно подключение различных версий БД MySQL, но одновременно работать можно только с одной версией. Выбор версии БД MySQL осуществляется на вкладке “Модули” окна настроек OpenServer. Рабочий конфигурационный файл выбранной БД MySQL находится по адресу - **C:\OpenServer\OSPanel\modules\database\***имя\_версии\_бд***\my.ini**. Этот файл так-же временный, перезаписываемый OpenServer при каждой перезагрузке на основании шаблона конфигурации данной версии БД, лежащего по адресу - **C:\OpenServer\OSPanel\userdata\config\***имя\_версии\_бд***\_my.ini** и значений конфигурационных переменных, выбранных в окне настроек OpenServer.

Администрировать СУБД MySQL можно с помощью командной строки, но, конечно, удобнее это делать с помощью визуальных редакторов, например, с помощью программы PhpMyAdmin, которая устанавливается вместе с OpenServer и доступна из панели меню управления OpenServer в закладке “Дополнительно”. По умолчанию, имя админа БД root, а пароль — пустой. Изменить пароли можно во вкладке “Учетные записи пользователей” или Users (для англоязычной установки) см. рис. 5.

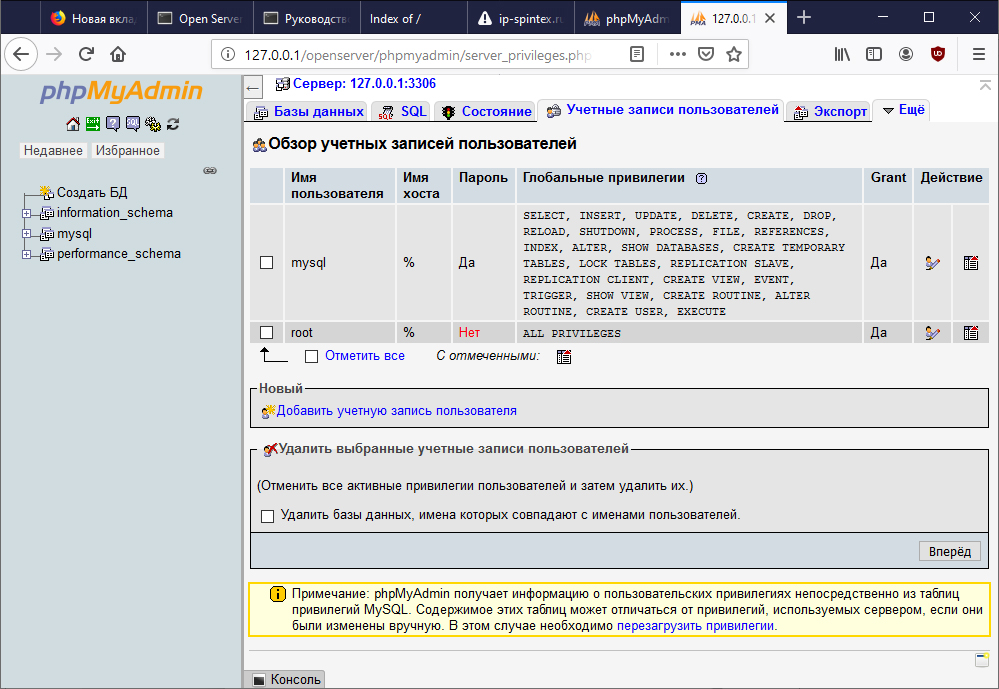


Рис. 5. Страница настройки учетных данных пользователей в PhpMyAdmin

Кодировка по умолчанию устанавливается в блоке “MySQL сервер” закладки “Кодировки” окна настроек OpenServer, см. рис. 6. в параграфе “Настройка кодировки”. Кроме этого, кодировку по умолчанию можно установить (и потом поменять) для каждой вновь создаваемой БД и для каждой таблицы в БД c помощью PhpMyAdmin.

## Настройка кодировки

Проблема кодировки в том, что она может быть установлена во многих местах:

1. Непосредственно кодировка файла (страницы)  
   Т.е. кодировка, в которой файл создан в файловой системе с помощью некоего редактора, например NotePad или NetBeans.
2. Кодировка в интерпретаторе  
   Для PHP это установка mbstring.internal\_encoding = UTF-8 в файле php.ini
3. Кодировка в коде файла с помощью атрибута charset  
   Пример: <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
4. Кодировка в конфигурационном файле сайта.  
   Например, для сервера Apache это запись AddDefaultCharset UTF8 в файле htaccess в корневой папке сайта
5. Кодировка базы данных сайта.  
   Например, для СуБД MySQL это запись character-set-server = utf8 в файле my.ini. Это кодировка по умолчанию. Для разных баз данных в рамках одной СУБД может быть разная кодировка таблиц.
6. Кодировка в настройках web-сервера.  
   Например, для сервера Apache это запись AddDefaultCharset UTF8 в файле httpdp.conf в папке conf корневой папки сервера.

Особенно чревата сюрпризами кодировка в настройках web-сервера. Этого достаточно, чтобы даже в случае всех остальных верных настроек кодировки русские буквы отражались на сайте «абракадаброй».

До выходя спецификаций UTF-8 и UTF-16, для русского языка использовалась кодировка win-1251 (windows-1251). Сейчас необходимо везде использовать кодировку UTF-8. Для установки в явном виде кодироки UTF-8 для сервера Apache, запускаемого с помощью OpenServer, необходимо в окне настроек OpenServer выбрать вкладку “Кодировки”, затем в выпадающем меню внутри блока “HTTP сервер” выбрать utf-8, см рис. 6.

