Содержание

Введение 3

1 Введение в практику 5

2 Знакомство с предприятием 8

3 Работа по индивидуальному заданию 9

4 Работа с базами данных 24

5 Работа в INTERNET 28

6 Работа с мультимедиа 31

Заключение 38

Список литературы 39

Введение

Целью прохождения производственной практики является: получение практических навыков, подготовка к написанию отчетной работы, сбор материалов. Производственная практика является важным этапом профессиональной подготовки студентов. В ходе практической деятельности предполагается получение наиболее глубоких и конкретных знаний, умений и навыков, которые в дальнейшем обеспечат успешную трудовую деятельность по направлениям. Проходил практику в ТОО «Институт Гипроуглегормаш» с 04.09.17 года по 27.11.17.

Основной целью производственной практики являлось закрепление в производственных условиях теоретических знаний по курсу и смежных с ним дисциплин, приобретение практических навыков в области управления конкретным предприятием и получение практики проектирования и разработки информационных систем.

Основными целями и задачами прохождения производственной практики на предприятии являются: закрепление и углубление теоретических знаний, полученных по специальным дисциплинам, получение профессиональных навыков работы, сбор фактического материала для написания дипломной работы.

В ходе практики было произведено знакомство с направлениями деятельности предприятия и его структурных подразделений. Современный период развития цивилизованного общества характеризуется процессом информатизации.

Современный период развития цивилизованного общества характеризуется процессом информатизации.

Информатизация общества - это глобальный социальный процесс, особенность которого состоит в том, что доминирующим видом деятельности в сфере общественного производства является сбор, накопление, продуцирование, обработка, хранение, передача и использование информации, осуществляемые на основе современных средств микропроцессорной и вычислительной техники, а также на базе разнообразных научных средств информационного обмена.

Информатизация общества обеспечивает: активное использование постоянно расширяющегося интеллектуального потенциала общества, сконцентрированного в печатном фонде, и научной производственной и других видах деятельности; интеграцию информационных технологий в научные и производственные виды деятельности, инициирующую развитие всех сфер общественного производства, интеллектуализации трудовой деятельности. Применение открытых информационных систем, рассчитанных на использование всего массива информации, доступной в данный момент обществу в определенной его сфере, позволяет усовершенствовать механизм управления общественным устройством, способствует германизации и демократизации общества, повышает уровень благосостояния его членов. Процессы, происходящие в связи с информатизацией общества, способствуют не только ускорению научно-технического прогресса, интеллектуализации всех видов деятельности, но и созданию новой информационной среды социума, обеспечивающей развитие творческого индивида. Распространение автоматизированных офисов, широкое использование персональных компьютеров приводит к существенным изменениям в труде специалистов, что в свою очередь требует их специальной компьютерной подготовки. Поэтому необходимы специалисты со знанием персональных компьютеров, умеющие внедрять информационные технологии в технологический процесс.

В процессе прохождения производственной практики было затронуто:

- ознакомление с ПО;

- ознакомление с ОС;

- Json;

- JavaSE8;

- JDBS;

- Spark;

- SQL и MySQL;

- было участие в сборке компьютеров;

- ознакомление в составлении баз данных в программе MySQL.

1 Введение в практику

Техника безопасности – это система технических средств, мероприятий и приемов труда, обеспечивающих безопасность работающих.

Техника безопасности при работе с компьютером на предприятии предусматривает наличие общедоступной инструкции, в которой указаны обязательные требования к обустройству рабочего места и процессу использования техники. Эти правила едины для всех организаций, их выполнение контролируется руководящими органами.

Основные правила организации пространства вокруг рабочего места:

- при длительном и интенсивном использовании, на поверхности модулей ПК (системный блок, монитор, мышка и т.д.) возникают небольшие разряды тока. Эти частицы активизируются во время прикосновений к ним и приводят к выходу техники из строя. Нужно регулярно использовать нейтрализаторы, увлажнители воздуха, антистатики;

- вокруг стола не должно быть свисающих проводов, пользователь не должен контактировать с ними;

- важна целостность корпуса розетки и штепсельной вилки;

- отсутствие заземления пред экранного фильтра проверяется с помощью измерительных приборов;

- желательно во время строительных работ в офисе использовать минимальное количество легко воспламеняемых материалов (дерева, пенопласта), а также горючего пластика в изоляции.

- рекомендуется отдавать предпочтение кирпичу, стеклу, металлу и т.д.;

- помещение должно хорошо вентилироваться и охлаждаться в жаркую пору года. Важен своевременный отвод избыточного тепла от техники.

Требования безопасности для работников офиса должен быть проведен устный базовый инструктаж, в дальнейшем его печатный текст должен предоставляться для подробного изучения. Организация в обязательном порядке размещает информационный лист на видном месте. Инструктаж охватывает полный цикл контакта человека с компьютером. Он начинается с установки оборудования сервисной службой и заканчивается утилизацией непригодного устройства.

Перед началом работы, даже если речь идет о рабочем месте, которое используется каждый день и регулярно проверяется специалистами (как, например, в офисе или учебном заведении), нельзя терять бдительность. Перед тем, как включить компьютер, необходимо уделить пару минут следующим действиям: нужно убедиться в том, что в зоне досягаемости отсутствуют оголенные провода и различные шнуры. Они не только мешают работе, но и несут потенциальную опасность в случае короткого замыкания; нельзя начинать работу на технике с видимым повреждением. В случае обнаружения трещины на корпусе или повреждений другого рода, нужно обратиться за помощью в сервисный центр. Это же относится к ПК с неисправным индикатором включения/выключения. предметы на столе не должны мешать обзору, пользованию мышкой и клавиатурой. Поверхность экрана должна быть абсолютно чистой; на системном блоке не должно находиться никаких предметов, так как в результате вибраций может нарушиться работа устройства. Нужно убедиться в том, что никакие посторонние предметы не мешают работе системе охлаждения. недопустимо включать персональный компьютер в удлинители и розетки, в которых отсутствует заземляющая шина. Не тяни за кабель, запрещается начинать работу в помещениях с повышенной влажностью, а также в случае, если рядом присутствуют открытые источники влажности (лужи, мокрый пол). Включить технику можно лишь после полного высыхания окружающих предметов. недопустимо часто включать и выключать компьютер в течение рабочего дня без особой нужды. Система просто не справляется с необходимостью быстро сворачивать все процессы.

Поскольку персональный компьютер обладает всеми свойствами электрического прибора, то на него распространяются основные правила безопасности при взаимодействии с проводниками тока: нельзя размещать какие-либо вещи на поводах, а также самостоятельно менять их расположение без особой нужды; рекомендуется избегать расположения жидкостей рядом с модулями компьютера. Поэтому кулер с водой или кофейный автомат необходимо размещать в стороне от рабочих мест в офисе. Пользователи должны осознавать опасность потенциального замыкания в случае пролития воды на клавиатуру или системный блок. Нельзя работать на ПК с мокрыми руками; нельзя очищать поверхность компьютера от загрязнений, когда он находится во включенном состоянии; недопустимо снимать корпус любой из составных частей ПК во время его работы. Кроме того, разбор и ремонт техники имеют совершают только специализированные работники; во время работы на компьютере нельзя одновременно прикасаться к другим металлическим конструкциям, которые стоят на той же поверхности. Это касается отопительных батарей или трубопроводов; в помещении с компьютерами непозволительно курить или употреблять пищу непосредственно на рабочем месте; при ощущении даже незначительного запаха гари, нужно как можно быстрее выключить ПК из сети и обратиться к ответственному за обслуживание компьютерной техники.

