Оглавление

Общая информация о курсе

Аннотация

Web-программирование и в первую очередь организация обмена данными по протоколу HTTP - одна из наиболее распространенных задач на современном рынке разработки ПО. Кроме огромного сегмента разработки сайтов web-программирование активно используется в мобильных приложениях и больших enterprise-решениях. На сегодняшний день существует огромное разнообразие инструментов для решения этой задачи, поэтому она идеально подходит в качестве примера для изучения средств конструирования ПО. В курсе рассматриваются базовые принципы проектирования приложений и реализация этих принципов на различных языках программирования в разнообразных web-фреймворках.

Состав курса

Курс состоит из:

- 6 лекций;
- 7 лабораторных работ;
- расчетного задания.

Итоговое испытание: **экзамен.**

Порядок сдачи и защиты лабораторных работ

1. Прочитать задание и теоретический материал. В методических указаниях приведены минимальные теоретические сведения, а также ссылки на документацию. Как правило, выполнение задания без использования соответствующей документации невозможно. Не пренебрегайте приведенными источниками.
2. Выполнить задание лабораторной работы. Результатом работы является работающий проект или несколько проектов, удовлетворяющих заданию соответствующей лабораторной работы.
3. Подготовиться к защите лабораторной работы. Используйте контрольные вопросы. Учитывайте, что цель лабораторных - не обезьянье повторение методических указаний, а знакомство с новыми технологиями. Убедитесь, что вы понимаете логику функционирования разработанного проекта.
4. Составить отчет. Отчет должен содержать титульный лист, задание и код разработанного проекта. Отчет предоставляется в электронном виде на электронный адрес mesdt.course@gmail.com. Разрешены следующие форматы: .rtf, .doc, .docx, .pdf. **Титульный лист должен быть напечатан и принесен на защиту лабораторной работы**.
5. На защите лабораторной работы необходимо продемонстрировать работоспособность проекта. Необходимо иметь при себе электронную версию отчета и распечатанный титульный лист.
6. **После успешной защиты студент должен отправить на почту mesdt.course@gmail.com отсканированный титульный лист с оценкой преподавателя**.

Требования и порядок оценивания лабораторной работы

1. Лабораторная работа, выполненная и защищенная в полном объеме и в срок, оценивается в 80 баллов.
2. Выполнение дополнительных заданий учитывается **только при сдачи лабораторной работы в срок** и может поднять оценку до 100 баллов.
3. Каждая просроченная неделя снижает максимальную оценку на 5 баллов.
4. Баллы могут быть снижены, если лабораторная выполнена не в полном объеме или студент не может ответить на вопросы при защите лабораторной работы.

Сроки лабораторных работ

- Лабораторная работа 1: **14 февраля 2015 года**
- Лабораторная работа 2: **21 февраля 2015 года**
- Лабораторная работа 3: **7 марта 2015 года**
- Лабораторная работа 4: **21 марта 2015 года**
- Лабораторная работа 5: **28 марта 2015 года**
- Лабораторная работа 6: **4 апреля 2015 года**
- Лабораторная работа 7: **25 апреля 2015 года**

Правила сдачи и защиты расчетного задания

Требования к расчетному заданию подробно описаны в методических указаниях по расчетному заданию. Требования по оформлению отчета - как для лабораторных работ. Срок выполнения расчетного задания: **1 мая 2015 года**.

Правила сдачи экзамена

Экзамен проводится на сессии. Автомат могут получить все желающие, прошедшие все контрольные точки и имеющие семестровый рейтинг *меньше* 75 баллов. **Для получения оценки "отлично" необходимо сдавать экзамен.**

Экзамен сдается в письменной форме. Билет состоит из двух теоретических вопросов и задачи.

Варианты заданий к лабораторным работам (кроме Mongo)

1. Свой вариант. Согласовать с преподавателем.
2. Сайт строительной компании. Компания занимается строительством коттеджей. На сайте представлен список типовых проектов домов. Для каждого проекта известен основной материал, площадь, этажность.
3. База данных редких растений Алтайского края. Для каждого растения хранится описание, исследователи, заповедники, статус.
4. База горных минералов и руд. Для минерала хранится описание, территории, на которых минерал распространен, месторождения, в состав каких руд входит, основные исследователи и публикации.
5. Каталог одежды. Для одежды хранится категория, размер и цвет. Требуется реализовать возможность заказа.
6. Каталог комнатных растений. Для каждого растения хранятся требования по уходу и на каком окне (с какой освещенностью: южное, северное и т.д.) его можно выращивать. Требуется реализовать возможность заказа.
7. База данных нестандартных экскурсий. Экскурсии по необычным местам городов. Хранится информация о месте экскурсии, экскурсоводе, график проведения нестандартных экскурсий.
8. Каталог туристических палаток. Для палатке указывается, сколько в ней спальных мест, габариты в разложенном и свернутом виде, диапазон температур, в которых она может применяться и другие параметры.
9. Каталог спец. техники. Для каждого автомобиля указывается год выпуска, база и надстройка.
10. Сайт для обмена паззлами. Для каждого паззла хранится его размер и тематика. Для паззла можно добавить фотографию. Дополнительно хранится, все ли детали в наличии.
11. База рецептов <<Всё, что есть в холодильнике>> с возможностью поиска по ингридиентам (например, блюда из остатков сметаны).
12. Каталог изделий мастеров-камнерезов Колыванского завода. Для каждого изделия кроме фотографии и названия имеется список видов камней, которые использовались при создании изделия.

Лабораторная работа 1. Знакомство с инструментарием

Цели работы

- познакомиться с веб-сервером (nginx или Apache);
- познакомиться с шаблоном верстки HTML5 Twitter Bootstrap;
- познакомиться СУБД MySQL и средствами администрирования Adminer и PhpMyAdmin.

Задание

1. Установить виртуальную среду разработки для выполнения лабораторных работ
2. Создать адаптивный html-шаблон для использования в последующих лабораторных работах.
3. Изучить инструменты администрирования для MySQL и создать базу данных для последующих лабораторных работ.

Теоретическая информация

HTTP-сервера

Разработка web-приложений основывается на клиент-серверной архитектуре. В качетве клиентского приложения выступает браузер, а в качестве сервера - либо специально разработанное ПО, либо (более распространенная ситуация) HTTP-сервер.

HTTP-сервер - это программа, которая обрабатывает GET и POST запросы по протоколу HTTP. Два наиболее распространенных HTTP-сервера - это Apache и Nginx. Оба сервера могут обрабатывать запросы к статическим ресурсам (страницам в формате html, файлам css, js и картинкам), а также имеют подключаемые модули, которые позволяют использовать скриптовые языки (такие как PHP, Pyton, Ruby и т.д.) и генерировать динамические html-страницы "на лету" в зависимости от запроса пользователя.

В большинстве случаев современные web-сервера - это не отдельные компьютеры, а виртуальные машины, предоставляемые на условия IaaS или PaaS. Зачастую настройки на реальном сервере существенно отличаются от настроек на локальной машине. Поэтому для выполнения лабораторных работ мы будем использовать виртуальную машину с настроенными серверами на основе Virtual Box и Vagrant.

Немного о доступе к серверу

Для доступа к веб-серверу используются, как правило, протоколы ftp (file transport protocol) и ssh. SSH - сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений (например, для передачи файлов). Чтобы использовать ssh, необходимо установить специальную программу - ssh-клиент.

Для лабораторных работ рекомендуется использовать клиент, который идет в комплекте системы контроля версий Git. Это самая популярная система контроля версий для проектов с открытым кодом.

Адаптивная верстка

Наиболее распространенным способом взаимодействия с пользователем в web являются html-страницы. Развитие мобильных устройств привело к тому, что от современных интернет-страниц требуется корректное отображение не только в разных браузерах, но и в очень большом диапазоне расширений экрана. Верстка, удовлетворяющая этому требованию, называется **отзывчивой**. Отзывчивая верстка - это компонент адаптивной, различия между этими понятиями вы можете найти в Википедии (см. список литературы). Такая верстка требует тщательного тестирования и учета множества нюансов. Производство сайтов - огромная индустрия, и создавать сложную адаптивную верстку с "нуля" невыгодно. В качестве основы для будущих интернет-страниц используются специальные html-фреймворки. Один из них - **Twitter Bootstrap**. Этот фреймворк состоит из одного css-файла (стили страницы) и нескольких js-файлов (скрипты) и обладает широкими возможностями: качественная адаптивная верстка на основе сетки (попробуйте изменить размер окна браузера на официальном сайте фреймворка), большое количество разнообразных элементов (формы, панели навигации, слайдеры, маркеры, всплывающие окна и многое другое). При создании страницы на основе этого фреймворка создается html-код (широкий спектр примеров находится на сайте фреймворка) и дополнительный файл css, который позволяет настроить верстку под конкретный дизайнерский макет. Примеры сайтов можно посмотреть здесь: http://expo.getbootstrap.com/.