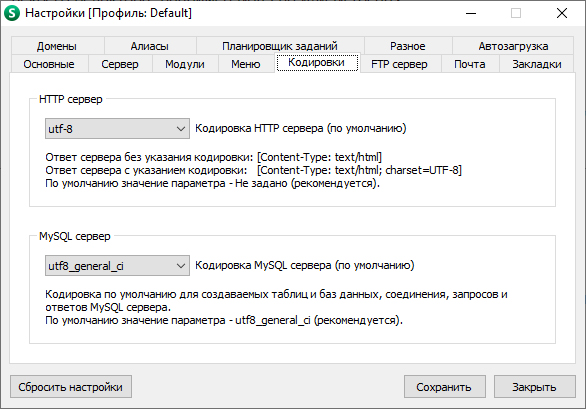


Рис. 6. Настройка кодировки в OpenServer

## Установка и настройка среды NetBeans для работы с PHP

Запустите на установку среду NetBeans с рабочего стола.

После установки можно создать новый PHP проект. Этот проект очевидно должен соответствовать некоторому сайту. Проект создается выбором в главном меню «Файл/Создать проект» и далее в списке типов проектов нужно выбрать PHP и в списке типов проектов PHP выбрать «Приложение PHP с существующим исходным кодом». Затем на второй закладке в поле “Папка источников” нужно выбрать тот сайт, который будет разрабатываться в NetBeans. На этой-же вкладке нужно установить кодировку файлов проекта — UTF-8 и версию PHP – 7.0. На третьей вкладке настройки проекта нужно добавить файл индекса. Для сайта test.ru это будет файл index.php.

Если все сделано верно, NetBeans создаст в папке test.ru папку nbproject в которой будет хранить свои служебные файлы. После создания проекта в IDE необходимо настроить два инструмента: подсветку синтаксиса и отладчик.

Для настройки подсветки синтаксиса щелкните правой клавишей мыши на заголовке проекта, а затем на пункте «Свойства» выпадающего меню. В открывшемся модальном окне в первом пункте “Исходные файлы” установите галочку у свойства «Разрешить краткие теги (<?)». Там-же в свойствах «Кодирование» должно стоять значение UTF-8, а в свойстве «Версия PHP» - PHP 7.0

Настройка отладчика более сложна и выполняется как в среде NetBeans так и в конфигурационном файле PHP. В OpenServer нужно отредактировать шаблон конфигурационный файла PHP по адресу **C:\OpenServer\OSPanel\userdata\config\***имя\_версии\_PHP***\_php.ini**., где *имя\_версии\_PHP* — имя версии PHP, указанной на закладке «Модули» в окне настроек OpenServer. Отладчиком в PHP является специальный модуль Xdebug. В конфигурационном файле PHP в блоке [xdebug] нужно раскоментировать (убрать «;» в начале строки) следующие директивы:

Листинг 2. Настройки запуска xdebug

;zend\_extension = php\_xdebug.dll

;xdebug.profiler\_output\_name = "cachegrind.out.%p"

;xdebug.remote\_enable = off *(заменить off на on)*

;xdebug.remote\_handler = "dbgp"

;xdebug.remote\_host = "localhost"

;xdebug.remote\_port = 9000

Имя dll-файла может отличаться для различных версий PHP. Полностью блок [xdebug] приведен в Приложении 2.

Затем в NetBeans нужно вызвать окно параметров из меню по пути: «Сервис/Параметры». В появившемся модальном окне выбрать «PHP/Отладка» и установить параметры, как показано на рис. 7.