Что бы минимизировать это влияние даже при длительном нахождении за монитором, стоит навсегда запомнить следующие постулаты: расстояние между глазами пользователя и экраном составляет не менее полуметра. Но пользователь должен быть в состоянии дотянуться кончиками пальцев до верхнего края монитора; стул стоит таким образом, что бы спина лишь немного упиралась в его спинку. Высота сидения позволяет держать ровную осанку; локти согнуты под прямым углом, а в кистях рук, лежащих на столе, не чувствуется напряжения; локти не висят в воздухе, а комфортно располагаются на подлокотниках кресла или столешнице. Их позиция существенно не меняется при передвижении мышки; ноги упираются в твердую поверхность, распрямлены вперед, а не подогнуты под себя; если пользователь носит очки, то нужно убедиться в том, что он может свободно регулировать угол наклона экрана. чрезвычайно важна периодическая зарядка. Каждый час нужно вставать с кресла, разминать мышцы и суставы. Ведь, несмотря на неподвижность, они испытывают огромную нагрузку, пребывая в неестественном положении. Обязательно нужно делать разминку для глаз: круговые и линейные движения открытыми глазами, моргание и расфокусирование.

В аварийных ситуациях своевременная бдительность поможет избежать опасных ситуаций для жизни и сохранить целостность техники. Действия в аварийных ситуациях: при неполадках любого рода в электроснабжении устройства необходимо сразу отключить компьютер от сети; если обнаружен оголенный провод, то необходимо оперативно оповестить всех работников офиса, не допуская чьего-либо контакта с ним; в каждом учреждении должны находиться огнетушители ОУБ-3 или ОУ-2, а также ведра и полотна в необходимом количестве. Персонал обязан знать о том, где находятся средства для гашения пламени и куда нужно звонить в случае пожара; при поражении человека электрическим током, прежде всего, оказывается первая помощь: искусственное дыхание и внешний интенсивный массаж сердца. В первые же мгновения после удара током, вызывается скорая помощь. По окончанию работы, перед завершением нужно правильно закрыть все программы и окна. Нельзя оставлять активные носители информации (диски и флэшки). Стоит отметить, что порядок выключения составляющих частей ПК отличается от порядка их включения ровно наоборот. Запуск компьютера происходит по цепочке: общее питание – периферия – системный блок. Выключение, соответственно, начинается с системного блока.

Правила размещения монитора. Монитор находится на расстоянии равной длине руки пользователя. Если диагональ экрана более двадцати дюймов, то дистанция больше; глаза находятся на уровне линии, которая на 5 сантиметров ниже верхнего края экрана. Для этого регулируется высота стула и монитора; экран находится по центру для того, что бы не нужно было задерживать шею в неестественном положении; окна в помещении не должны создавать блики на экране. Свет от окна не ярче, чем свет от ПК; чрезмерная контрастность и яркость изображения утомляют зрение, необходимо настраивать эти показатели; в офисах, где компьютеры расположены в два ряда, нужно устанавливать защитный промежуточный экран для устранения избыточного облучения.

2 Знакомство с предприятием

ТОО «Институт Гипроуглегормаш» является филиалом компании “Гефест”.

Начало рабочего дня - 9.00 часов. Окончание рабочего дня - 15.00 часов. Выходные дни - суббота, воскресенье. В течение ежедневной работы предоставляется обеденный перерыв продолжительностью 1 час в интервале с 12.00 до 13.00. Занимаются созданием WEB-сайтов.



Рисунок 1 – Сервер корпорации «Гефест».

Был выделен один персональный компьютер. Всего персональных компьютеров 52 в ТОО «Институт Гипроуглегормаш». Было знакомство с сервером корпорации «Гефест». На сервере (по окончанию практики) тестировался законченный проект.

3 Работа по индивидуальному заданию

SQL

Для начала работы с MySQL, нужно выучить и закрепить базовые знания по SQL. Создаем БД (Базу данных) и смотрим на результат создания БД.

Название БД «Tutorial». Создание БД показано на рисунке 2.

Листинг SQL:

SHOW DATABASES;

CREATE DATABASE Tutorial;

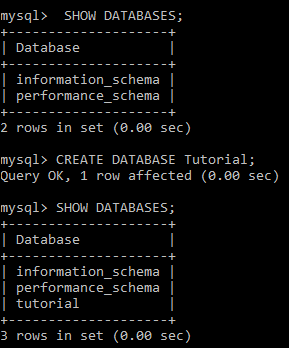


Рисунок 2 – Создаем БД

Вначале была проверка на существование БД под названием «tutorial», после чего была создана новая БД.

Нужно создать новую таблицу в «tutorial» . Название таблицы «user», атрибуты:

- id – порядковый номер user (целочисленный тип или int)

- first\_name – имя пользователя (строка или varchar(10))

- last\_name – фамилия пользователя (строка или varchar(10))

Листинг SQL:

USE tutorial;

CREATE TABLE user (id int primary key auto\_increment,

first\_name varchar(10),last\_name varchar(10));

primary key – это ключ, колонка помечается индивидуальностью

auto\_increment – это авто-заполнение при занесении данных в таблицу. Успешное создание таблицы показано на рисунке 3.



Рисунок 3 – Создаем таблицу

Заполнить таблицу данными.

1) id-1 first\_name-Vlad last\_name-Weber

2) id-2 first\_name-Oleg last\_name-Belocercovsciy

3) id-3 first\_name-Sergey last\_name-Zurow

Заполнение таблицы показано на рисунке 4.

Листинг SQL:

INSERT INTO user (first\_name, last\_name) values

(‘Vlad’,’Weber’);

INSERT INTO user (first\_name, last\_name) values

(‘Oleg’,’Belocer’);

INSERT INTO user (first\_name, last\_name) values

(‘Sergey’,’Zurow’);

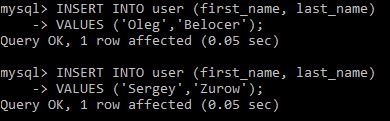


Рисунок 4 – Вносим данные в таблицу

Проверка внесенный данных в таблицу «user». Демонстрация данных в таблици показано на рисунке 5.

Листинг таблицы «user»:

SELECT \* FROM user;

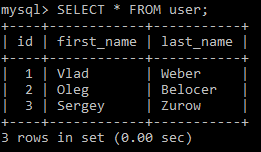


Рисунок 5 - Данные с первой таблицы

Изменить имя пользователя искать по id=2 изменить «first\_name» на «Denis» и удалить пользователя с id=3. Результат показан на рисунке 6

Листинг SQL:

UPDATE user SET firs\_name = ‘Denis’ WHERE id = 2;

DELETE user WHERE id = 3;

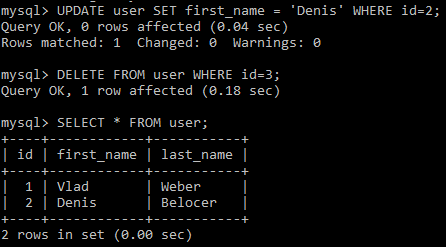


Рисунок 6 – Изменяем вторую строку

Создать новую таблицу, в которой будет занесен возраст и дата рождения пользователя.