СУБД MySQL

MySQL - свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией.

Существует множество инструментов для администрирования MySQL, как платных, так и бесплатных. Наиболее распространенный среди них - **phpMyAdmin**, бесплатное php-приложение, которое, как правило, установлено на серверах интернет-провайдеров. Аналогом является приложение **Adminer.**

MySQL поддерживает несколько типов таблиц, что позволяет гибко настраивать базу в зависимости от соотношений операций чтения-записи, необходимого уровня скорости работы и надежности.

Рассмотрим два наиболее используемых типа таблиц. Другие типы таблиц вы можете изучить в документации (см. ссылку в списке литературы).

MyISAM

Данный тип таблиц предназначен для быстрого доступа к информации. Исторически - один из ранних типов таблиц в MySQL. Многие недостатки не позволяют использовать этот тип таблиц в больших проектах, но для блога, сайта-визитки и других маленьких и средних проектов, не требующих большого количества операций записи и не хранящих критическую информацию, данный вариант наиболее эффективен.

- транзакций нет
- макс. размер на диске: 256Тб
- блокировка: вся таблица
- возможность полнотекстового индекса и быстрого поиска
- отсутствует поддержка целостности и внешних ключей
- поддержка репликации
- каждый столбец может иметь свою кодировку
- медленные операции записи (не подходит для высоконагруженных систем)
- невысокая надежность (не используйте для финансовых операций и делайте бэкапы!)

InnoDB

Один из наиболее быстрых и надежных современных движков для таблиц (не только в MySQL). Операции чтения работают значительно медленнее, чем в MyISAM, особенно если отсутствуют индексы.

- полная поддержка транзакций (4 уровня изоляции)
- макс. диск: 64Тб
- блокировка записи (не таблицы), два вида блокировок (SHARED, EXCLUSIVE)
- полнотекстовый индекс: нет
- индексы кластеризуются для «типичных запросов»
- поддержка целостности (внешние ключи)

Инструменты

- Ваш любимый браузер (Google Chrome или Mozilla Firefox последней версии, не умничайте про IE).
- Virtual Box, Vagrant, ssh-клиент и виртуальная машина для выполнения лабораторных работ.
- СУБД MySQL в составе набора серверов, инструменты Adminer и PHP MyAdmin.
- Twitter Bootstrap.

Порядок выполнения

Установка виртуальной машины mesdt

1. Установить Virtual Box, если он еще не установлен. Проверить в настройках BIOS, разрешена ли виртуализация.
2. Установить Vagrant, если он еще не установлен. Для пользователей Windows: создайте переменную среды VAGRANT_HOME и пропишите в ней адрес пустого каталога, где Vagrant будет хранить свое глобальное состояние и образы виртуальных машин. Название должно быть латинскими буквами. Название папки для виртуальных машин тоже должно быть латинскими буквами.
3. Для работы с консолью сервера установите ssh-клиент. Мы рекомендуем Git ssh. Чтобы на Windows заработала команда `vagrant ssh`, добавьте путь до ssh-клиента в переменную среды Path:
   `PATH=%PATH%;C:\Program Files (x86)\Git\bin`
4. Скачать и распаковать образ виртуальной машины.
5. Выполнить следующие команды:

```
6. vagrant box add --name mesdt mesdt.box
7. vagrant init mesdt # команду выполнять в новой пустой директории
```

8. После выполнения этих команд в папке появится файл `Vagrantfile`. Настройте папку виртуальной машины на каталог вашей системе, который будет вам удобен, раскомментировав и исправив строчку
9. `config.vm.synced_folder "c:/vagrant/data", "/home/q/hws"`

Здесь "/home/q/hws" - папка на виртуальной машине для выполнения лабораторных работ, "c:/vagrant/data" - любая пустая папка на вашем компьютере. 1. Запустите виртуальную машину:

```
vagrant up
vagrant ssh
```

1. В консоли сервера выполните команду `ifconfig` и узнайте ip-адрес сервера.
2. Для Windows-пользователей. Зайдите в папку `C:\Windows\System32\drivers\etc` и скопируйте оттуда файл hosts. Как правило, даже с администраторскими правами отредактировать его не удается. Поэтому надо скопировать в другую папку, отредактировать и вернуть на место. Добавьте в этот файл следующие строчки, заменив `172.28.128.3` на ваш адрес сервера:

```
# Лабораторные работы
172.28.128.3    0.mesdt
172.28.128.3    1.mesdt
172.28.128.3    2.mesdt
172.28.128.3    3.mesdt
172.28.128.3    4.mesdt
172.28.128.3    5.mesdt
 172.28.128.3    6.mesdt
 172.28.128.3    7.mesdt
 # Клон http://getbootstrap.com/
 172.28.128.3    twbs.mesdt
 # Администраторы базы mysql
 172.28.128.3    adminer.mesdt
 172.28.128.3    phpmyadmin.mesdt
```

17. Проверьте, что всё заработало, набрав адреса в браузере.

Если по каким-то причинам у вас не получается, воспользуйтесь страничкой помощи, или воспользуйтесь альтернативным (не рекомендуется!) вариантом: поставьте XAMPP.

Адаптивная верстка с использованием Twitter Bootstrap

1. Открыть задание, разработать макеты двух страниц в соответствии с заданием. Реализовать макеты с помощью Twitter Bootstrap. Требования к шаблону:
2. шаблон должен состоять не менее чем из двух различных страниц;
3. одна из страниц должна содержать список объектов в соответствии с заданием (на странице должны быть перечислены объекты и их основные характеристики);
4. вторая страница должна содержать полную информацию об одном объетке: название, характеристики, изображение и др.;
5. страницы должны быть в формате html5, и в кодировке utf8;
6. на страницах должны использоваться навигационные панели;
7. на страницы должны быть элементы с различными фонами.
8. Сложите полученный результат в папку `/0` в каталоге, который вы сделали общей папкой с виртуальной машиной, зайдите в браузере на страницу `0.mesdt` и убедитесь, что всё работает. **Обратите внимание! Вы создаете шаблон, заготовку сайта! Загружать туда данные из базы не надо, это просто статические страницы.**

Разработка схемы базы данных и реализация схемы в MySQL

1. С помощью любого CASE-средства или вручную разработать физическую модель базы данных для MySQL в соответствии с заданием (3-5 таблиц).
2. Создать базу данных на сервере MySQL. Логин `root`, пароль `wut`. База данных должна поддерживать ссылочную целостность и русские символы. При создании базы используйте кодировку utf8.
3. С помощью PhpMyAdmin или Adminer внести на сервер несколько записей.
4. Выполнить экспорт базы данных.

Дополнительно

В качестве дополнительного задания студент может:

- настроить другое доменное имя для сайта ;
- использовать слайдер, модальные окна и другие дополнительные компоненыты в страницах описания объектов.

Порядок защиты работы

При защите лабораторной работы необходимо предоставить:

- шаблон верстки двух страниц в Twitter Bootstrap в соответствии с заданием; страницы должны быть доступны как сайт на виртуальном сервере;
- база данных на сервере MySQL, в соответствии с заданием; необходимо продемонстрировать умение пользоваться инструментом PhpMyAdmin и Adminer;
- отчет, содержащий титульный лист, задание, исходный код вёрстки, скриншоты верстки, структуру базы данных, код скрипта генерации базы данных.

Ссылки на скачивание инструментов

- Virtual Box https://www.virtualbox.org/wiki/Downloads
- Vagrant https://www.vagrantup.com/downloads.html
- Виртуальная машина Mesdt https://drive.google.com/file/d/0Bwl2Kb4AFw4ISFlYdnNHLUdua1k/view?usp=sharing
- Виртуальная машина Mesdt2 (исправлена проблема с правами доступа к каталогу hws) https://drive.google.com/file/d/0Bwl2Kb4AFw4IQll4alZ4cVFtVWc/view?usp=sharing
- Git http://git-scm.com/downloads

Ссылки на документацию

- Nginx http://nginx.org/ru/
- MySQL www.mysql.com
- Twitter Bootstrap http://getbootstrap.com/, копия - twbs.mesdt

Дополнительная литература

- Краткая справка по REST
- Статистика популярности веб-серверов.

- Адаптивный веб-дизайн

Контрольные вопросы

1. Что такое веб-сервер?
2. Что такое REST?
3. Что такое PaaS и IaaS?
4. Что такое адаптивная верстка?
5. MySQL. Особенности, типы таблиц.
6. Что такое репликация?