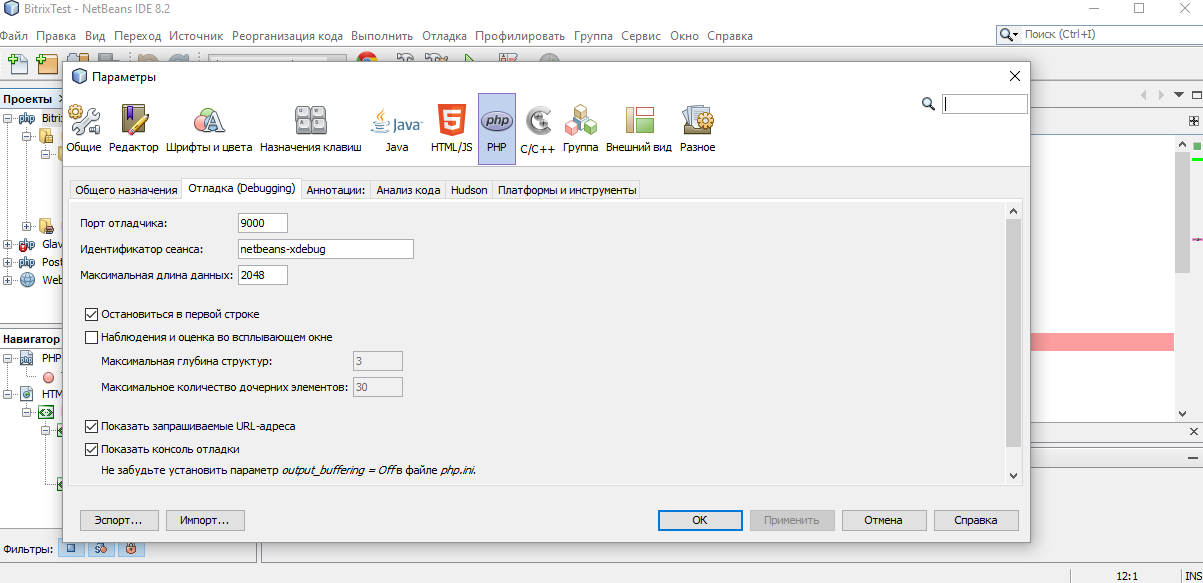


Рис. 7. Настройка отладчика xdebug в NetBeance

Если в закладке «PHP/Общего назначения» не указан путь к интерпретатору PHP нужно ввести его, а также установить галочку в свойстве «Открыть результаты в» в «Веб-браузер» см. рис. 8. Путь к интерпретатору указывается с учетом запускаемой версии PHP – **C:\OpenServer\OSPanel\modules\php\***имя\_версии\_PHP***\php.exe**.

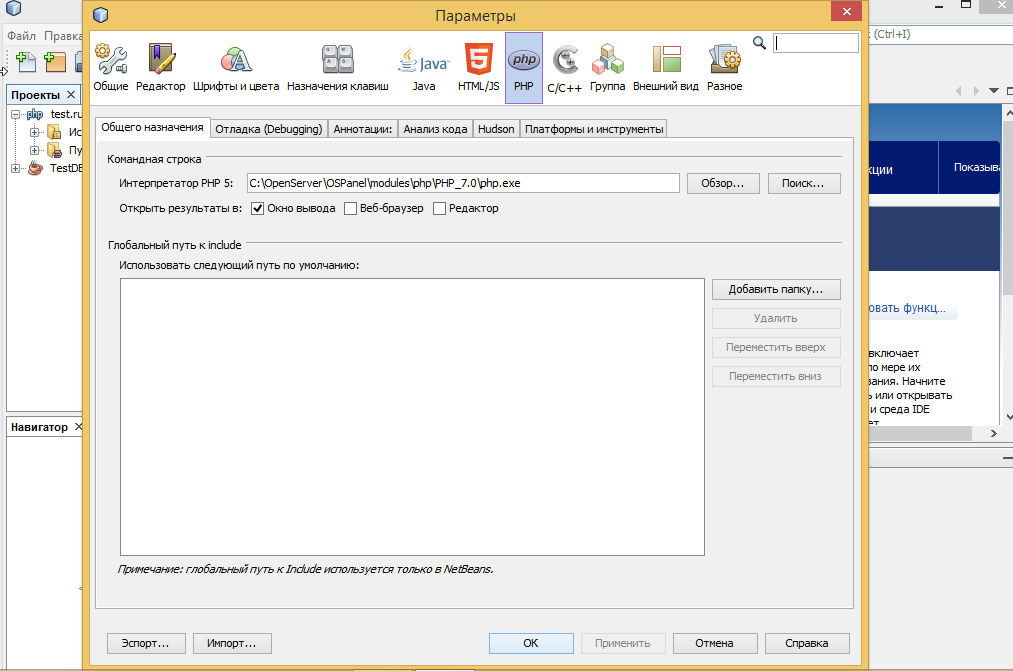


Рис. 8. Настройка отладчика xdebug в NetBeance

Также необходимо настроить проект в NetBeance. Параметры настройки для сайта test1.ru показаны на двух скриншотах ниже:

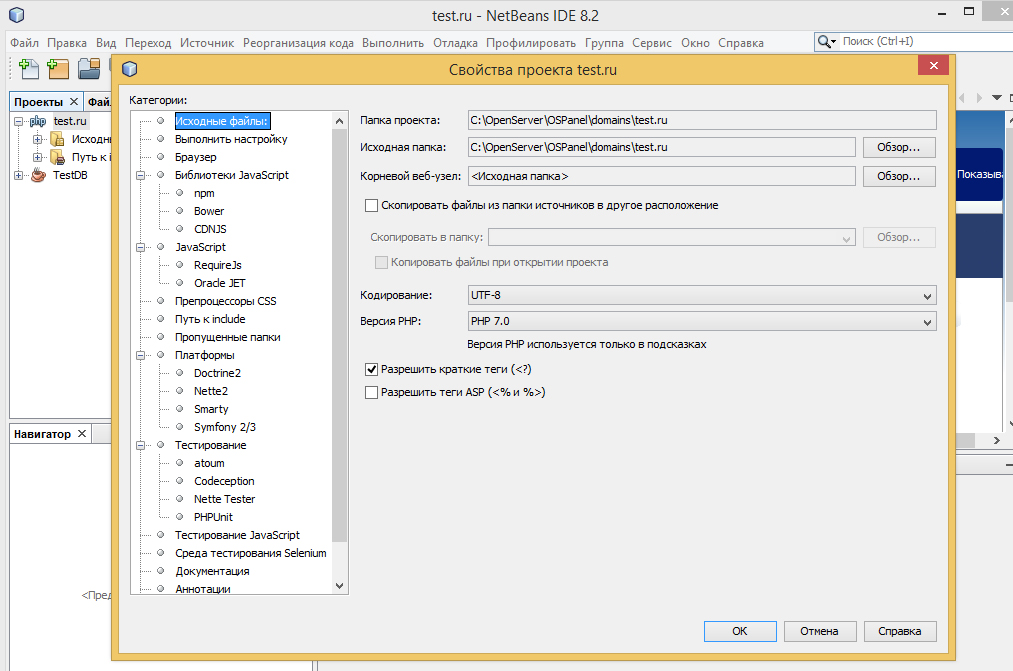


Рис. 9. Настройка папок с исходным кодом для PHP проекта в NetBeance

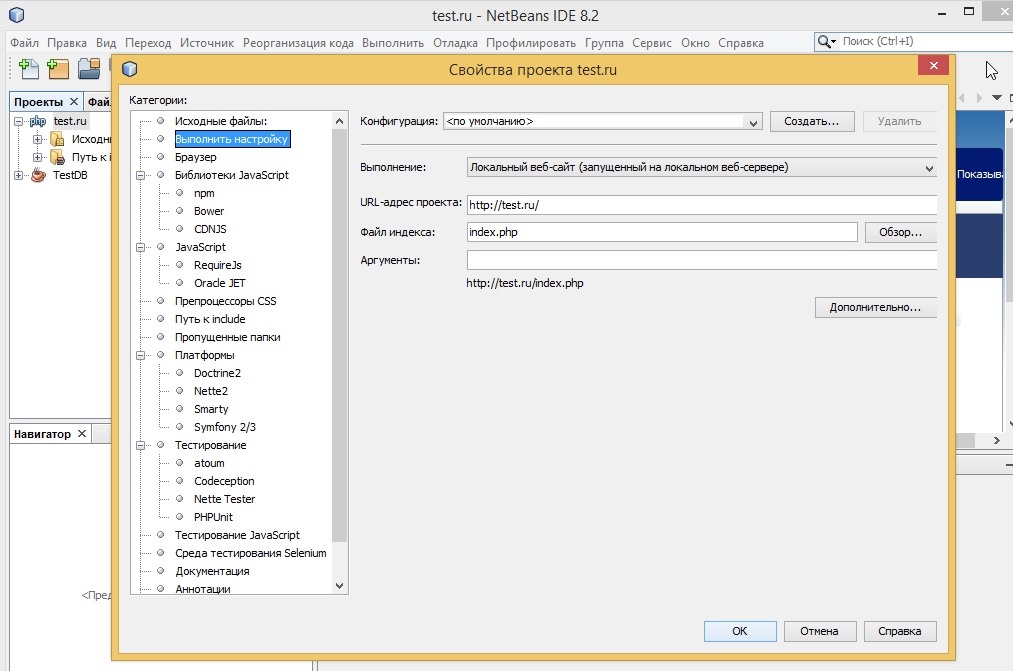


Рис. 10. Настройка виртуального адлеса для PHP проекта в NetBeance

После этого необходимо проверить отладку. Например, в созданном ранее файле index.php установить точку останова на функции phpinfo(); и запустить отладку из меню или комбинацией «Ctrl+F5». NetBeans запустит броузер с параметром в адресе: /index.php?XDEBUG\_SESSION\_START=netbeans-xdebug и остановится на точке останова (если не была оставлена галочка «Останавливаться в первой строке») см. рис. 11. Соответственно появится инструментарий для отладки: Переменные, Наблюдения и т. д.

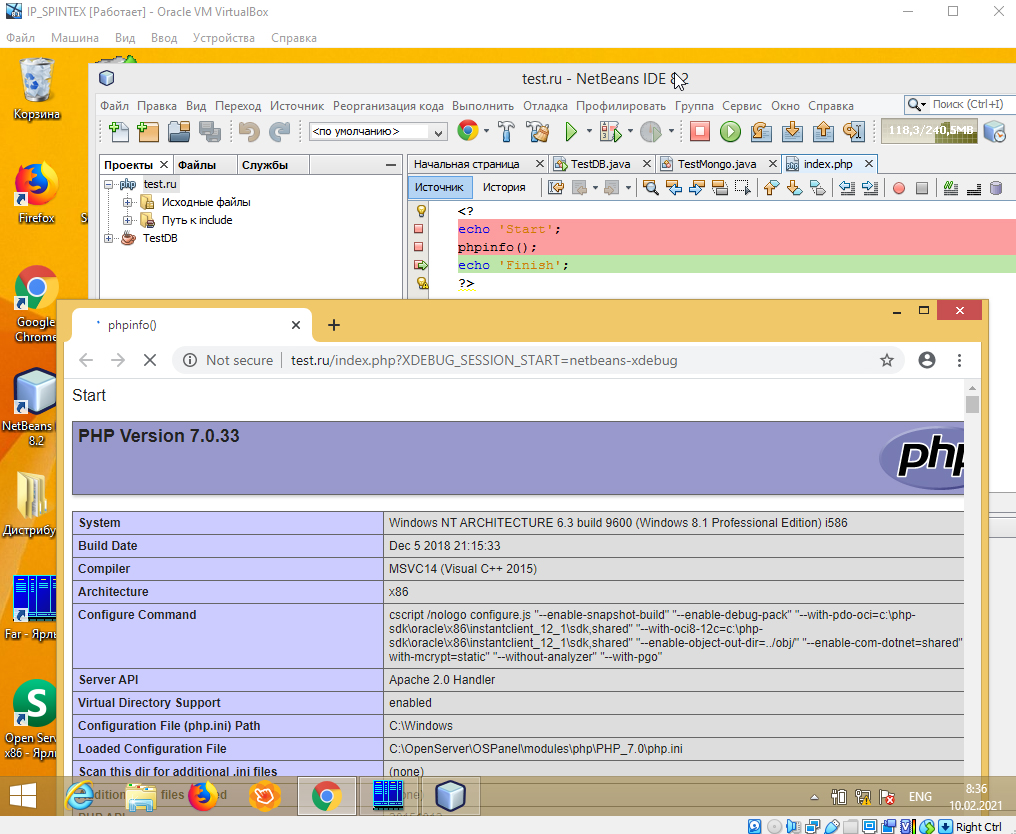


Рис. 11. Выполнение отладки PHP проекта в NetBeance

Также как и сервисы Денвера NetBeans имеет конфигурационный файл по адресу **<NetBeans\_dir>/etc/netbeans.conf**, где <NetBeans\_dir> - папка в которую был установлен NetBeans.