Таблица будет называться «info», в которой следующие атрибуты:

- id – порядковый номер user (целочисленный тип или int)

- age - возраст пользователя (целочисленный тип или int)

- dataus – дата рождения пользователя (дата тип или data)

Успешное создание второй таблицы показано на рисунке 7

Листинг SQL:

CREATE TABLE info (id int primary key auto\_increment,

age int, dataus date);

INSERT INTO info (age, dataus) values

(19,’1998-01-19’);

INSERT INTO info (age, dataus) values

(19,’1998-06-05’);

SELECT user.first\_name, user.last\_name, info.age, info.dataus

FROM user, info WHERE user.id=info.id;

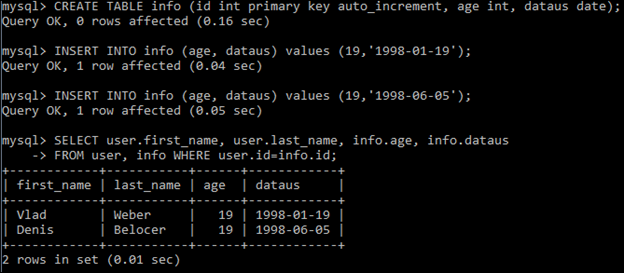


Рисунок 7 – Данные со второй таблицы

MySQL

MySQL - свободная реляционная система управления базами данных.

При открытии «MySQL Workbench 6.3 CE» нужно установить соединение MySQL.

Создаем новое соединение, нажав на значок “плюс”. На рисунке 8 показан значок.

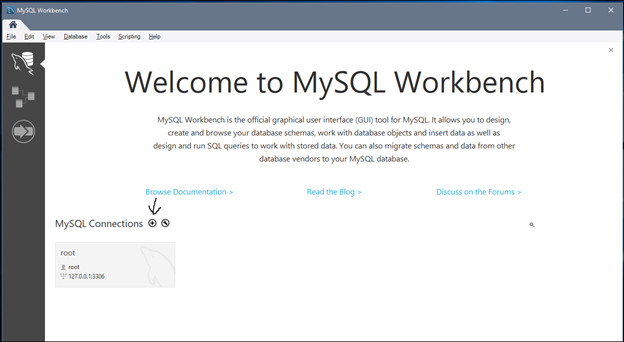


Рисунок 8 – Значок в MySQL

Делаем соединение по TCP/IP, имя хоста 127.0.0.1 (Для локального соединения) и порт 3306. UserName и Password «root».

Сверху кликаем на значок создания SQL файла, чтобы работать с БД.

Чтобы создать новую БД, нужно нажать на значок создания БД.

Нужно будет ввести название БД и выбрать кодировку «utf8-utf8\_general\_ci», чтобы поддерживать большое количество символов, в том числе и русский язык. На рисунке 9 показано как нужно добавлять новую БД.

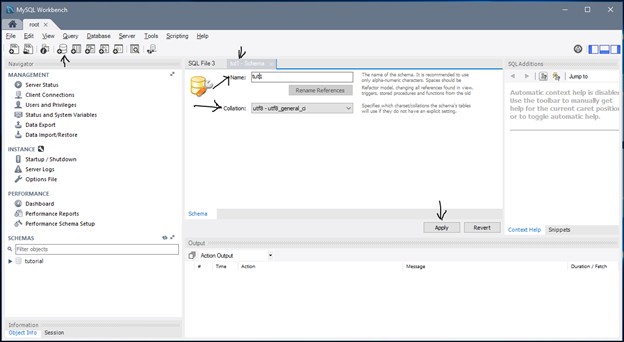


Рисунок 9 – Создание БД в MySQL

В колонке «Schemas» нужно выбрать созданную БД и кликнуть два раза. После того, как название БД выделяется жирным шрифтом, нужно нажать на значок создания новой таблицы и снова сделать «user» (Задание с прошлой SQL).

Наименование таблицы «User», кодировка «utf8-utf8\_general\_ci». Ниже составляем колонки в таблице, где:

PK – ключ для колонки;

NN – нельзя оставлять пустой;

AI – автозаполнение при внесении новых данных в таблицу

Для большей наглядности на рисунке 10 продемонстрировано создание таблицы.

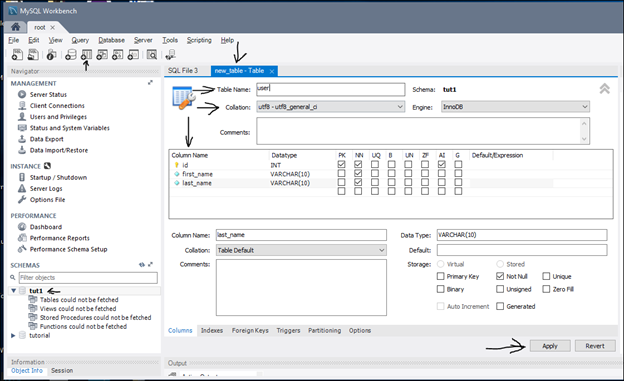


Рисунок 10 – Создание таблицы в MySQL

Переходим к SQL файлу и вводим нужный нам синтаксис. На рисунке 11 показан SQL и его результат.

Листинг SQL:

insert into user (first\_name, last\_name) values

('Vlad','Weber');

insert into user (first\_name, last\_name) values

('Oleg','Beloc');

Select \* from user;

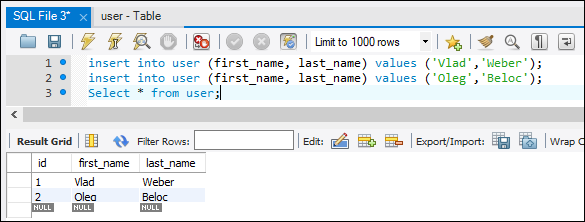


Рисунок 11 – Файл SQL в MySQL

HTTP-запросы

Этот протокол описывает взаимодействие между двумя компьютерами (клиентом и сервером), построенное на базе сообщений, называемых запрос (Request) и ответ (Response).

Стартовые строки для запроса и ответа имеют различный формат — нам интересна только стартовая строка запроса, которая выглядит так: METHOD URI HTTP/VERSION, где METHOD — это метод HTTP-запроса, URI — идентификатор ресурса, VERSION — версия протокола (на данный момент актуальна версия 1.1).

Заголовки — это набор пар имя-значение, разделенных двоеточием.

Тело сообщения — это, собственно, передаваемые данные.

Пример Http-взаимодействия

Листинг запроса:

GET /index.php HTTP/1.1

Host: example.com

User-Agent: Mozilla/5.0 (X11; U; Linux i686; ru; rv:1.9b5) Gecko/2008050509 Firefox/3.0b5

Accept: text/html

Connection: close

Первая строка — это строка запроса, остальные — заголовки; тело сообщения отсутствует.

Листинг ответа:

HTTP/1.0 200 OK

Server: nginx/0.6.31

Content-Language: ru

Content-Type: text/html; charset=utf-8

Content-Length: 1234

Connection: close

После получаем Html-страницу.

Для разграничения действий с ресурсами на уровне HTTP-методов были придуманы следующие варианты:

- GET — получение ресурса

- POST — создание ресурса

- PUT — обновление ресурса

- DELETE — удаление ресурса

REST

REST (REpresentational State Transfer) — это термин был введен в 2000-м году Роем Филдингом (Roy Fielding) — одним из разработчиков протокола HTTP — в качестве названия группы принципов построения веб-приложений. Вообще REST охватывает более широкую область, нежели HTTP — его можно применять и в других сетях с другими протоколами. REST описывает принципы взаимодействия клиента и сервера, основанные на понятиях «ресурса» и «глагола» (можно понимать их как подлежащее и сказуемое). В случае HTTP ресурс определяется своим URI, а глагол — это HTTP-метод.

Для просмотра, нужно использовать метод GET. Метод GET в REST зарезервирован как «Посмотреть данные».

Листинг REST:

GET example.com/user?id=1

Для изменения, нужно использовать метод POST. Метод POST в REST зарезервирован как «Создать данные».