Cheat list Vagrant

Основные команды

`vagrant box add --name mesdt mesdt.box` - добавление коробки с именем `mesdt` из файла `mesdt.box`

`vagrant init mesdt` - создает в текущем каталоге файл конфигурации `vagrantfile`.

`vagrant box list` - посмотреть список короборк

`vagrant box remove mesdt` - удалить коробку с именем `mesdt`

`vagrant up` - создает виртуальную машину и конфигурирует ее (в частности, добавляет сетевой порт для сервера, пробрасывает ssh-порт и расшаривает папки).

`vagrant halt` - останавливает виртуальную машину и удаляет дополнительные настройки.

`vagrant ssh` - подключение к консоли виртуальной машины по ssh.

Что делать, если не работает

- Зайдите в Virtual Box, выберите созданную виртуальную машину и отключите на ней поддержку usb.
- Если `vagrant up` работает очень медленно, то зайдите в настройки виртуальной машины, удалите все общие папки, сетевой интерфейс eth1 и проброску портов на eth0. Повторите запуск `vagrant up`.

Дополнительно

- На сервере установлен Midnight Commander (запускается из консоли сервера или через ssh командой `mc`).
- Конфигурация виртуальных хостов сервера nginx находится в каталоге `/etc/nginx/sites-available/*.conf`

Cheat list nginx

Сервер nginx конфигурируется специальными файлами. На виртуальной машине они расположены в каталоге `/etc/nginx/sites-available`.

При создании нового виртуального хоста его необходимо сделать разрешенным. Это делается с помощью следующей команды:

```
sudo ln -s /etc/nginx/sites-available/1.mesdt /etc/nginx/sites-
enabled/1.mesdt
```

После этого в каталоге `/etc/nginx/sites-enabled` появится линк на файл `1.mesdt`.

После внесения изменений в конфигурацию, чтобы она вступила в силу, сервер nginx необходимо перезагрузить командой:

```
nginx -s reload
```

Перезапускать остальные сервера и всю виртуальную машину не нужно.

Пример виртуального хоста для Silex на nginx

```
server {
    listen 80;
    server_name 1.mesdt;
    root /home/q/hws/1/web;

    location = / {
        try_files @site @site;
    }

    location / {
        try_files $uri $uri/ @site;
    }

    location ~ \.php$ {
        return 404;
    }

        location @site {
                fastcgi_pass unix:/var/run/php5-fpm.sock;
                include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root/index.php;
        }
}
```

Сслыка на документацию

[Руководство для начинающих]

Благодарности

Данное руководство по настройке nginx на основе инструкций Дядюшенко Александра :)

Установка серверов XAMPP

Ссылки для скачивания

- XAMPP http://www.apachefriends.org/index.html

Краткая справка

Полный пакет содержит:

- Web-сервер Apache с поддержкой SSL;
- СУБД MySQL;
- PHP (интерпретатор);
- Perl (интерпретатор);
- FTP-сервер FileZilla (файловый сервер);
- POP3/SMTP сервер (почтовый сервер. может быть заменен на заглушку sendmail);
- TomCat сервер (сервер запуска java-сервлетов);
- утилиту phpMyAdmin (администратор СУБД MySQL на PHP).

XAMPP свободно распространяется согласно лицензии GNU General Public License и является бесплатным web-сервером, способным обслуживать динамические страницы.

Ядро Apache включает в себя основные функциональные возможности, такие как обработка конфигурационных файлов, протокол HTTP и система загрузки модулей. Ядро (в отличие от модулей) полностью разрабатывается Apache Software Foundation, без участия сторонних программистов. Теоретически, ядро apache может функционировать в чистом виде, без использования модулей. Однако, функциональность такого решения крайне ограничена.

Система конфигурации Apache основана на текстовых конфигурационных файлах. Имеет три условных уровня конфигурации. - Конфигурация сервера (`httpd.conf`). - Конфигурация виртуального хоста (`httpd.conf` с версии 2.2, `extra/httpd-vhosts.conf`). - Конфигурация уровня директории (`.htaccess`).

Apache имеет собственный язык конфигурационных файлов, основанный на блоках директив. Практически все параметры ядра могут быть изменены через конфигурационные файлы. Большая часть модулей имеет собственные параметры. Часть модулей использует в своей работе конфигурационные файлы операционной системы (например \texttt{/etc/passwd} и \texttt{/etc/hosts}). Помимо этого, параметры могут быть заданы через ключи командной строки.

Установка

1. Скачать и распаковать пакет XAMPP. При установки отключайте анивирус.
2. Запустить панель управления xampp-control.exe, стартовать сервер Apache и MySQL. Запускать сервера необходимо с административными правами и в доверенной зоне антивирусного ПО. Сервер Apache занимает порт 80 и 8080, и может вступать в конфликт с другим ПО, например, Skype. В случае возникновения проблем используйте навыки поиска в интернете для решения проблемы перед тем, как обратиться к преподавателю. Если преподаватель за один поисковый запрос может решить вашу проблему, вы будете жестоко затроллены.

3. Сконфигурировать Apache. В xampp изначально закрыты права доступа к ресурсам. Зайдите в файл `xampp\apache\conf\extra\httpd-xampp.conf` и исправьте блок в самом конце файла:

```
<LocationMatch
"^/(?i:(?:xampp|security|licenses|phpmyadmin|webalizer|server-
status|server-info))">
    Require local
    ErrorDocument 403 /error/XAMPP_FORBIDDEN.html.var
</LocationMatch>
```

8. Сконфигурировать хост-файл для Apache. Зайдите в файл `xampp\apache\conf\extra\httpd-vhosts.conf` (здесь хранится конфигурация сайтов, о ней будет ниже) и добавьте конфигурацию для инструментов xampp:

```
<VirtualHost *:80>
     DocumentRoot "C:/WORK/xampp/htdocs"
     ServerName localhost
 </VirtualHost>
```

13. Убедиться, что всё заработало, зайдя в браузере `http://localhost/`. Зайдите по ссылке phpMyAdmin, чтобы убедиться в работе MySQL.
14. Установить уровень отображения ошибок для PHP. Изначально XAMPP сконфигурирован для разработчика, т.е. в браузере будут отображаться не только критические ошибки PHP, но и различного рода предупреждения, например, "Strict Standards: ...", "Notice: ..." и другие. Для изменения уровня отображения ошибок PHP нужно открыть файл `C:/xampp/php/php.ini` в любом редакторе и в нем поставить значение: `error_reporting = E_ALL`. При этом значении будут отображаться только критические ошибки PHP. После внесения изменений в файл конфигурации PHP (`php.ini`) так же нужно перезагружать сервер Apache. В контрольной панели XAMPP напротив "Apache" нажать кнопку "Stop", если сервис уже запущен, дальше после остановки нажать кнопку "Start".

Конфигурирование домена

1. Создать в любом месте на диске, где вам удобнее, папку для нового проекта (для примера будет использоваться название `MyDir`). Создать внутри папки `MyDir` папки `web` для содержимого сайта и `logs` для хранения логов.
2. Создать в папке `web` файлindex.htmс тектсомHello, world!`.
3. Перейти в файл `xampp/apache/conf/extra/httpd-vhosts.conf}, отвечающий за настройку виртуальных хостов. Открыть файл в любом текстовом редакторе и добавить новый хост`mysite.local`(это доменное имя сайта, как`yandex.ru`):

```
<VirtualHost *:80>
    ServerAdmin myaddress@mysite.local
    DocumentRoot "c:/full/path/to/MyDir/web"
    ServerName mysite.local
    ServerAlias www.mysite.local
    ErrorLog "c:/full/path/to/MyDir/logs/error.log"
      CustomLog "c:/full/path/to/MyDir/logs/access.log" combined
   <Directory "c:/full/path/to/MyDir/web">
       AllowOverride All
       Order allow,deny
       Allow from all
         Require all granted
   </Directory>
 </VirtualHost>
```

18. Перезапустить Apache. Добавить в файл `c:/Windows/System32/drivers/etc/hosts` строки:

```
            127.0.0.1     mysite.local
            127.0.0.1     www.mysite.local
```

21. Убедиться, что сайт `mysite.local` заработал.

Лабораторная работа 2. PHP Silex

Цели работы

- Познакомиться с языком PHP.
- Изучить реализацию паттерна MVC в микрофреймворке Silex.
- Изучить принципы шаблонизации на примере Twig.

Задание

Реализовать CRUD для одной из сущностей из лабораторной работы 1 с использованием фреймворка PHP Silex и шаблонов из лабораторной работы 1.

Краткие теоретические сведения

Принцип работы HTTP-сервера

HTTP-сервер занимается тем, что получает http-запросы (request) и отправляет на них ответы (response). Запросы могут быть отправлены несколькими методами. Два основных - это POST и GET.