Например, чтобы установить кодировку по умолчанию UTF-8 нужно добавить в строку **netbeans\_default\_options**, дополнительную опцию "**-J-Dfile.encoding=UTF-8**". Опции в строке отделяются друг от друга пробелами

С помощью xdebug можно отлаживать проекты в других IDE, например в PHPStorm, см. статью «Пошаговая настройка Xdebub+OpenServer+PHPStorm» gj flhtce <https://ospanel.io/forum/viewtopic.php?t=1249>.

# **Основы HTML и CSS**

**HTML** (от англ. HyperText Markup Language — «язык гипертекстовой разметки»), применяется для форматирования документов в сети Интернет, отображаемых броузерами. Появление HTML относится к 1993 году. Первый стандарт HTML 2.0 опубликован в 1995 году. HTML 3.\* - январь 1997 г. HTML 4.\* - декабрь 1997. HTML5 – 2014 г.

С 1998 по 2009 год, на замену HTML разрабатывался более «строгий» язык XHTML, однако развития он не получил как раз по причине своей строгости. Наоборот, развитие форматирования документов в HTML пошло в сторону упрощения синтаксиса.

Например, если в спецификации HTML 4, заголовок типа документа мог выглядеть так:

**<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">**

То в HTML 5 теперь во всех случаях достаточно только описания вида:

**<!DOCTYPE html>**

Листинг 3. Пример простого HTML документа

<!DOCTYPE html>  
<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>МИЭТ</title>

</head>

<body>

<center>Здравствуй мир!</center>

</body>

</html>

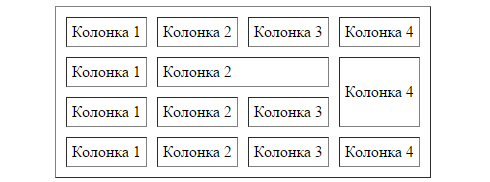
Все содержимое между тегами html делится на два блока заголовок: <title> и тело документа <body>. Все, что описано внутри тега title загружается до начала отображения содержимого документа.

Подробное рассмотрение всей полезной функциональности HTML не входит в тему данной лабораторной работы, поэтому здесь будут только основные моменты достаточные для выполнения задания по многоколоночной верстке.

До появления спецификации HTML 4.\* основным способом создания многоколоночной верстки для Интернет-документов был тег создания таблицы - <table>. Внутри парного тега <table> задавались строки <tr> и ячейки <td> таблицы. Строки и ячейки можно объединять с помощью атрибутов collspan для колонок и rowspan для строк. На рис ниже показан HTML код и соответствующий ему вид страницы.

Листинг 4. Пример оформления таблицы

<table cellpadding=5 cellspacing=10 border=1 align=center>  
  
<tr>  
<td>Колонка 1</td>  
<td>Колонка 2</td>  
<td>Колонка 3</td>  
<td>Колонка 4</td>  
</tr>  
  
<tr>  
<td>Колонка 1</td>  
<td colspan=2>Колонка 2</td>  
<td rowspan=2>Колонка 4</td>  
</tr>  
  
<tr>  
<td>Колонка 1</td>  
<td>Колонка 2</td>  
<td>Колонка 3</td>  
</tr>  
  
<tr>  
<td>Колонка 1</td>  
<td>Колонка 2</td>  
<td>Колонка 3</td>  
<td>Колонка 4</td>  
</tr>  
  
</table>



Как видно в открывающем теге table есть несколько атрибутов, которые задают внешний вид таблицы: cellspacing – задает отступ между ячейками, cellpadding – задает отступ содержимого от контура ячейки, border – устанавливает наличие или отсутствие визуального отображения контуров ячейки и таблицы, align – задает положение таблицы на странице.

Недостаток такого задания вида таблицы, очевидно заключается в том, что один и тот-же принятый вид оформления нужно копировать в каждую таблицу. Кроме того, при необходимости смены оформления таблиц изменения нужно вносить в атрибуты всех таблиц.

## CSS — каскадные таблицы стилей

Чтобы уйти от данных неудобств в оформлении были принят формат внешнего описания вида документа в каскадных таблицах стилей — CSS (Cascading Style Sheets). Первая редакция CSS 1 была принята еще в 1996 году, однако она не позволяла осуществлять блочной верстки и сайты верстались таблицами. Вторая версия CSS 2 появилась в 1998 году и позволяла блочную верстку, однако производители броузеров не сразу реализовали ее. Следующая итерация CSS 3 появилась только в 2011 г но до сих пор производители броузеров реализовали ее не в полном объеме.

Проверить поддержку какого либо тега HTML или инструкции CSS можно, например, по адресу htmlbook.ru. Как видно, например, тег <main> спецификации HTML5 поддерживается только 2-мя броузерами из 7 — см. <http://htmlbook.ru/html/main>. Описание и поддержку броузерами правил CSS для разных версий можно посмотреть по адресу - <http://htmlbook.ru/css/>.