Листинг REST:

POST example.com/user?id=1&str=”hello”

Для изменения, нужно использовать метод DELETE. Метод DELETE в REST зарезервирован как «Удалить данные».

Листинг REST:

DELETE example.com/user?id=1

Для изменения, нужно использовать метод PUT. Метод PUT в REST зарезервирован как «Изменить данные».

Листинг REST:

PUT example.com/user?id=1&name=”vlad”

Json

Преимущества JSON:

- Простота использования;

- Упрощенный синтаксис обеспечивающий легкость прочтения кода.

Преимущества XML:

- XML поддерживается почти всеми языками программирования;

- XML расширяем.

Начало Json всегда начинается с фигурных скобок.

Литералы JSON:

- Строка;

- Число;

- Логическое значение;

- Значение null.

Листинг простого кода в Json:

{“key”:”Vlad”}

Ключ всегда заносится в скобки, чтобы дать тип string. После ключа открываются двоеточия для ссылки на результат самого ключа. Значение “key” будет “Vlad”, а точнее строка.

В массивах можно записывать и значения и объекты.

Листинг массива значений:

{“key”:[“str1”,”str2”]}

Листинг массива объектов:

{“key”:{“k1”:”Vlad”,“k2”:”Oleg”}}

Листинг большого кода:

{“k1”:[“k1.2”:{“k1.2.1”:”str1”,”k1.2.”:23},“k1.3”:[false,null]],“k2”:{“k2.1”:”str2”,“k2.2”:5,“k2.3”:true,“k2.4”:null}}

JavaSE8

Java - сильно типизированный объектно-ориентированный язык программирования, Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, с помощью виртуальной Java-машины. Дата официального выпуска - 23 мая 1995 года.

Создать простое приложение, которое выводит в консоли «Hello Java»

Листинг программы:

public class Main { // Класс

public static void main(String[] args){ //Функция

System.out.print("Hello Java");}} // Стандартный вывод текста на консоль

Создать функцию, которая выводит на консоль «Hello Java»

Листинг программы:

public class Main { // Класс

public static void main(String[] args){ //Функция

prt();} // Вызов функции

public static void prt(){ // Функция вывода текста на консоль

System.out.print("Hello Java");}// Стандартный вывод текста на консоль}

Создать объект в котором будет функция с выводом на консоль «Create new Class»

Листинг программы:

public class Main { // Главный класс

public static void main(String[] args){ // Главная функция программы

new C1();}} // Создание нового объекта

public class C1 { // Класс «C1»

public C1() { // Конструктор класса

prt();} // Вызов функции

public void prt(){ // Функция для вывода текста на консоль

System.out.print("Hello Java");}} // Стандартный вывод текста на консоль

Фреймворк Spark

Spark Framework - это простой и выразительный веб-фреймворк Java / Kotlin DSL, созданный для быстрого развития. Стремительное намерение - предоставить альтернативу разработчикам Kotlin / Java, которые хотят развивать свои веб-приложения как можно более выразительно и с минимальным шаблоном.

Для подключения фреймворка в gradle нужно добавить следующую строку: compile "com.sparkjava:spark-core:2.6.0"

Создать программу которая выводит слово “Spark”

Листинг Java:

import static spark.Spark.\*; //Импортируем фреймворк

public class Main { //Создаем класс

public static void main(String[] args){ //Создаем метод

port(80); //Задаем порт

get("/get", (request, response) -> "Spark");}} //Создаем get запрос с выводом текста

Создать программу в которой будет создаваться новое имя и новое “ID” для имени, вывести имя.

Листинг Java:

import java.util.ArrayList; //Импортируем класс коллекций

import static spark.Spark.\*; //Импортируем фреймворк

public class Main { //Создаем класс

private static ArrayList<String> str; //Создаем ссылку на коллекцию

public static void main(String[] args) { //Создаем метод

str = new ArrayList<>(); //Инициализация коллекции

port(80); //Задаем порт

get("/:id", (request, response) ->

str.get(Integer.parseInt(request.params(":id")))); //id имени

post("/add", (request, response) -> { //Создаем POST-запрос

str.add(request.queryParams("name")); //Добавляем в коллекцию новое имя

return "Success"; });}} //Вывод успешного добавления

JDBC – MySQL на JavaSE8

Для работы с JDBC нужно сделать БД в MySQL

Листинг SQL

CREATE SCHEMA `jdbcjava` DEFAULT CHARACTER SET utf8 ;

USE jdbcjava;

CREATE TABLE `jdbcjava`.`tab` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

Код создаёт новую БД с названием “jdbcjava”. В БД создается новая таблица с названием “tab” и с атрибутами “id” и “name”. БД имее кодировку UTF8, как и внутренняя таблица “tab”. Результат готовой программы показан на рисунке 12.

Для работы с JDBC нужно записать код в gradle.

Листинг gralde подключения библиотеки JDBC:

compile 'mysql:mysql-connector-java:5.1.6'

Листинг программы с подключением к БД “jdbcjava”:

import java.sql.\*; //Для рабоы с библиотекой JDBC

public class Main {

private static final String url =

"jdbc:mysql://localhost:3306/jdbcjava"; //Ссылка на БД

private static final String login = "root"; //Логин БД

private static final String pass = "root"; //Пароль БД

private static Statement statement; //Результат БД

public static void main (String[] args){

initialization(); //Инициализации БД

createName("Vlad"); //Создаем в таблице БД строку

createName("Vlad"); //Создаем в таблице БД строку

createName("Vlad"); //Создаем в таблице БД строку

createName("Vlad"); //Создаем в таблице БД строку

selectName(); //Метод для вывода данных на консоль

}private static void initialization(){

try { //Исключение

Connection jdbc = DriverManager.getConnection(url,login,pass); //Подкл. к БД

statement = jdbc.createStatement(); //Передаем управление с БД в Java

} catch (SQLException e) {e.printStackTrace();}}

private static void createName(String name){

try { //Исключение

statement.execute("insert into tab (name) values

('"+name+"')"); //Создаем новую строку в таблице

} catch (SQLException e) {e.printStackTrace();}}

private static void selectName(){

try { //Исключение

ResultSet resultSet = statement.executeQuery("select \* from tab"); //Помещаем опеределенную базу в результат, для дальнейшего вывода на консоль

while (resultSet.next()) { //Цикл, чтобы обойти всю таблицу

System.out.println(resultSet.getString("id")+" "+resultSet.getString("name")); //Выводим данные в консоль

}} catch (SQLException e) {e.printStackTrace();}}}

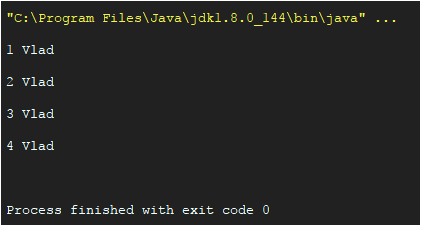


Рисунок 12 – Программа читает с БД

Закрепительный проект

Проект делается для закрепления знаний. Нужно создать веб-сервер, чтобы можно было взаимодействовать через REST-запросы с БД.

В БД две таблицы:

- Пользователи

- Подробно

В таблице “Пользователь” должны быть аргументы:

- Индивидуальный ключ пользователя

- Имя пользователя

- Фамилия пользователя

В таблице “Подробно” должны быть аргументы:

- Индивидуальный ключ пользователя

- Дата рождения пользователя

Ответ должен быть в виде json. Программа должна иметь кодировку utf8 для того, чтобы в программе можно было записываться различные символы.