GET используется для запроса содержимого указанного ресурса. Это самый обычный http-запрос. С помощью метода GET можно также начать какой-либо процесс. В этом случае в тело ответного сообщения следует включить информацию о ходе выполнения процесса.

Клиент может передавать параметры выполнения запроса в URI целевого ресурса после символа «?»:

```
/path/resource?param1=value1&param2=value2
```

POST применяется для передачи пользовательских данных заданному ресурсу. Например, в блогах посетители обычно могут вводить свои комментарии к записям в HTML-форму, после чего они передаются серверу методом POST и он помещает их на страницу. При этом передаваемые данные (в примере с блогами — текст комментария) включаются в тело запроса. Аналогично с помощью метода POST обычно загружаются файлы на сервер.

Что конкретно ответить на запрос, сервер определяет по URL, который прописан в запросе. Это может быть как файл, так и результат работы какой-нибудь программы, например, php-скрипта.

Следует различать пути для сайта (URL) и пути в операционной системе. Через браузер пользователь не может попасть за пределы папки `DOCUMENT_ROOT`. В частности, это означает, что в html-коде и других фронтенд-скриптах нельзя получить доступ к содержимому за пределами этой папки. Однако серверные скрипты работают в файловой системе сервера, и на них это ограничение не распространяется.

Silex

Silex - MVC-микрофреймворк для разработки веб-приложений на языке PHP.

В принципе, для реализации модели и представления могут быть подключены разные библиотеки. В стандартный fat-пакет Silex входит DBAL Doctrine для реализации модели и шаблонизатор Twig для реализации представления.

Рассмотрим пример.

```
require_once __DIR__.'/../vendor/autoload.php';

$app = new Silex\Application();

$app->get('/hello/{name}', function($name) use($app) {
    return 'Hello '.$app->escape($name);
});

$app->run();
```

Это минимально приложение на Silex, которое обрабатывает get-запросы по одному адресу: `'/hello/{name}'`. Такие шаблоны называются маршрутами (routes). `{name}` - это параметр маршрута. Если URL http-запроса соответствует этому шаблону, то будет вызвана соответствующая анонимная функция. Функция контроллера, формирующая http-ответ для определенного маршрута, называется действием (action).

В приложении на Silex сначала создается объект (в нашем примере - `$app`), который представляет собой контейнер компонентов. Далее в него добавляются необходимые компоненты и маршруты с действиями в порядке обработки (то есть если URL соответствует нескольким шаблонам, то будет выполнено то действие, которое встретилось раньше). После этого выполняется метод `$app->run()`, который выполняет поиск подходящего маршрута и выполнение соответствующего ему действия.

Twig

Twig - это шаблонизатор для PHP. Шаблонизатор предназначен для того, чтобы вставлять данные, полученные из контроллера в сверстанные html-макеты, вроде тех, которые делали в лабораторной работе 1. В принципе, PHP можно использовать для вставок и без шаблонизатора, но код получается громоздким, плохо читаемым, небезопасным. Twig предоставляет простой и лаконичный язык для вставки данных в html. Кроме того, шаблоны Twig очень легко вставлять друг в друга. Это делается с помощью наследования. Например, рассмотрим создадим простой базовый шаблон и сохраним его в файле `base.twig`:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>{% block title %}{% endblock %}</title>
</head>
<body>
<div class="container">
{% block content %}{% endblock %}
</div>
</body>
</html>
```

В этом шаблоне два пустых блока: `title` для вставки заголовка страницы и `content` для генерации содержимого. Создадим шаблон-наследник от этого шаблона:

```
{% extends "BASE.twig" %}
{% block content %}
<h1>{{ object.data }}</h1>
{% endblock %}
```

Краткий набор примеров можно найти тут. Полную документацию смотрите на официальном сайте.

Doctrine

Doctrine - одно из популярных решений для уровня абстракции данных в PHP. Это двухуровневый фреймворк. Нижний слой DBAL (Data base adstraction layer - уровень абстракции базы данных) - это библиотека, которая позволяет делать приложение на php существенно независимым от используемой СУБД. Именно этот уровень будет использоваться в данной лабораторной работе. Второй уровень - ORM (object-relational mapping - объектно-реляционное отображение) отображает записи базы данных в объекты, переводя реляционную парадигму в более удобную для прикладного программиста объектную. На этом уровне Doctrine использует свой язык запросов DQL.

Порядок выполнения

Репозиторий с примером лабораторной работы

Установка и настройка

1. Скачать Silex и распаковать в папку, общую с vagrant-коробкой.
2. Настроить виртуальный хост `1.mesdt` на сервере nginx. Проверить работоспособность. Сконфигурируйте перенаправление на `index.php`, используйте настройки для вашего веб-сервера (http://silex.sensiolabs.org/doc/web_servers.html).
    - Для **nginx**. На виртуальной машине создать файл `/etc/nginx/sites-available/1.mesdt` по аналогии с файлом `0.mesdt`. Добавить конфигурацию из документации.
    - Для **apache**. Настроить виртуальный хост и создать в папке `web` файл `.htaccess` в соответствии с документацией.
    - Можно также запустить встроенный веб-сервер интерпретатора php. Выполните на машине, на которой установлен php (то есть, в консоли виртуальной машины, или в консоли вашего компьютера для пользователей XAMPP):
3. `php -S localhost:8000`
4. Вся логика веб-приложения будет реализована в файле `/web/index.php`. Однако, кроме собственно html-страниц наш сервер должен отдавать браузеру css, js-скрипты и картинки. Это просто файлы, которые лежат в директории сайта. Все эти файлы называются статическими ресурсами сайта. Для того, чтобы браузер имел доступ к ним, добавим в начало `index.php` соответствующую проверку. Если пришедший от пользователя запрос содержит в url имя файла, то скрипт передает управление веб-серверу.

```
// Для php -S host:port -t web web/index.php, чтобы статика отдавалась
if (php_sapi_name() === 'cli-server' && is_file(__DIR__ .
preg_replace('/(\?.*)$/', '', $_SERVER['REQUEST_URI']))) {
return false;
}
use Silex\Application;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
```

Подключение к базе данных

1. Подключите пространства имен для работы с Doctrine

```
use Doctrine\DBAL\Connection;
use Silex\Provider\DoctrineServiceProvider;
```

4. Зарегистрируйте компонент в контейнере.

```
$sapp->register(new DoctrineServiceProvider(),
['db.options' => ['driver' => 'pdo_mysql', 'dbname' => 'hw1', 'charset'
=> 'utf8']]);
```

7. Используйте функции DBAL для получения данных

```
$conn = $app['db'];
$students = $conn->fetchAll('select * from students');
```

Подключение шаблонов

1. Для подключения сервиса Twig используется следующее пространство имен:

```
use Silex\Provider\TwigServiceProvider;
```

3. Создайте папку для шаблонов рядом с папками `vendor` и `web`. Подключаем в контейнер:

```
$sapp->register(new TwigServiceProvider(),['twig.path' => __DIR__ .
'/../views'])
```

5. Создайте базовый шаблон из макета, который вы сверстали на первой лабораторной.
6. Создайте шаблоны для списка объектов и для одного объекта.
7. Подключите их в соответствующих действиях.

```
return $app['twig']->render('students.twig', ['students' => $students]);
```

Порядок защиты работы

При защите лабораторной работы необходимо предоставить работающее web-приложение, реализованное на фреймворке Silex и запущенное на web-сервере. Приложение должно выводить список объектов из базы данных и предоставлять возмжности редактирования одной из сущностей. Отчет должен содержать титульный лист, задание, код контроллера, код шаблонов Twig.

Ссылки на документацию

- Документация по PHP http://php.net/docs.php
- Документация по PHP на русском http://php.su
- PHP Silex Docs http://silex.sensiolabs.org/documentation
- Документация по Twighttp://twig.sensiolabs.org/
- Документация по Doctrine http://www.doctrine-project.org/
- Конфигурирование сервера для сайта под Silex http://silex.sensiolabs.org/doc/web_servers.html

Composer - менеджер зависимостей для PHP. Вы можете описать от каких библиотек зависит ваш проект и Composer установит нужные библиотеки за вас.

Для совместимости различных компоновщиков был разработан стандарт именования PSR-0. Ниже перечислены эти требования:

- Класс и пространство имен должны иметь следующую структуру: `\<Vendor Name>\(<Namespace>)*\<Class Name>`.
- Каждое пространство имен должно иметь пространство верхнего уровня (`Vendor Name`).

- Каждое пространство имен может иметь столько подпространств, сколько ему необходимо.
- Каждый разделитель пространства имен будет заменен на *DIRECTORY_SEPARATOR (константа php, отвечающая за разделение каталогов путях к файлам, например, прямой или обратный слеш),*\ когда будет загружаться из файловой системы.
- Каждое нижнее подчеркивание в имени класса будет заменено на `DIRECTORY\_SEPARATOR`. Нижнее подчеркивание не имеет такого значения для пространства имен.
- Пространства имен и имя класса будут иметь суффиксы `.php` при загрузке из файловой системы.
- Алфавитные символы в именах поставщика (`Vendor Name`), пространствах имен и именах классов могут быть в любой последовательности нижнего и верхнего регистров.

Согласно стандарту PSR-0 должен быть верхний уровень каталогов с именем поставщика и затем названием пакета.

Таким образом, путь к классу `Example.php` в пакете `Package` производителя `Vendor` будет иметь вид:`Vendor\ Package\ Example.php`

Классы будут использовать пространства имен соответственно:

```
<?php
  namespace Vendor\Package;
  class Example
  {
  }
```

Таким образом, определение класса для `\ Doctrine\ Common\ Connections` будет найдено в `/path/to/project/lib/Doctrine/Common/Connections.php`, а `\ Symfony\ Core\ Request` в `/path/to/project/lib/Symfony/Core/Request.php`. Стандарт PSR-0 не определяет базовую `/path/to/project/lib` часть пути, и соответствующие автозагрузчики предлагают различные методы для решения этой задачи. Composer по умолчанию скачивает библиотеки в папку `vendor`.

Ссылка на скачивание Composer и подробная документация по нему находятся на сайте https://getcomposer.org/

Cheat-list PHP

Краткое описание

Синтаксис PHP подобен синтаксису языка Си. Некоторые элементы, такие как ассоциативные массивы и цикл foreach, заимствованы из Perl.

Для работы программы не требуется описывать какие-либо переменные, используемые модули и т. п. Любая программа может начинаться непосредственно с оператора PHP.

Простейшая программа Hello world на PHP выглядит следующим образом:

```
<?php
  echo 'Hello, world!';
?>
```

PHP исполняет код, находящийся внутри ограничителей `<?php ?>`. Всё, что находится вне ограничителей, выводится в http-ответ без изменений. В основном это используется для вставки PHP-кода в HTML-документ, например, так:

```
<html>
 <head>
  <title>Тестируем PHP</title>
 </head>
 <body>
  <?php echo 'Hello, world!'; ?>
 </body>
</html>
```

Имена переменных начинаются с символа `$`, тип переменной объявлять не нужно. Имена переменных, функций и классов чувствительны к регистру. Константы также чувствительны к регистру.

PHP рассматривает переход на новую строку как пробел, так же как HTML и другие языки со свободным форматом. Инструкции разделяются с помощью точки с запятой (`;`) по правилам очень близким к языку Си (есть несколько исключений, но если вы ставите точку с запятой как в Си, то проблем не будет).

Переменные в функцию можно передавать как по значению, так и по ссылке (используется знак &).

PHP поддерживает три типа комментариев: в стиле языка Си (ограниченные `/* */`), C++ (начинающиеся с `//` и идущие до конца строки) и оболочки UNIX (с `#` до конца строки).

Типы данных

PHP является языком программирования с динамической типизацией, не требующим указания типа при объявлении переменных, равно как и самого объявления переменных. Преобразования между скалярными типами зачастую осуществляются неявно без дополнительных усилий (впрочем, PHP предоставляет широкие возможности и для явного преобразования типов).

К скалярным типам данных относятся:

- целый тип (`integer`),
- вещественный тип данных (`float, double`),
- логический тип (`boolean`),
- строковый тип (`string`),
- специальный тип `NULL`.

К нескалярным типам относятся:

- <<ресурс>> (`resource`),
- массив (`array`),
- объект (`object`),

К псевдотипам относятся:

```
- `mixed` один или несколько необязательных параметров,
```

- `number` число (`integer` либо `float`)
- `callback` (`string` или анонимная функция)
- `void` отсутствие параметров

Диапазон целых чисел (`integer`) в PHP зависит от платформы (обычно, это диапазон 32-битных знаковых целых чисел, то есть, от -2 147 483 648 до 2 147 483 647). Числа можно задавать в десятичной, восьмеричной и шестнадцатеричной системах счисления. Диапазон вещественных чисел (`double`) также зависит от платформы (для 32-битной архитектуры диапазон позволяет оперировать числами от ±1.7?10-308 до ±1.7?10+308).

PHP предоставляет разработчикам логический тип (`boolean`), способный принимать только два значения TRUE («истина») и FALSE («ложь»). При преобразовании в логический тип число 0, пустая строка, ноль в строке «0», NULL и пустой массив считаются равными FALSE. Все остальные значения автоматически преобразуются в TRUE.

Специальный тип NULL предназначен для переменных без определенного значения. Единственным значением данного типа является константа NULL. Тип NULL принимают неинициализированные переменные, переменные инициализированные константой NULL, а также переменные, удаленные при помощи конструкции `unset()`.

Строки могут заключаться в одинарные или в двойные кавычки. Если строка в двойных кавычках, то непосредственно в ней можно указывать переменные, и вместо переменной будет вставлено ее значение. Поэтому строки в двойных кавычках обрабатываются медленнее, чем в одинарных. Конкатенация строк осуществляется с помощью операции "."">. С помощью этого оператора можно собрать строку из нескольких элементов различных типов (числа автоматически переводятся в строки, у объектов вызывается метод `toString()`).

Ссылки на внешние ресурсы имеют тип «ресурс» (`resource`). Переменные данного типа, как правило, представляют собой дескриптор, позволяющий управлять внешними объектами, такими как файлы, динамические изображения, результирующие таблицы базы данных и т. п.

Массивы (`array`) поддерживают числовые и строковые ключи и являются гетерогенными. Массивы могут содержать значения любых типов, включая другие массивы. Порядок элементов и их ключей сохраняется. Не совсем корректно называть php-массивы массивами, на самом деле это, скорее всего, упорядоченный хеш. Возможно неожиданное поведение при использовании цикла `for` со счетчиком вместо `foreach`. Так, например, при сортировке массива с численными индексами функциями из стандартной библиотеки, сортируются и ключи тоже.

Указатель на функцию в PHP может быть представлен замыканием или псевдотипом `callback`. Замыкание доступно с версии 5.3 и в коде выглядит как простое определение функции, в которую явно можно утянуть значения из контекста, например:

```
function($args..$argsN) use($ctxVar,$ctxVar1) { definition ; }
```

Для проверки является ли значение вызываемым следует использовать `is_callable($var)`.

Обращение к переменным и функциям

Обращение к переменным осуществляется с помощью символа `${}, за которым следует имя переменной. Данная конструкция может быть применена также для создания динамических переменных и функций. Например:

```
$a = 'I am a'; // Запись значения в переменную $a
echo $a; // Вывод переменной $a

$b = 'a';
echo $$b; // Вывод переменной $a (дополнительный $ перед переменной $b)

echo ${'a'}; // Вывод переменной $a

function_name(); // Вызов функции function_name
$c = 'function_name';
$c(); // Вызов функции function_name,

$d = 'Class_name';
$obj = new Class_name; // Создание объекта класса Class_name
$obj = new $d(); // Создание объекта класса Class_name

$obj->b; // Обращение к полю b объекта
$obj->c(); // Вызов метода c() объекта

$obj->$b; // Обращение к полю a объекта, так как $b = 'a'
$obj->$c(); // Вызов метода function_name() объекта, так как $c =
'function_name'
```

В PHP `echo` и `print` не являются функциями (хотя `print` имеет возвращаемое значение), а являются синтаксическими единицами. При их использовании можно опустить скобки.

Суперглобальные массивы

Суперглобальными массивами в PHP называются предопределенные массивы, имеющие глобальную область видимости без использования директивы global. Большая часть этих массивов содержит входные данные запроса пользователя (параметры GET-запроса, поля форм при посылке методом POST, куки и т. п.).

Все суперглобальные массивы, кроме `$GLOBALS` и `$_REQUEST`, имеют устаревшие аналоги с длинными именами, которые доступны вплоть до версии 5.3.x (начиная с 5.4.0 были удалены). Таким образом, обращения `$_GET['year']` и `$HTTP_GET_VARS['year']` идентичны (за исключением области видимости: массивы с «длинными» именами не являются суперглобальными).

`$GLOBALS` Массив всех глобальных переменных (в том числе и пользовательских).

`$_SERVER` (устаревший аналог - `$HTTP_SERVER_VARS`) Содержит переменные окружения, которые операционная система передает серверу.

`$_ENV` Текущие переменные среды. Их набор специфичен для платформы, на которой выполняется скрипт.

`$_GET` Содержит параметры GET-запроса, переданные в URI после знака вопроса <<?>>.

`$_POST` Ассоциативный массив значений полей HTML-формы при отправке методом POST. Индексы элементов соответствуют значению свойства name объектов(кнопки, формы, радио-кнопки, флажки и т.д.) HTML-формы.

`$_FILES` Ассоциативный массив со сведениями об отправленных методом POST файлах. Каждый элемент имеет индекс, идентичный значению атрибута «name» в форме, и, в свою очередь, также является массивом со следующими элементами:

```
        -    `['name']` - исходное имя файла на компьютере пользователя.
    -              `['type']` - указанный агентом пользователя MIME-тип
файла. PHP не проверяет его, и поэтому нет никаких гарантий, что указанный
тип соответствует действительности.
  -  `['size']` - размер файла в байтах.
 -   `['tmp_name']` - полный путь к файлу во временной папке. Файл необходимо
переместить оттуда функцией `move_uploaded_file}. Загруженные файлы из
временной папки PHP удаляет самостоятельно.
  -  `['error']` - код ошибки. Если файл удачно загрузился, то этот элемент
будет равен 0 (`UPLOAD_ERR_OK}).
```

`$_COOKIE` Ассоциативный массив с переданными агентом пользователя значениями куки.

`$_REQUEST` Содержит элементы из массивов `$_GET`, `$_POST`, `$_COOKIE`. С версии PHP 4.1 включает `$_FILES`.

`$_SESSION` Содержит данные сессии.

Объектно-ориентированное программирование

PHP поддерживает широкие объектно-ориентированные возможности, полная поддержка которых была введена в пятой версии языка.

Класс в PHP объявляется с помощью ключевого слова `class`. Методы и поля класса могут быть общедоступными (`public`, по умолчанию), защищенными (`protected`) и скрытыми (`private`). PHP поддерживает все три основных механизма ООП - инкапсуляцию, полиморфизм и наследование (родительский класс указывается с помощью ключевого слова `extends` после имени класса). Поддерживаются интерфейсы (ставятся в соответствие с помощью `implements`). Разрешается объявление финальных, абстрактных методов и классов. Множественное наследование классов не поддерживается, однако класс может реализовывать несколько интерфейсов. Для обращения к методам родительского класса используется ключевое слово `parent`.

Классы в PHP имеют ряд специальных методов (magic methods), начинающихся с двух символов подчеркивания. Особо стоит отметить конструктор (`__construct()`, в версиях до 5.0 конструктором служил метод, одноименный с классом) и деструктор (__destruct()), а также методы чтения (`__get()`) и записи (`__set()`) и др. Эти методы являются достаточно гибким инструментом: переопределяя их, можно добиться существенного изменения поведения объекта.

Экземпляры класса создаются с помощью ключевого слова new, обращение к полям и методам объекта производится с использованием оператора ->. Для доступа к членам класса из его методов используется переменная `$this`.

```
class C1 extends C2 implements I1, I2
```

```
{
  private $a;
  protected $b;

  function __construct($a, $b)
  {
    parent::__construct($a, $b);
    $this->a = $a;
    $this->b = $b;
  }

  public function plus()
  {
    return $this->a + $this->b;
  }
/* ............... */
}

$d = new C1(1, 2);
echo $d->plus(); // 3
```

Начиная с пятой версии PHP, объекты передаются по ссылке:

```
class a
{
  public $color = 'red';
}

$a = new a();
echo $a -> color; // red
$b = $a;
$b -> color = 'blue';
echo $a -> color; // blue
<?php
class MyClass {
  const CONST_VALUE = 'Значение константы';
}
// Использование :: вне объявления класса
echo MyClass::CONST_VALUE;
?>
```

Сслыка на документацию

- Документация по PHP http://php.net/docs.php
- Документация по PHP на русском http://php.su

Cheat list Twig

How to

Вставка значения из контроллера

Вывести значение параметра `foo1`

```
{{ foo1 }}
```

Вывести значение из массива `foo` по ключу `bar`. Варианты эквивалентны.

```
{{ foo2.bar }}
{{ foo2['bar'] }}
```

Вставка блока и пример наследования

Базовый шаблон `base.html`:

```
<!DOCTYPE html>
<html>
    <head>
        {% block head %}
            <link rel="stylesheet" href="style.css" />
            <title>{% block title %}{% endblock %} - My Webpage</title>
        {% endblock %}
    </head>
    <body>
        <div id="content">{% block content %}{% endblock %}</div>
        <div id="footer">
            {% block footer %}
                &copy; Copyright 2011 by <a
href="http://domain.invalid/">you</a>.
            {% endblock %}
        </div>
    </body>
</html>
```

Шаблон наследник, переопределяет блоки, остальное остается на месте;

```
{% extends "base.html" %}

{% block title %}Index{% endblock %}
{% block head %}
    {{ parent() }}
    <style type="text/css">
        .important { color: #336699; }
    </style>
{% endblock %}
{% block content %}
    <h1>Index</h1>
    <p class="important">
        Welcome to my awesome homepage.
    </p>
{% endblock %}
```

Условия

```
{% if kenny.sick %}
    Kenny is sick.
{% elseif kenny.dead %}
    You killed Kenny! You bastard!!!
{% else %}
    Kenny looks okay --- so far
{% endif %}

{% if users|length > 10 %}
    ...
{% endif %}
```

Циклы

```
        {% for item in navigation %}
            <li><a href="{{ item.href }}">{{ item.caption }}</a></li>
        {% endfor %}
```

Сслыка на документацию

http://twig.sensiolabs.org/documentation

Краткий справочник по функциям Doctrine DBAL

Ссылка на документацию по Doctrine DBAL

Doctrine DBAL Connection API

Функции для выборки из таблицы

Если вам необходима более сложная функциональность команд изменения, используйте функцию `executeQuery()`.

fetchAll

```
fetchAll( string $sql, array $params = array() )
```

Выполняет запрос на выборку данных и возвращает ассоциативный массив с результатом.

Параметры:

- `$sql` - SQL-запрос, как правило, `select`.
- `$params` - параметры запроса в виде массива.

Например:

```
 $students = $conn->fetchAll('select * from students');
```

Результат будет массивом:

```
array(4) {
  [0]=>
  array(2) {
    ["id"]=>
    string(1) "1"
    ["name"]=>
    string(12) "Иванов"
  }
  [1]=>
  array(2) {
    ["id"]=>
    string(1) "4"
    ["name"]=>
    string(14) "Иванова"
  }
  [2]=>
  array(2) {
    ["id"]=>
    string(1) "3"
    ["name"]=>
    string(12) "Петров"
```

```
  }
  [3]=>
  array(2) {
    ["id"]=>
    string(1) "2"
    ["name"]=>
    string(16) "Сидорова"
  }
}
```

Пример с параметром:

```
 $students = $conn->fetchAll('select * from students where id > ?',array(2));
```

Результат на тех же данных:

```
array(2) {
  [0]=>
  array(2) {
    ["id"]=>
    string(1) "4"
    ["name"]=>
    string(14) "Иванова"
  }
  [1]=>
  array(2) {
    ["id"]=>
    string(1) "3"
    ["name"]=>
    string(12) "Петров"
  }
}
```

fetchAssoc

```
fetchAssoc( string $statement, array $params = array() )
```

Возвращает первую запись из результатов запроса в виде ассоциативного массива.

Параметры:

- `$sql` - SQL-запрос, как правило, `select`.
- `$params` - параметры запроса в виде массива.

Например:

```
 $students = $conn->fetchAssoc('select * from students');
```

Результат будет массивом:

```
array(2) {
  ["id"]=>
  string(1) "1"
  ["name"]=>
  string(12) "Иванов"
}
```

fetchArray

```
fetchArray( string $statement, array $params = array() )
```

Возвращает первую запись из результатов запроса в виде массива с индексами - целыми числами.

Параметры:

- `$sql` - SQL-запрос, как правило, `select`.
- `$params` - параметры запроса в виде массива.

Например:

```
 $students = $conn->fetchArray('select * from students');
```

Результат будет массивом:

```
array(2) {
  [0]=>
  string(1) "1"
  [1]=>
  string(12) "Иванов"
}
```

Функции для изменения в таблице

Если вам необходима более сложная функциональность команд изменения, используйте функцию `executeUpdate()`.

insert

```
insert( string $tableName, array $data, array $types = array() )
```

Добавляет строку в таблицу. Возвращает количество добавленных строк.

Параметры:

- `$tableName` - имя таблицы, которую будем обновлять.
- `$data` - данные, которые надо записать в таблицу; представляет собой ассоциативный массив, где ключи - названия полей, значения - данные, которые надо записать в эти поля.
- `$types` - список типов, в которые надо преобразовать данные (если это необходимо).

delete