Стили CSS могут быть внедрены на страницу 3-мя способами:

1. непосредственно в тег, например так:

<table style=”text-align:center; width:100px;”>

1. с помощью блока <style>, например так:

<style>  
table {

text-align:center;

width:100px;

}

1. С помощью подключения файла стилей, например так:  
   <link href="[/themes/hb/css/style.css](http://htmlbook.ru/themes/hb/css/style.css)" rel="stylesheet">

Стили применяются с помощью селекторов. Селектором может быть сам тег, например table, а также селектор может быть задан для класса или уникального идентификатора:

**Задание для класса**

<style>

table.first { width:100%; }

</style>

<table class=”first”>

…

</table>

**Задание для идентификатора**

<style>

#first { width:100%; }

</style>

<table id=”first”>

…

</table>

При совпадении стилей c одними именами селекторов, но разным содержанием, применяется тот стиль, селектор которого был определен последним.

Для блочной верстки используются теги <div> и <span>. Это теги — контейнеры для включения в них прочего содержимого страницы. Тег div изначально блочный т. е. По умолчанию занимает 100% доступной ширины и любой следующий за ним контент показывается под ним. Тег <span> является «строчным», т. е. Занимает столько места, сколько контента есть в нем, а следующий за ним контент следует за ним на той-же строке.

## **Многоколоночная верстка макета страницы**

Наибольшее распространение получили трехколоночные макеты. Ширина (width) макета может быть задана в % от доступной ширины (экрана), в пикселях (px) и в стиле - «занимать все доступное пространство», обозначаемом значком бесконечности - ∞.

Свойство float со значениями: right или left задает сторону экрана относительно которой выстраивается последовательность элементов, имеющих совйства left или right.

Рассмотрим построение трехколоночного макета правая и левая колонка которого фиксированной ширины, а центр является «резиновым» т.е. - ∞.

Листинг 5. Вид макета и стили

<!DOCTYPE html>

<html >

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>Макет 3 колонки</title>

<style type="text/css">

.header, .footer { background: #a7a7a7; }

.layout { overflow: hidden; min-width:600px; }

.leftCol { background: #ff0000; float: left; width: 180px; }

.centerCol { background: #00ff00; margin: 0 180px 0 180px; }

.rightCol { background: #0000ff; width: 180px; float: right; }

</style>

</head>

<body>

<div class="header">Шапка сайта</div>

<div class="layout">

<div class="leftCol">Колонка 1</div>

<div class="rightCol">Колонка 3</div>

<div class="centerCol">Колонка 2</div>

</div>

<div class="footer">Подвал</div>

</body>

</html>

Рис. 12. Результат выполнения Листинга 5 в браузере.

## Адаптивная верстка

Адаптивность означает изменение видимой разметки (дизайна) страницы при изменении размеров окна просмотра (viewport).

Для применния адаптивности используют CSS правило @media, имеющее следующий синтаксис:  
  
@media *тип\_устройства* and (CSS *условие для устройства*) {

/\* *стили, применяемые для устройства при выполнении условия* \*/

}

Пример

@media screen and (max-width: 767px) {

p { margin:10px; }

}

Код в примере применяет внешний отступ в 10px для параграфов при достижении видимой областью размера 767px и меньше.

Основные используемые значения типов устройств: screen - экран, print - принтер, all — все.

CSS условия могут использовать все инструкции CSS-стилей, определенные в CSS но для отслеживания изменения размеров окна просмотра, используют стили задающие ширину или высоту: max-width, max-height.

## Фреймворк разметки Bootstrap

Создание макета страницы и оформление элементов сайта является трудоемким процессом. Для его облегчения разработаны специализированные фреймворки (библиотеки). Bootstrap (<https://getbootstrap.com/>) является одним из наиболее распространенных фреймворков разметки и стилевого оформления сайтов. Дата создания Bootstrap – 2011 год.

Для подключения Bootstrap необходимо скачать архив дистрибутив с официального (<https://getbootstrap.com/docs/4.4/getting-started/download/>) или русскоязычного (<https://bootstrap-4.ru/docs/4.4/getting-started/download/>) сайта Bootstrap и распаковать в выбранной папке сайта, а затем подключить на странице(ах) сайта, head, например так:

Листинг 6. Подключение Bootstrap

<head>

<link rel="stylesheet" href="/bootstrap/css/bootstrap.css">

<script src="/bootstrap/js/jquery/jquery-3.4.1.js"></script>

<script src="/bootstrap/js/popper/popper.js"></script>

<script src="/bootstrap/js/bootstrap/bootstrap.js"></script>

</head>

Bootstrap имеет встроенный механизм разметки сетки страницы, который включает 12 колонок, которые можно комбинировать объединяя. Документация по разметке с помощью Bootstrap доступна по адресу - <https://bootstrap-4.ru/docs/4.4/layout/grid/>.

Внешнему контейнеру для колонок присваивается класс .container для установки ступенчато (100%, 576px, 768px, 992px, 1200px ) фиксированной ширины или .container-fluid для 100%-ной ширины для всех разрешений.