Листинг MySQL:

CREATE SCHEMA `sqlcollegeweber` DEFAULT CHARACTER SET utf8 ;

USE sqlcollegeweber;

CREATE TABLE `sqlcollegeweber`.`users` (

`id` INT NOT NULL AUTO\_INCREMENT,

`first\_name` VARCHAR(45) NOT NULL,

`last\_name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

CREATE TABLE `sqlcollegeweber`.`info` (

`id` INT NOT NULL AUTO\_INCREMENT,

`age` INT NULL DEFAULT NULL,

`data` DATE NULL DEFAULT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

Создается новая БД с кодировкой utf8 и названием “sqlcollegeweber”. После создания БД передается указатель работы “sqlcollegeweber”. Создаются две таблицы. В-первой таблице три атрибута: id, first\_name, last\_name. Во-второй таблице три атрибута: id, age, data. Где age и data не обязательно заполнять.

В gradle были добавлены следующие строки:

compile 'mysql:mysql-connector-java:5.1.6' //jdbc

compile 'com.google.code.gson:gson:2.8.0' //gson

compile 'com.sparkjava:spark-core:2.6.0' //spark

Описание готовой программы.

GET:

Для просмотра всех пользователей в БД (Без подробного описания). Нужно сделать запрос http://localhost/users

Для просмотра одного конкретного пользователя в БД (С подробным описанием). Нужно сделать запрос http://localhost/users/{id} - где id - это любой id пользователя взятый из БД. Если был запрошен некорректный id пользователя то, тогда будет выведен текст с ошибкой.

POST:

Синтаксис для добавления новых пользователей:

http://localhost/add ?

& first = <Имя пользователя>

& last = <Фамилия пользователя>

& age = <Возраст пользователя>

& data = <Дата рождения пользователя>

Имеются ограничения по возрасту. Нельзя вносить возраст меньше 14 и более 100 лет. Обязательно нужно внести информацию об “Имени” и “Фамилии” пользователя.

DELETE:

Из БД можно удалить одного человека, не всех сразу. Сам метод http://localhost/users/{id} - где id - это любой id пользователя взятый из БД. Если был запрошен некорректный id пользователя то, тогда будет выведен текст с ошибкой.

PUT:

Синтаксис для изменения информации пользователя в БД:

[http://localhost/users/{id](http://localhost/users/%7bid)} ? - где id - это любой id пользователя взятый из БД

& first = <Имя пользователя>

& last = <Фамилия пользователя>

& age = <Возраст пользователя>

& data = <Дата рождения пользователя>

Можно изменить одну информацию или сразу несколько. Нельзя изменять информацию у всех пользователей (Или у выбранных по id) разом.

Листинг готовой программы:

package kz.kcollege;

import kz.kcollege.jdbc.JDBCDELETE;

import kz.kcollege.jdbc.JDBCGET;

import kz.kcollege.jdbc.JDBCPOST;

import kz.kcollege.jdbc.JDBCPUT;

import static spark.Spark.\*;

public class Main {

public static void main(String[] args) {

//Обращение начинается с http://localhost/

port(80);

//Синтаксис для подробного просмотра пользователя из БД

//http://localhost/users/{id} - где id - это любой id пользователя взятый из БД

get("/users/:id", (request, response) ->

new JDBCGET().print(true, Integer.parseInt(request.params(":id"))));

//Синтаксис для просмотра всех пользователей из БД

// http://localhost/users

get("/users", (request, response) ->

new JDBCGET().print(false));

//Синтаксис для добавления новых пользователей

// http://localhost/add ?

// & first = <Имя пользователя>

// & last = <Фамилия пользователя>

// & age = <Возраст пользователя>

// & data = <Дата рождения пользователя>

post("/add", (request, response) ->

new JDBCPOST().start(request));

//Синтаксис для удаления пользователей в БД

//http://localhost/users/{id} - где id - это любой id пользователя взятый из БД

delete("/users/:id", (request, response) ->

new JDBCDELETE().delete(Integer.parseInt(request.params(":id"))));

//Синтаксис для изменения информации пользователя в БД

//http://localhost/users/{id} - где id - это любой id пользователя взятый из БД

put("/users/:id", (request, response) ->

new JDBCPUT().update(request, Integer.parseInt(request.params(":id"))));}}

Все готовые файлы данного проекта можно посмотреть, ознакомиться с DOC и скачать по ссылке https://github.com/FromSi/FS\_CollegeSpark

4 Работа с базами данных

SQL «язык структурированных запросов» - язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

В начале 1970-х годов в одной из исследовательских лабораторий компании IBM была разработана экспериментальная реляционная СУБД IBM System R, для которой затем был создан специальный язык SEQUEL, позволявший относительно просто управлять данными в этой СУБД. Аббревиатура SEQUEL расшифровывалась как Structured English QUEry Language - «структурированный английский язык запросов». Позже по юридическим соображениям[3] язык SEQUEL был переименован в SQL.

SQL является, прежде всего, информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. SQL можно назвать языком программирования, при этом он не является тьюринг-полным, но вместе с тем стандарт языка спецификацией SQL/PSM предусматривает возможность его процедурных расширений.

Изначально SQL был основным способом работы пользователя с базой данных и позволял выполнять следующий набор операций:

- создание в базе данных новой таблицы;

- добавление в таблицу новых записей;

- изменение записей;

- удаление записей;

- выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);

- изменение структур таблиц.

Со временем SQL усложнился — обогатился новыми конструкциями, обеспечил возможность описания и управления новыми хранимыми объектами (например, индексы, представления, триггеры и хранимые процедуры) - и стал приобретать черты, свойственные языкам программирования.

При всех своих изменениях SQL остаётся единственным механизмом связи между прикладным программным обеспечением и базой данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

Язык SQL представляет собой совокупность операторов, инструкций, вычисляемых функций

Согласно общепринятому стилю программирования, операторы (и другие зарезервированные слова) в SQL обычно рекомендуется писать прописными буквами.

Операторы SQL делятся на:

1. CREATE создает объект БД (саму базу, таблицу, представление, пользователя и т. д.),
2. ALTER изменяет объект,
3. DROP удаляет объект;
4. операторы манипуляции данными (Data Manipulation Language, DML):
5. SELECT выбирает данные, удовлетворяющие заданным условиям,
6. INSERT добавляет новые данные,
7. UPDATE изменяет существующие данные,
8. DELETE удаляет данные;
9. операторы определения доступа к данным (Data Control Language, DCL):
10. GRANT предоставляет пользователю (группе) разрешения на определенные операции с объектом,
11. REVOKE отзывает ранее выданные разрешения,
12. DENY задает запрет, имеющий приоритет над разрешением;
13. операторы управления транзакциями (Transaction Control Language, TCL):
14. COMMIT применяет транзакцию,
15. ROLLBACK откатывает все изменения, сделанные в контексте текущей транзакции,
16. SAVEPOINT делит транзакцию на более мелкие участки.