```
delete( string $tableName, array $identifier )
```

Удаляет из таблицы данные, соответствующие описанным условиям.

Параметры:

- `$tableName` - имя таблицы, которую будем обновлять.
- `$identifier` - описание критерия, по которому определяются редактируемые записи; это тоже ассоциативный массив, где ключами являются имена полей, а значением - данные. На основе этого критерия формируются условия для `WHERE`. Например, идентификатор конкретной записи: `array('id' => 4)` или, например, все записи, у которых установлен флаг в поле `moderated`: `array('moderated' => 1)`.

update

```
update( string $tableName, array $data, array $identifier, array $types =
array() )
```

Выполняет операцию обновления записи в таблице. Возвращает количество измененных записей.

Параметры:

- `$tableName` - имя таблицы, которую будем обновлять.
- `$data` - данные, которые надо записать в таблицу; представляет собой ассоциативный массив, где ключи - названия полей, значения - данные, которые надо записать в эти поля.
- `$identifier` - описание критерия, по которому определяются редактируемые записи; это тоже ассоциативный массив того же вида, что и `$data`, на основе которого формируются условия для `WHERE`. Например, идентификатор конкретной записи: `array('id' => 4)` или, например, все записи, у которых установлен флаг в поле `moderated`: `array('moderated' => 1)`.
- `$types` - список типов, в которые надо преобразовать данные (если это необходимо).

Лабораторная работа 3. Java Spring Boot

Цели работы

- Познакомиться с фреймворком Spring Boot и уровнем абстракции базы данных Java Persistence Api.
- Изучить паттерн ORM и Dependency Injection на примере Spring Boot.

Задание

Реализовать CRUD для одной из сущностей из лабораторной работы 1 с использованием фреймворка Spring Boot и шаблонов из лабораторной работы 1.

Краткие теоретические сведения

- Java и сборщик проектов Maven
- Java Spring
- Java Persistence Api

Порядок выполнения

- Репозиторий с примером обработки REST-запросов (для ознакомления)
- Репозиторий с примером лабораторной работы

Рекомендуемая версия: Java 1.8 Возможно использование Java 1.6 и выше, но лучше обновитесь, новая Java круче!

Установка и настройка

1. Установите Maven. Для linux это можно сделать командой командой `apt-get install maven`. Для остальных операционных систем - смотрите на официальном сайте.
2. Зайдите на сервис для создания заготовки проекта Spring Boot. Изучите доступные параметры. Создайте проект. Для этого задайте в форме основные параметры для конфигурационного файла Maven. Введите идентификатор группы (например, `mesdt`) и идентификатор артефакта (например, `hw3`). Дополнительно можете задать имя проекта, описание и другие параметры. Отметьте необходимые зависимости: web для генерации веб-приложений, шаблонизатор Thymeleaf, библиотеку для работы с данными JPA и драйвер базы данных MySQL. Нажмите кнопку генерации пакета и скачайте архив.
3. В скачанном архиве находится заготовка проекта на Spring Boot. Распакуйте архив. Изучите структуру каталогов:

```
└── src
|   └── main
|   |   └── java
|   |   |   └── yourProjectName
|   |   |   |   └── SpringHomeWorkApplication.java
|   |   └── Resources
|   |       └── static
|   |       └── templates
|   |       └── application.properties
|   └── test
|       └── java
|           └── yourProjectName
```

```
16. |               └── SpringHomeWorkApplicationTests.java
17. └──  pom.xml
```

- `pom.xml` - файл конфигурации Maven;
- `test` - директория для тестов, в этом курсе рассматриваться не будет;
- `main` - каталог, содержащий исходные коды приложения, в подкаталоге `java/yourProjectName` будут располагарться файлы с исходным кодом.
- `static` - каталог для статических ресурсов;
- `templates` - каталог для файлов представлений;
- `application.properties` - конфигурационный файл Spring Boot;
- `SpringHomeWorkApplication.java` - главный файл проекта.

18. Настройте встроенный сервер Spring Boot в файле `application.properties`:

```
19. server.port=2345
```

В этом же файле указывается конфигурация базы данных для JPA:

```
spring.datasource.url=jdbc:mysql://localhost/hw2?useUnicode=yes&characterEncoding=UTF-8
spring.datasource.username=root
spring.datasource.password=
```

Включите автоматическую генерацию схемы и логгирование sql-запросов:

```
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
```

Теперь если вы укажете пустую базу данных, то схема, описанная аннотациями JPA, сгенерируется автоматически при первом запуске приложения.

Отключите кэширование шаблонов:

```
spring.thymeleaf.cache=false
```

Это позволит вам редактировать шаблоны без перезапуска сервера, что очень удобно при разработке. В готовом приложении кэширование необходимо включать для ускорения работы.

Разработка приложения

1. Создайте классы сущностей и репозитории. Опишите структуру сущностей с помощью аннотаций.
2. Создайте класс контроллера и необходимые действия (actions) в нем. Опишите инъекцию зависимости от репозиториев сущностей для контроллера. Необходимо создать CRUD для одной из сущностей.
3. Создайте представления с использованием шаблонизатора Thymeleaf.

Для сборки приложения со всеми зависимостями приложения используется команда `mvn clean install`.

Для запуска приложения используется команда `mvn spring-boot:run`.

Дополнительное задание на повышение рейтинга

Реализуйте слой бизнес-логики как сервис.

Порядок защиты работы

При защите лабораторной работы необходимо предоставить работающее web-приложение, реализованное на фреймворке Spring Boot. Приложение должно выводить список объектов из базы данных и предоставлять возмжности редактирования одной из сущностей. Отчет должен содержать титульный лист, задание и код приложения.

Ссылки на скачивание инструментов

- Maven http://maven.apache.org/
- Сервис для создания заготовки проекта Spring Boot http://start.spring.io/

Ссылки на документацию

- Документация по Java 8 http://docs.oracle.com/javase/8/
- Документация по Spring Boot http://docs.spring.io/spring-boot/docs/current/reference/html/
- Документация по JPA-аннотациям
- Документация по шаблонизатору Thymeleaf http://www.thymeleaf.org/

Java

Java - универсальный язык программирования со строгой статической типизацией. Программы на java компилируются не непосредственно в команды процессора, а в специальный байт-код, который выполняется в специальной среде выполнения - JVM (java virtual mashine). Как правило, скомпилированные файлы имеют расширение `.class`. Байт-код не привязан к платформе, поэтому программы на java являются кросс-платформенными. Кроме java, JVM является средой выполнения для других языков: Scala, Groovy, Clousure и т.д.

JVM поставляется в двух видах: JRE (java runtime environment) и JDK (java development kit). JRE содержит всё необходимое для выполнения программ, а JDK дополнительно содержит документацию и утилиты для разработки программ.

Библиотеки на java представляют собой архивы с расширением `jar`, содержащие `class`-файлы и вспомогательные файлы. Такие библиотеки называются артефактами, и идентифицируются следующим набором параметров:

- group id - идентификатор группы пакетов (например, все пакеты одной компании могут принадлежать к одной группе);
- artifact id - идентификатор артефакта;
- version - версия артефакта.

Генерики и аннотации

Генерики (Generics) в Java - это аналог шаблонов в языке C++. Изначально они появились для упрощения кода работы с коллекциями. Сравните: - код без генериков:

```
```
List myIntList = new LinkedList();
myIntList.add(new Integer(0));
```
```

```
Integer x = (Integer) myIntList.iterator().next();
```

- код с генериками:

```
List<Integer> myIntList = new LinkedList<Integer>();
myIntList.add(new Integer(0));
Integer x = myIntList.iterator().next();
```

Аннотации в Java - специальная форма синтаксических метаданных, которая может быть добавлена в исходный код. Аннотации используются для анализа кода, компиляции или выполнения. Аннотируемы пакеты, классы, методы, переменные и параметры. Выглядит как `@ИмяАннотации`, предваряющее определение переменной, параметра, метода, класса, пакета.

Аннотация выполняет следующие функции:

- аннотация может использоваться для комментирования или генерации докуменатции;
- даёт необходимую информацию для компилятора;
- даёт информацию различным инструментам для генерации другого кода, конфигураций и т. д.;
- может использоваться во время выполнения для получения данных через отражение (reflection).

В частности, аннотации могут применяться для использования генериков.

Сервлеты

Java изначально разрабатывалась с ориентацией на сетевое взаимодействие. Классическим способом реализации веб-сервера на java являются сервлеты.

Сервлет является Java-интерфейсом, реализация которого расширяет функциональные возможности сервера. Сервлет взаимодействует с клиентами посредством принципа запрос-ответ. Сервлет не может выполняться самостоятельно, и требует специальной среды - контейнера сервлетов.

Контейнер сервлетов - программа, представляющая собой сервер, который занимается системной поддержкой сервлетов и обеспечивает их жизненный цикл в соответствии с правилами, определёнными в спецификациях.

Tomcat - контейнер сервлетов с открытым исходным кодом, разрабатываемый Apache Software Foundation. Реализует спецификацию сервлетов и спецификацию JavaServer Pages (JSP) и JavaServer Faces (JSF). Написан на языке Java. Tomcat позволяет запускать веб-приложения, написанные на java. Tomcat используется в качестве самостоятельного веб-сервера, в качестве сервера контента в сочетании с веб-сервером Apache HTTP Server, а также в качестве контейнера сервлетов в серверах приложений JBoss и GlassFish.

Жизненный цикл сервлета:

1. Сервлет собирается компилятором java в байт-код в виде `class`-файла.
2. В случае отсутствия сервлета в контейнере:
    1. Класс сервлета загружается контейнером.

    2. Контейнер создает экземпляр класса сервлета.
    3. Контейнер вызывает метод `init()`. Этот метод инициализирует сервлет и вызывается в первую очередь, до того, как сервлет сможет обслуживать запросы. За весь жизненный цикл метод `init()` вызывается только один раз.
3. Обслуживание клиентского запроса. Каждый запрос обрабатывается в своем отдельном потоке. Контейнер вызывает метод `service()` для каждого запроса. Этот метод определяет тип пришедшего запроса и распределяет его в соответствующий этому типу метод для обработки запроса. Разработчик сервлета должен предоставить реализацию для этих методов. Если поступил запрос, метод для которого не реализован, вызывается метод родительского класса и обычно завершается возвращением ошибки инициатору запроса.
4. В случае если контейнеру необходимо удалить сервлет, он вызывает метод `destroy()`, который снимает сервлет из эксплуатации. Подобно методу `init()`, этот метод тоже вызывается единожды за весь цикл сервлета.

Развертывание - это процесс размещения и запуска web-приложения и всех необходимых компонентов на сервере, в данном случае -- в сервлет-контейнере.

Дескриптор развертывания - это конфигурационный файл артефакта, который будет развернут в сервлет-контейнере.

Maven

Apache Maven - фреймворк для автоматизации сборки проектов, специфицированных на XML-языке POM (Project Object Model). Maven позволяет эффективно управлять зависимостями проекта, автоматически скачивая необходимые библиотеки.

Многие артефакты используют в своей работе другие артефакты. Вручную отслеживать эти зависимости очень трудоемко. Maven автоматизирует этот процесс. Кроме этого Maven позволяет стандартизировать условия сборки и запуска проектов, а также имеет множество плагинов с различной функциональностью.

Пример pom-файла для лабораторной работы 3:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>mesdt</groupId>
    <artifactId>hwspring</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>Spring Home Work</name>
    <description>Spring Home Work</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.2.1.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
```

```
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <start-class>hwspring.SpringHomeWorkApplication</start-class>
        <java.version>1.7</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Для работы с фреймворком Spring Boot используется специальный плагин.

Дополнительная литература

- Документация по Java 8 http://docs.oracle.com/javase/8/
- Документация по Maven http://maven.apache.org/guides/index.html
- Список плагинов Maven http://maven.apache.org/plugins/index.html
- Документация по плагину Spring Boot для Maven
- Официальный учебник по аннотациям
- Официальный учебник по генерикам
- Конспект по генерикам на сайте ИТМО

Java Spring

Java Spring - огромный фреймворк для языка Java. Приложение на Java Spring реазилует паттерн Dependency Injection и представляет собой контейнер компонентов, которые называаются бинами (beans) или артефактами (artifacts). В лабораторной мы будем рассматривать фреймворк Spring Boot, который представляет собой Spring с набором стандартных настроек для реализации веб-приложения. Приложение Spring Boot содержит в себе встроенный сервер Tomcat.

Конфигурирование контейнера компонентов может производиться двумя способами: с помощью конфигурационных файлов или с помощью аннотаций непосредственно в коде.

Основные аннотации Java Spring

`@Autowired` - внедрение зависимости. В поле, помеченное этой аннотацией, фреймворк подставит компонент соответствующего типа. Если компонентов заданного типа несколько, нужно указать название необходимого компонента. Аннотация `@Autowired` может быть указана к сеттеру, конструктору или методу. Тогда внедрение зависимости происходит в параметры этих функций.

`@Component` - аннотация говорит, что класс должен управляться контейнером Spring. `@Repository`, `@Service` и `@Controller` - специальные виды компонентов.

`@Repository` - аннотация указывает, что класс выполняет роль репозитория данных.

`@Service` - компонент уровня бизнес-логики.

`@Controller` - контроллер или компонент уровня приложения.

`@Bean` - помечает методы, ответственные за создание компонентов.

`@SpringBootApplication` - дополнительная аннотация Spring Boot, предназначена для обозначения главного класса приложения. Включает в себя три аннотации Spring: - `@Configuration` означает, что класс содержит описание методов, отвечающих за создание компонентов. - `@EnableAutoConfiguration` говорит фреймворку, что надо автоматически включить инструменты создания компонентов, которые расположены в Class Path. - `@ComponentScan` конфигурирует поиск компонентов, которые необходимо включить в приложение.

Аннотации контроллера

`@RequestMapping` - аннотация, описывающая маршруты. Имеет два основных параметра:

- `method` - http-метод;
- `value` - шаблон маршрута.

Фрагменты маршрута, являющиеся параметрами, заключаются в фигурные скобки. Пример:

```
@RequestMapping(method = RequestMethod.GET, value = "/students/{id}")
public String student(@PathVariable Long id, Model vars) {
    ...
}
```

`@RequestParam` - параметр запроса.

`@PathVariable` - параметр запроса, извлеченный из пути.

Дополнительная литература

- Документация по Spring Boot http://docs.spring.io/spring-boot/docs/current/reference/html/
- Документация по плагину Spring Boot для Maven

Java Persistence Api

Java Persistence Api - это один из классических способов реализации уровня абстракции данных в Java.

Entity (Сущность) - класс, связанный с БД с помощью аннотации (`@Entity`) или через XML. К такому классу предъявляются следующие требования:

- должен иметь пустой конструктор (`public` или `protected`);
- не может быть вложенным, интерфейсом или `enum`;
- не может быть final и не может содержать `final`-полей/свойств;
- должен содержать хотя бы одно `@Id`-поле.

При этом сущность может:

- содержать непустые конструкторы;
- наследоваться и быть наследованным;
- содержать другие методы и реализовывать интерфейсы.

Сущности могут быть связаны друг с другом.

Документация по JPA-аннотациям

Документация по методам репозиториев Spring Data JPA

Аннотации описания таблиц

`@Entity` сообщает фреймворку, что данный класс используется как класс сущности. Пишется перед определением класса.

`@Table(foo)` конфигурирует имя таблицы, в которой будет храниться сущность (в данном случае имя таблицы `foo`).

`@Id` указывайте перед полями класса, которые создают первичный ключ таблицы. Классический пример:

```
@Id
private int ID;
```

Если вы хотите создать составной ключ, используйте `@IdClass` или `@EmbeddedId`.

`@Column` используется для задания свойств хранения отдельных полей сущности в таблице:

```
@Column(nullable = false, unique = true)
private String name;
```

Аннотации описания связей

Аннотации `@OneToOne`, `@ManyToOne`, `@OneToMany`, `@ManyToMany` используются для описания связей между сущностями.

Пример:

```
@OneToMany(mappedBy = "id.student", cascade = CascadeType.REMOVE)
private Collection<Score> scores;
```

Аннотации описания событий

JPA позволяет задать функции для обработки собылий в классе сущности:

- `@PrePersist` - перед сохранением сущности в базу;
- `@PostPersist` - после сохранения сущности;
- `@PreRemove` - перед удалением записи;
- `@PostRemove` - после удаления записи;
- `@PreUpdate` - перед обновлением записи;
- `@PostUpdate` - после обновления записи.

Репозитории Spring Data JPA

В Spring для работы с базой данных используется специальный вид компонентов - репозитории. Это интерфейсы, расширяющие класс `JpaRepository<EntityClass, EntityIdClass>`.

Кроме набора стандартных методов (например, `findAll`) вы можете получить довольно сложные запросы, используя генерируемые методы. Например, метод `findByAgeOrderByLastnameDesc` будет искато по полю `age` (значение должно быть передано в параметрах) и сортировать по убыванию по полю `lastname`. **Вам достаточно описать такой метод в вашем интерфейсе репозитория. Код писать не надо, Spring Data сгенерирует его автоматически!** Все ключевые слвоа для создания таких методов описаны в документации.

Документация по методам репозиториев Spring Data JPA