Ширина колонок устанавливается классом в виде: col-код разрешения-номер. Ниже показан код описания двух-колоночного макета шириной в 8 и 4 колонки. Существует возможность изменять компоновку колонок и их порядок в зависимости от разрешения экрана. В примере ниже для разрешения экрана1200px и больше, левая колонка включает 4 колонки, правая 8. Для разрешения 576-1200px левая колонка в ключает 4, а правая — 8. При разрешении меньше 576px ширина каждой колонки 100% от ширины экрана, причем правая расположена над левой.

Листинг 7. Код контейнеров.

<div class="container-fluid">

<div class="row">

<div class="col-xl-4 col-sm-8 order-sm-1 order-2">Левая</div>

<div class="col-xl-8 col-sm-4 order-sm-2 order-1">Правая</div>

</div>

</div>

Таблица 1. Имена CSS-классов Bootstrap, применяемых в заданных диапазонах

|  |  |
| --- | --- |
| Класс | Диапазон применения в px |
| .col- | <576px |
| .col-sm- | 576<768 |
| .col-md- | 768<992 |
| .col-lg- | 992<1200 |
| .col-lx- | >1200 |

По умолчанию внутренние отступы колонок равны 15 px. Но иногда требуется отключить эти отступы для бесшовного отображения колонок. Эти отступы можно удалить с помощью класса .no-gutters, который удаляет отрицательный margin из .row и горизонтальный padding из всех колонок.

## Инструмент***ы*** разметки Flexbox Layout и Grid Layout

Развитие и популярность фрейворков разметки привели к включению в поддерживаемые браузерами инструметы CSS-разметки новых модулей, в той или иной мере перекрывающий инструментарий фреймворков, основанных на более ранних версиях CSS-стилей. Примерами таких модулей являются модули Flexbox Layout и Grid Layout. Данные модули появились в CSS3, как рекомендация W3C сравнительно давно - в 2015 году (<https://www.w3.org/TR/css-template-3/>), однако полноценная поддержка этих модулей во всех значимых браузерах была реализована на 3-4 года позже, в 2018-2019 г.

Документация:

1. Flexbox Layout - <https://html5book.ru/css3-flexbox/>
2. Grid Layout - <https://html5book.ru/css-grid/>

Листинг 8. Пример разметки трехколоночного макета с использованием Flexbox

<html>

<head>

<style>

#main\_cont {

display:flex;

flex-direction: row;

background-color: #ededed;

padding: 20px;

height: auto;

min-width: 700px;

}

#main\_cont div {

margin:10px;

background-color: #ed0000;

border-radius: 10px;

padding: 10px;

flex: 1 1 100px;

}

div#f1 {

flex: 0 0 180px;

}

div#f3 {

flex: 0 0 180px;

}

div#f2 {

min-width: 200px;

}

</style>

</head>

<body>

FlexBox

<hr>

<div id="main\_cont">

<div id="f1">Flex 1</div>

<div id="f2">Flex 2<br>a<br>a<br>a<br></div>

<div id="f3">Flex 3</div>

</div>

</body>

</html>

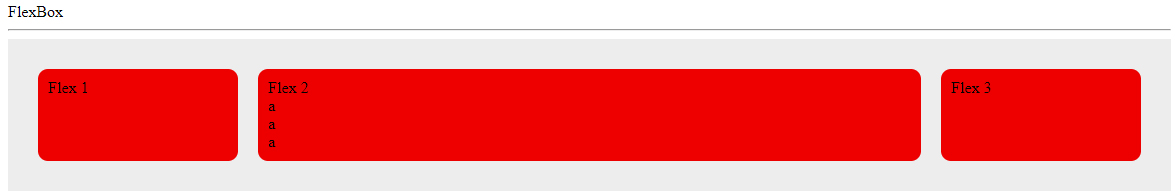


Рис. 13. Результат выполнения кода из Листинга 8

## Встаивание шрифтов

Для отображения текста на странице художественными шрифтами их можно загрузить с ресурса <https://fonts.google.com/>. Выбранный шрифт можно всторить на страницу с помощью ссылки вида: <link href="https://fonts.googleapis.com/css?family=**Oswald**&display=swap" rel="stylesheet">

Если перейти по ссылке, то можно увидеть, что шрифты встраиваются в страницу конструкцией вида:

@font-face {

font-family: 'Oswald';

font-style: normal;

font-weight: 400;

font-display: swap;

src: url(https://fonts.gstatic.com/s/oswald/v30/TK3\_WkUHHAIjg75cFRf3bXL8LICs1\_FvsUtiZTaR.woff2) format('woff2');

unicode-range: U+0460-052F, U+1C80-1C88, U+20B4, U+2DE0-2DFF, U+A640-A69F, U+FE2E-FE2F;

}

Это позволяет скачать данные шрифты и установить их локально.

**Задание на лабораторную работу**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Задание | Баллы | % |
| 1 | Установить и настроить рабочее место разработчика для сайта **№студбилета.ru**. Проверить работу отладчика. | 1 | 20 |
| 2 | Сделать трехколоночный макет по варианту из таблицы «Варианты заданий».  С помощью адаптивной верстки преобразовать макет в одноколоночный при ширине экрана менее 600px Встроить локально на страницу кириллический шрифт по собственному выбору с <https://fonts.google.com/> | 2 | 40 |
| 3 | С помощью Bootstrap создать трехколоночный макет с количеством «подколонок» в каждой колонке по варианту из таблицы «Варианты заданий». Реализовать обратный порядок колонок (вверху 3-я, потом 2-я, потом 1-я) при ресайзинге окна для ширины менее 576 px | 1 | 20 |
| 4 | Реализовать техколоночный макет пункта 2 с помощью инструментария Flexbox Layout или Grid Layout | 1 | 20 |
|  |  | 5 | 100 |