Несмотря на наличие диалектов и различий в синтаксисе, в большинстве своём тексты SQL-запросов, содержащие DDL и DML, могут быть достаточно легко перенесены из одной СУБД в другую. Существуют системы, разработчики которых изначально ориентировались на применение по меньшей мере нескольких СУБД (например: система электронного документооборота Documentum может работать как с Oracle Database, так и с Microsoft SQL Server и DB2). Естественно, что при применении некоторых специфичных для реализации возможностей такой переносимости добиться уже очень трудно. Наличие стандартов и набора тестов для выявления совместимости и соответствия конкретной реализации SQL общепринятому стандарту только способствует «стабилизации» языка. Правда, стоит обратить внимание, что сам по себе стандарт местами чересчур формализован и раздут в размерах (например, базовая часть стандарта SQL:2003 состоит из более чем 1300 страниц текста). С помощью SQL программист описывает только то, какие данные нужно извлечь или модифицировать. То, каким образом это сделать, решает СУБД непосредственно при обработке SQL-запроса. Однако не стоит думать, что это полностью универсальный принцип — программист описывает набор данных для выборки или модификации, однако ему при этом полезно представлять, как СУБД будет разбирать текст его запроса. Чем сложнее сконструирован запрос, тем больше он допускает вариантов написания, различных по скорости выполнения, но одинаковых по итоговому набору данных. Хотя SQL и задумывался как средство работы конечного пользователя, в конце концов он стал настолько сложным, что превратился в инструмент программиста. Ранее диалекты SQL большинства СУБД не предлагали способа манипуляции древовидными структурами. Некоторые поставщики СУБД предлагали свои решения (например, в Oracle Database используется выражение CONNECT BY). В настоящее время в ANSI стандартизована рекурсивная конструкция WITH из диалекта SQL DB2. В Microsoft SQL Server рекурсивные запросы (Recursive Common Table Expressions) появились лишь в версии 2005.

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. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов Денвер, XAMPP, VertrigoServ. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

26 февраля 2008 года Sun Microsystems приобрела MySQL AB за 1 млрд долларов[6], 27 января 2010 года Oracle приобрела Sun Microsystems за 7,4 млрд долларов и включила MySQL в свою линейку СУБД.

Сообществом разработчиков MySQL созданы различные ответвления кода, такие как Drizzle (англ.), OurDelta, Percona Server и MariaDB. Все эти ответвления уже существовали на момент поглощениякомпании Sun корпорацией Oracle.

MySQL портирована на большое количество платформ: AIX, BSDi, FreeBSD, HP-UX, Linux, Mac OS X, NetBSD, OpenBSD, OS/2 Warp, SGI IRIX, Solaris, SunOS, SCO OpenServer, UnixWare, Tru64, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Windows Server 2003, WinCE, Windows Vista, Windows 7 и Windows 10. Существует также порт MySQL к OpenVMS. Важно отметить, что на официальном сайте СУБД для свободной загрузки предоставляются не только исходные коды, но и откомпилированные и оптимизированные под конкретные операционные системы готовые исполняемые модули СУБД MySQL.

При использовании Юникода UTF-8 этот оператор выглядит следующим образом: SET NAMES 'utf8'

Кодировка ISO 8859-5 не поддерживается.

Реляционная система управления базами данных (РСУБД), реже - система управления реляционными базами данных (СУРБД) - СУБД, управляющая реляционными базами данных.

Практически все разработчики современных приложений, предусматривающих связь с системами баз данных, ориентируются на реляционные СУБД:59. По данным аналитиков на 2010 год, реляционные СУБД используются в абсолютном большинстве крупных проектов по разработке информационных систем. По результатам исследований компании IDC 2009 года всего около 7% составляют проекты, в которых используются СУБД нереляционного типа.

По оценке Gartner в 2013 году рынок реляционных СУБД составлял 26 млрд долларов с годовым приростом около 9 процентов, а к 2018 году рынок реляционных СУБД достигнет 40 млрд долларов. В настоящее время абсолютными лидерами рынка СУБД являются компании Oracle, IBM и Microsoft, с общей совокупной долей рынка около 90%, поставляя такие системы как Oracle Database, IBM DB2 и Microsoft SQL Server.

Единственной коммерчески успешной СУБД российского производства является реляционная СУБД Линтер.

5 Работа в INTERNET

HTTP - широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам).

Аббревиатура HTTP расшифровывается как HyperText Transfer Protocol, «протокол передачи гипертекста». В соответствии со спецификацией OSI, HTTP является протоколом прикладного (верхнего, 7-го) уровня. Актуальная на данный момент версия протокола, HTTP 1.1, описана в спецификации RFC 2616.

Протокол HTTP предполагает использование клиент-серверной структуры передачи данных. Клиентское приложение формирует запрос и отправляет его на сервер, после чего серверное программное обеспечение обрабатывает данный запрос, формирует ответ и передаёт его обратно клиенту. После этого клиентское приложение может продолжить отправлять другие запросы, которые будут обработаны аналогичным образом.

Задача, которая традиционно решается с помощью протокола HTTP — обмен данными между пользовательским приложением, осуществляющим доступ к веб-ресурсам (обычно это веб-браузер) и веб-сервером. На данный момент именно благодаря протоколу HTTP обеспечивается работа Всемирной паутины.

Также HTTP часто используется как протокол передачи информации для других протоколов прикладного уровня, таких как SOAP, XML-RPC и WebDAV. В таком случае говорят, что протокол HTTP используется как «транспорт».

API многих программных продуктов также подразумевает использование HTTP для передачи данных — сами данные при этом могут иметь любой формат, например, XML или JSON.

Как правило, передача данных по протоколу HTTP осуществляется через TCP/IP-соединения. Серверное программное обеспечение при этом обычно использует TCP-порт 80 (и, если порт не указан явно, то обычно клиентское программное обеспечение по умолчанию использует именно 80-й порт для открываемых HTTP-соединений), хотя может использовать и любой другой.

Самый простой способ разобраться с протоколом HTTP — это попробовать обратиться к какому-нибудь веб-ресурсу вручную. Представьте, что вы браузер, и у вас есть пользователь, который очень хочет прочитать статьи Анатолия Ализара.

Предположим, что он ввёл в адресной строке http://alizar.from.ru/

Соответственно вам, как веб-браузеру, теперь необходимо подключиться к веб-серверу по адресу alizar.habrahabr.ru.

Для этого вы можете воспользоваться любой подходящей утилитой командной строки. Например, telnet: telnet alizar.from.ru 80

Сразу уточню, что если вы вдруг передумаете, то нажмите Ctrl + «]», и затем ввод — это позволит вам закрыть HTTP-соединение. Помимо telnet можете попробовать nc (или ncat) — по вкусу.

После того, как вы подключитесь к серверу, нужно отправить HTTP-запрос. Это, кстати, очень легко — HTTP-запросы могут состоять всего из двух строчек.

Для того, чтобы сформировать HTTP-запрос, необходимо составить стартовую строку, а также задать по крайней мере один заголовок — это заголовок Host, который является обязательным, и должен присутствовать в каждом запросе. Дело в том, что преобразование доменного имени в IP-адрес осуществляется на стороне клиента, и, соответственно, когда вы открываете TCP-соединение, то удалённый сервер не обладает никакой информацией о том, какой именно адрес использовался для соединения: это мог быть, например, адрес alizar.from.ru, from.ru или m.from.ru — и во всех этих случаях ответ может отличаться. Однако фактически сетевое соединение во всех случаях открывается с узлом 212.24.43.44, и даже если первоначально при открытии соединения был задан не этот IP-адрес, а какое-либо доменное имя, то сервер об этом никак не информируется — и именно поэтому этот адрес необходимо передать в заголовке Host.

Например (такая стартовая строка может указывать на то, что запрашивается главная страница сайта): GET / HTTP/1.1

Метод (в англоязычной тематической литературе используется слово method, а также иногда слово verb — «глагол») представляет собой последовательность из любых символов, кроме управляющих и разделителей, и определяет операцию, которую нужно осуществить с указанным ресурсом. Спецификация HTTP 1.1 не ограничивает количество разных методов, которые могут быть использованы, однако в целях соответствия общим стандартам и сохранения совместимости с максимально широким спектром программного обеспечения как правило используются лишь некоторые, наиболее стандартные методы, смысл которых однозначно раскрыт в спецификации протокола.

URI (Uniform Resource Identifier, унифицированный идентификатор ресурса) — путь до конкретного ресурса (например, документа), над которым необходимо осуществить операцию (например, в случае использования метода GET подразумевается получение ресурса). Некоторые запросы могут не относиться к какому-либо ресурсу, в этом случае вместо URI в стартовую строку может быть добавлена звёздочка (астериск, символ «\*»). Например, это может быть запрос, который относится к самому веб-серверу, а не какому-либо конкретному ресурсу. В этом случае стартовая строка может выглядеть так: OPTIONS \* HTTP/1.1

Версия определяет, в соответствии с какой версией стандарта HTTP составлен запрос. Указывается как два числа, разделённых точкой (например 1.1).

Для того, чтобы обратиться к веб-странице по определённому адресу (в данном случае путь к ресурсу — это «/»), нам следует отправить следующий запрос: GET / HTTP/1.1 Host: alizar.from.ru

При этом учитывайте, что для переноса строки следует использовать символ возврата каретки (Carriage Return), за которым следует символ перевода строки (Line Feed). После объявления последнего заголовка последовательность символов для переноса строки добавляется дважды.

Впрочем, в спецификации HTTP рекомендуется программировать HTTP-сервер таким образом, чтобы при обработке запросов в качестве межстрочного разделителя воспринимался символ LF, а предшествующий символ CR, при наличии такового, игнорировался. Соответственно, на практике большая часть серверов корректно обработает и такой запрос, где заголовки отделены символом LF, и он же дважды добавлен после объявления последнего заголовка.

6 Работа с мультимедиа

Лабораторная работа №1

Тема: «Интерфейс программы Flash»

Цель работы: изучение инструментов и интерфейса программы Flash для приобретения навыков в работе с программой Flash.

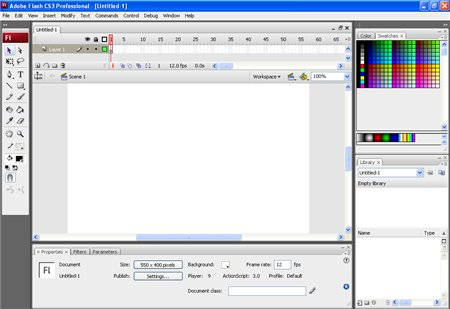


Рисунок 13 – Главный интерфейс Flash

Контрольные вопросы:

1. Для чего предназначена программа Flash?

2. Какую работу выполняет временная шкала программы Flash?

3. Для чего предназначены плавающие панели программы Flash?

4. Какие параметры имеет Flash-фильм?

Ответы на контрольные вопросы:

1. Macromedia Flash— мультимедийная платформа компании Adobe Systems для создания веб-приложений или мультимедийных презентаций. Широко используется для создания рекламных баннеров, анимации, игр, а также воспроизведения на веб-страницах видео- и аудиозаписей.

2. Временная шкала - основной инструмент при работе с анимацией во Flash. На ней отображается информация о слоях, о том какие кадры являются ключевыми, а какие генерирует Flash. С помощью временной шкалы можно понять, какие кадры содержат действия или метки. Она позволяет перемещать ключевые кадры и целые куски анимации.

3. Плавающие панели программы Macromedia Flash позволяют просмотреть, организовать и изменить элементы ролика, а также с помощью этих панелей можно изменить свойства выделенных элементов.

4. По умолчанию фильм имеет размер 550x400 пикселов, за вычетом текущей границы.

Лабораторная работа №2

Тема: «Покадровая анимация графических объектов посредством програмы Flash.»

Цель работы: закрепить теоретический материал на тему «покадровая анимация», а также приобрести практические навыки в работе с покадровой анимацией графических объектов.

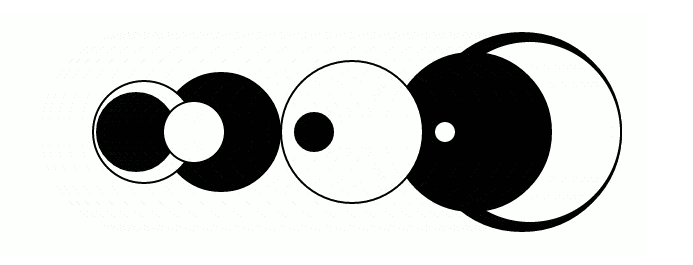


Рисунок 14 – Покадровая анимация

Контрольные вопросы:

1. Дать определение понятия анимации?

2. Дать определение понятия кадра, что такое ключевой кадр? И как его можно создать?

3. Что вы понимаете под понятием покадровая анимация?

4. Назовите преимущества и недостатки покадровой анимации.

Ответы на контрольные вопросы:

1. Анимация- вид мультипликации, создаваемый при помощи компьютера. В отличие от более общего понятия «графика CGI», относящегося как к неподвижным, так и к движущимся изображениям, компьютерная анимация подразумевает только движущиеся.

2. Кадр - фрагмент кино- или видеоряда, отдельное изображение.

Ключевой кадр - это разновидность кадра, в котором происходит что-то новое .Основным инструментом при создании покадровой анимации является панель временной диаграммы. С ее помощью вы можете создавать, удалять и перемещать кадры анимации, изменять режимы просмотра отдельных кадров и всей сцены, выполнять другие операции.

3. Покадровая анимация - представляет собой набор кадров, хранящихся как отдельные изображения и сменяющих друг друга с большой скоростью.

4. Преимущества:

- относительная очевидность создания. В самом деле, для того чтобы изготовить анимационный фильм, нужно всего лишь нарисовать все входящие в него кадры и перевести их на какой-нибудь информационный носитель;

- широкие возможности для творчества.

Недостатки:

- большая трудоемкость создания фильмов. В случае если каждый кадр рисуется вручную и при этом не применяются никакие технические средства, облегчающие работу, процесс создания фильма может затянуться на многие месяцы, а то и годы;

- большие проблемы, возникающие при сохранении покадровой анимации в цифровом виде.

Лабораторная работа №3

Тема: «Автоматическая анимация движения и изменения формы графических объектов»

Цель работы: научиться использовать средства программы Flash 5.0 для выполнения автоматической анимации движения графических объектов, а также автоматической анимации изменения формы графических объектов.



Рисунок 15 - Яхта

Контрольные вопросы:

1. Какие виды автоматической анимации вы знаете?

2. Что такое слой траектории. Как он создается? Какие еще слои вы знаете?

Ответы на контрольные вопросы:

1. Покадровая анимация, анимация, созданная с помощью операции автоматического заполнения промежуточных кадров, эффекты временной шкалы, создание анимации при помощи сценариев на языке ActionScript, в который.

2. По слою траектории направляется объект. Кликаем по слою с мячом и добавляем новый слой-путеводитель. Линия.

Лабораторная работа №4

Тема: «Создание и использования библиотеки символов»

Цель работы: научиться создавать графические символы, анимированные графические символы, клипы; создавать и использовать библиотеку символов.

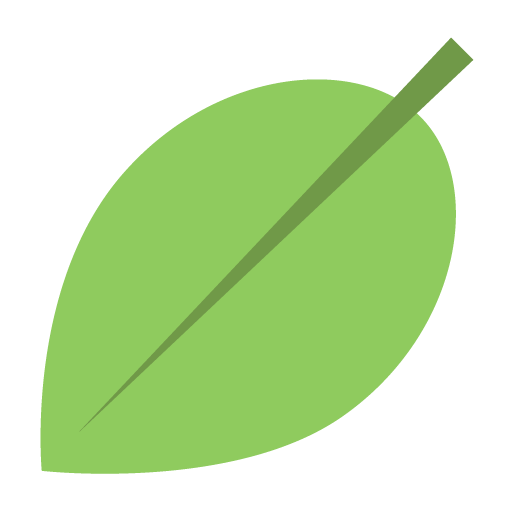


Рисунок 16 — Символ

Контрольные вопросы:

1. Какие символы Flash вы знаете?