Варианты заданий.

|  |  |  |
| --- | --- | --- |
|  | Для задания 2 | Для задания 3 |
| 1 |  | 4,2,6 |
| 2 |  | 3,2,7 |
| 3 |  | 5,2,5 |
| 4 |  | 3,6,3 |
| 5 |  | 8,2,2 |
| 6 |  | 2,4,6 |

**Контрольные вопросы:**

1. Что такое web-сервер?
2. Что такое сессия?
3. Что такое индексный файл?
4. Какая кодировка предпочтительна? Почему?
5. Как настроить отладчик PHP?
6. Как уменьшить размер памяти, потребляемой web-сервером?
7. Как уменьшить размер памяти, потребляемой интерпретатором PHP?
8. К каким тегам HTML применимо оформление средствами CSS?
9. Какова последовательность применения стилей CSS?
10. Как отладить программу на PHP с помощью NetBeans?

**Приложение 1**

# Список литературы и ссылок

1. Работа с PHP в NetBeans
   1. <https://netbeans.org/kb/74/php/editorguide_ru.html>
2. Инструменты просмотра отчетов xdebug:
   1. <http://www.simplecoding.org/webgrind-udobnyj-interfejs-dlya-analiza-bystrodejstviya-php-skriptov.html>
3. Справочники по HTML и CSS
   1. [https://html5book.ru](https://html5book.ru/)
   2. [http://htmlbook.ru](http://htmlbook.ru/)
   3. [https://webref.ru](https://webref.ru/)
4. Ресурсы изучения Bootstrap
   1. <https://getbootstrap.com/> (англ.)
   2. <https://bootstrap-4.ru/> (рус.)

**Приложение 2**

# Настройки xdebug в php.ini

[xdebug]

;xdebug.auto\_trace = off

;xdebug.cli\_color = 0

;xdebug.coverage\_enable = on

xdebug.default\_enable = off

;xdebug.extended\_info = 1

;xdebug.file\_link\_format = ""

;xdebug.filename\_format = ""

;xdebug.force\_display\_errors = 0

;xdebug.force\_error\_reporting = 0

;xdebug.halt\_level = 0

;xdebug.idekey = ""

;xdebug.max\_nesting\_level = 256

;xdebug.max\_stack\_frames = -1

;xdebug.overload\_var\_dump = 2

; disable the @ (shut-up) operator

;xdebug.scream = off

;xdebug.collect\_assignments = off

;xdebug.collect\_includes = on

;xdebug.collect\_params = off

;xdebug.collect\_return = off

;xdebug.collect\_vars = off

;xdebug.dump.COOKIE = ""

;xdebug.dump.ENV = ""

;xdebug.dump.FILES = ""

;xdebug.dump.GET = ""

;xdebug.dump.POST = ""

;xdebug.dump.REQUEST = ""

;xdebug.dump.SERVER = ""

;xdebug.dump.SESSION = ""

;xdebug.dump\_globals = on

;xdebug.dump\_once = on

; dump undefined values from the superglobals

;xdebug.dump\_undefined = off

;xdebug.gc\_stats\_enable = off

xdebug.gc\_stats\_output\_dir = "%sprogdir%/userdata/temp/xdebug"

;xdebug.gc\_stats\_output\_name = "gcstats.%p"

; 1 - single profiler file will be written for multiple requests

;xdebug.profiler\_aggregate = 0

; 1 - append, 0 - overwrite

;xdebug.profiler\_append = 0

;xdebug.profiler\_enable = 0

; XDEBUG\_PROFILE as GET/POST/COOKIE parameter

;xdebug.profiler\_enable\_trigger = 0

; XDEBUG\_PROFILE secret

;xdebug.profiler\_enable\_trigger\_value = ""

xdebug.profiler\_output\_dir = "%sprogdir%/userdata/temp/xdebug"

;xdebug.profiler\_output\_name = "cachegrind.out.%p"

;xdebug.remote\_addr\_header = ""

;xdebug.remote\_autostart = off

;xdebug.remote\_connect\_back = off

;xdebug.remote\_cookie\_expire\_time = 3600

;xdebug.remote\_enable = off

;xdebug.remote\_handler = "dbgp"

;xdebug.remote\_host = "localhost"

;xdebug.remote\_log = ""

; req - connect on every script starts, jit - connect only on error

;xdebug.remote\_mode = "req"

;xdebug.remote\_port = 9000

;xdebug.remote\_timeout = 200

;xdebug.show\_error\_trace = 0

;xdebug.show\_exception\_trace = 0

;xdebug.show\_local\_vars = 0

;xdebug.show\_mem\_delta = 0

; XDEBUG\_PROFILE as GET/POST/COOKIE parameter

;xdebug.trace\_enable\_trigger = off

; XDEBUG\_PROFILE secret

;xdebug.trace\_enable\_trigger\_value = ""

; 0 - human readable, 1 - computer readable, 2 - HTML

;xdebug.trace\_format = 0

; 1 - append, 0 - overwrite

;xdebug.trace\_options = 0

xdebug.trace\_output\_dir = "%sprogdir%/userdata/temp/xdebug"

;xdebug.trace\_output\_name = "trace.%c"

;xdebug.var\_display\_max\_children = 128

;xdebug.var\_display\_max\_data = 512

;xdebug.var\_display\_max\_depth = 3