2. Что такое библиотека символов Flash для чего она создается?

3. Какие режимы имеет библиотека символов?

4. Чем отличается статический графический символ от анимированного графического символа?

Ответы на контрольные вопросы:

1. Графические символы, символы-кнопки, символы-мувиклипы, шрифтовые символы.

2. Применение символов уменьшает размер файла, так как независимо от того, сколько образцов создано.

3. Создание, подключение, редактирование, копирование, экземпляры, удаление.

4. Статический мало весит и не воспроизводит движение.

Лабораторная работа №5

Тема: «Анимация сложных сцен»

Цель работы: научиться создавать и использовать несколько слоев для анимации сложных сцен.



Рисунок 17 — Движущиеся объекты

Контрольные вопросы:

1. Чем сложная сцена отличается от простой?

2. Определение обычного слоя.

3. Определение слоя маски.

4. Определение направляющего слоя.

5. Для чего необходимо использовать несколько слоев?

Ответы на контрольные вопросы:

1. В сложной сцене имеются дополнительные возможности работы с масками и слоями, а так же имеется оптимизация.

2. Слои позволяют упорядочить графические объекты в документе. Рисование и редактирование объектов в одном слое не влияет на объекты, находящиеся в другом слое. Сквозь участки рабочей области, где на слое ничего нет, виден нижележащий слой.

3. Маска представляет собой слой, в котором любая область заливки интерпретируется как отверстие, через которое виден нижележащий слой. При этом цвет заливки, градиент и контур игнорируются. По умолчанию слой-маска закрывает (маскирует) только тот слой, который лежит непосредственно под ним.

4. В направляющем слое рисуется траектория, по которой должен двигаться объект, затем к ней привязывается сам объект, а направляющий слой вместе с траекторией делается невидимым

5. Чтобы легче было различить, какому слою принадлежат объекты

Лабораторная работа №6

Тема: «Маскирование слоев и анимация маски»

Цель работы: научиться создавать и использовать слои масок, производить их анимацию.



Рисунок 18 — Картина с маской

Контрольные вопросы:

1. Что представляет собой слой маски?

2. Для чего используются слои маски?

Ответы на контрольные вопросы:

1. Слой-маска в программе Macromedia Flash Professional 8 применяется в flash-ролике для создания эффекта замочной скважины.

2. Использование слоя маски (mask layer) позволяет закрывать часть слоя (или слоев), который находится под слоем маски.

Лабораторная работа №7

Тема: «Интерактивная анимация»

Цель работы: научиться создавать интерактивные элементы и применять их к анимационным фильмам.

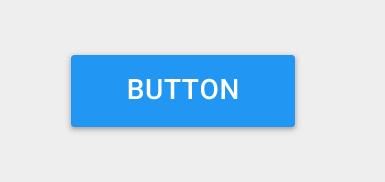


Рисунок 19 - Кнопка

Контрольные вопросы:

1. Что вы понимаете под элементом управления?

2. Для чего необходимы элементы управления?

3. Структура элемента управления.

4. Назначение встроенного в Flash языка ActionScript.

Ответы на контрольные вопросы:

1. Элемент управления. Если элемент управления определен в шаблоне диалогового окна, автоматическое создание диалоговое окно при создании. (Необходимо иметь член CAnimateCtrl в классе диалогового окна, соответствующий элементу управления анимацией). Также можно использовать функцию-член Создать создает элемент управления, как дочернее окно любого окна.

2. Использование векторной графики как графического режима по умолчанию делает Flash незаменимым средством разработки для Web. Векторная графика – это объекты, определяемыми математическими уравнениями, или векторами, которые содержат информацию о размере, форме, цвете, границе и местоположении.

3. Основные принципы работы с Macromedia Flash структура программы использование векторной графики передача данных в потоковом режиме в Macromedia Flash разделение данных и их представление.

4. ActionScript — это язык программирования, посредством которого во Flash осуществляется отправка команд и запросов о временных зависимостях, видеоклипах, кнопках и других объектах. Очень часто с помощью ActionScript можно легко достичь целей, которые в противном случае представляли бы собой труднодостижимую или, вообще, невозможную задачу.

Заключение

В процессе практики, усовершенствовал навыки работы с компьютером, научился эффективно использовать советы наставника. В работе они помогали найти ответы на возникающие в процессе деятельности вопросы, такие как что нужно для того, что бы компьютер хорошо работал. Во время прохождения производственной практике поручалось собирать и чистить компьютеры. Опыт сборки и чистки компьютера необходим каждому программисту.

В ходе прохождения практики ознакомился с:

- IntelliJ IDEA Community Edition;

- Postman;

- MySQL;

- JavaSE8;

- Json;

- SQL;

- JDBC MySQL;

- Spark;

- HTTP;

- REST.

В процессе прохождения практики мне пришлось, в частности, столкнуться с переустановки компьютера. В ходе выполнения данной работы пришлось консультироваться с наставником, глубоко изучить БД. Понравилось работать с «JavaSE8», узнал, как на одном сервере можно совместить БД и «Json». За период прохождения практики пришлось дважды заниматься переустановкой ОС. В связи с этим были получены практические навыки.

Список литературы

1. Нотон П. JAVA:Справ.руководство Пер.с англ./Под ред.А.Тихонова.-М.:БИНОМ:Восточ.Кн.Компания,1996:Восточ.Кн.Компания.-447с..-(Club Computer)
2. Патрик Нотон, Герберт Шилдт Полный справочник по Java.- McGraw-Hill,1997, Издательство "Диалектика",1997
3. Дэвид Флэнэген Java in a Nutshell.- O'Reilly & Associates, Inc., 1997, Издательская группа BHV, Киев, 1998
4. Ренеган Э.Дж.(мл.) 1001 адрес WEB для программистов Новейший путеводитель программиста по ресурсам World Wide Web:Пер.с англ..-Минск:Попурри,1997.-512с.ил.
5. Сокольский М.В. Все об Intranet и Internet .-М.:Элиот,1998.-254с.ил.
6. Чен М.С. и др. Программирование на JAVA:1001 совет:Наиболее полное руководство по Java и Visual J++:Пер.с англ./Чен М.С.,Грифис С.В.,Изи Э.Ф..-Минск:Попурри,1997.-640с.ил.+ Прил.(1диск.)
7. Майкл Эферган Java: справочник.- QUE Corporation, 1997, Издательство "Питер Ком", 1998
8. Джо Вебер Технология Java в подлиннике.- QUE Corporation, 1996, "BHV-Санкт-Петербург",1997
9. Джейсон Мейнджер Java: Основы программирования.- McGraw-Hill,Inc.,1996, Издательская группа BHV, Киев,1997
10. И.Ю.Баженова Язык программирования Java.- АО "Диалог-МИФИ", 1997
11. Майкл Томас, Пратик Пател, Алан Хадсон, Доналд Болл(мл.) Секреты программирования для Internet на Java .- Ventana Press, Ventana Communications Group, U.S.A.,1996, Издательство "Питер Пресс", 1997
12. Кен Арнольд, Джеймс Гослинг Язык программирования Java.- Addison-Wesley Longman,U.S.A.,1996, Издательство "Питер-Пресс", 1997