[Тема 1. Проектирование 3-х уровневой архитектуры. 3](#_Toc410253503)

[1.1 Технологии JEE 4](#_Toc410253504)

[1.2 Шаблоны J2EE 5](#_Toc410253505)

[1.3 Приоритеты в Business Web Application 6](#_Toc410253506)

[1.4 Распределенная архитектура Веб-приложений. 6](#_Toc410253507)

[Задание 8](#_Toc410253508)

[Тема 2.Основы Apache Maven 9](#_Toc410253509)

[2.1 Настройка Apache Maven для Windows 9](#_Toc410253510)

[Задание 1 11](#_Toc410253511)

[2.2 Создание нового проекта 11](#_Toc410253512)

[Задание 2 13](#_Toc410253513)

[2.3 Компилирование и тестирование проекта 13](#_Toc410253514)

[Задание 3 15](#_Toc410253515)

[2.4 Описание Объектной модели проекта (POM) 16](#_Toc410253516)

[2.5 Описание жизненного цикла сборки. 17](#_Toc410253517)

[2.6 Описание профайлов сборки. 18](#_Toc410253518)

[Задание 4 19](#_Toc410253519)

[Тема 3.Инженерные техники при применении Apache Maven. 20](#_Toc410253520)

[3.1 Автоматизация сборки 20](#_Toc410253521)

[3.2 Модульность проекта 22](#_Toc410253522)

[Задание 5 24](#_Toc410253523)

[3.3 Управление зависимостями 24](#_Toc410253524)

[Задание 6 28](#_Toc410253525)

[3.4 Автоматизация установки проекта. 28](#_Toc410253526)

[3.5 Сборка web-приложения. 31](#_Toc410253527)

[Задание 7 33](#_Toc410253528)

[Тема 4. Apache Tomcat Сервер 34](#_Toc410253529)

[4.1 Описание Apache Tomcat. 34](#_Toc410253530)

[4.2 Архитектура Tomcat 34](#_Toc410253531)

[4.3 Установка и конфигурирование Tomcat 37](#_Toc410253532)

[4.4 Тестирование установки сервера Tomcat 39](#_Toc410253533)

[4.5 Структура директорий сервера Tomcat 40](#_Toc410253534)

[Задание 8 41](#_Toc410253535)

[4.6 Выполнение скриптов сервера Tomcat 42](#_Toc410253536)

[4.7 Передача конфигурационных опций времени выполнения скриптам Catalina. 43](#_Toc410253537)

[Тема 5. Конфигурирование сервера Tomcat 44](#_Toc410253538)

[5.1 Установка WAR-архива 44](#_Toc410253539)

[Задание 9 46](#_Toc410253540)

[5.2 Конфигурация хостов 46](#_Toc410253541)

[6.3 Конфигурация контекстов 47](#_Toc410253542)

[5.4 Менеджер веб-приложений Tomcat 48](#_Toc410253543)

[5.5 Получение доступа к Менеджеру веб-приложений 49](#_Toc410253544)

[5.6 Доступ к менеджеру веб-приложений, используя веб-интерфейс. 51](#_Toc410253545)

[5.7 Прослушивание установленных веб-приложений 52](#_Toc410253546)

[Задание 10 52](#_Toc410253547)

[5.8 Установка нового веб-приложения 52](#_Toc410253548)

[Задание 12 53](#_Toc410253549)

[5.9 Перезагрузка, остановка и запуск существующего веб-приложения 54](#_Toc410253550)

[5.10 Управление сессиями веб-приложения 55](#_Toc410253551)

[5.11 Tomcat Maven plugin. 56](#_Toc410253552)

[Задание 11 57](#_Toc410253553)

[тема 6. Основы Subversion 58](#_Toc410253554)

[6.1 Основные понятия управления версиями 58](#_Toc410253555)

[6.2 Установка сервера subversion 60](#_Toc410253556)

[Задание 12 60](#_Toc410253557)

[6.3 Установка клиента subversion и его использование 61](#_Toc410253558)

[6.4 Сохранение своих изменений в репозиторий 63](#_Toc410253559)

[Задание 13 64](#_Toc410253560)

[Задание 14 64](#_Toc410253561)

[6.5 Работа с репозиторием напрямую на сервере 64](#_Toc410253562)

[Задание 15 65](#_Toc410253563)

[6.6 Откат изменений в рабочей копии 66](#_Toc410253564)

[Задание 16 66](#_Toc410253565)

[6.7 Работа с утилитой LogViewer 67](#_Toc410253566)

**Тема 1. Проектирование 3-х уровневой архитектуры.**

Наиболее интересными и полезными качествами, которыми привлекала технология JSP, можно считать:

* **удобный способ объединение Server-Side Script c HTML;**
* **скриптовый  подход (интерпретируемый язык)** - т.е. файл с исходным кодом JSP одновременно является его исполняемым файлом, что упрощает процессы разработки и поддержки;
* **концепция "Session"**- переменные для каждого пользовательского соединения, как удачное решение вечной проблемы stateless-протокола HTTP;
* **удобный набор объектов-утилит:** Request, Response, Session, Context.

Проблемы, присущие плохо структурированным JSP-проектам:

* Смесь бизнес-кода и HTML приводит к трудностям поддержки и того и другого;
* Наличие большого количества DB-зависимого кода в JSP-страницах привязывает их к источнику данных;
* Перегруженность JSP-страниц функциональностью приводит к перегрузкам IIS (хотя это можно решить кластеризацией IIS);
* Смысловая перегрузка JSP-страниц затрудняет их поддержку;
* Хранение бизнес-логики в JSP-страницах в "размазанном" виде приводит к затруднению ее вынесения в объекты 2-nd tier (при необходимости масштабирования и поддержки разных видов 1st tier-клиентов);
* Полная зависимость кода проекта от самой технологии JSP

***Что предлагается делать:***

1. Вынести HTML из JSP-страниц в отдельные файлы;
2. Вынести SQL из JSP-страниц;
3. Абстрагировать JSP-специфические возможности в объекты общей библиотеки;
4. Организовать все часто используемые функции в виде методов общей объектно-ориентированной библиотеки;
5. Использовать JavaScript и отслеживать пути быстрого перехода на JSP/PHP, при возникновении подобной необходимости.

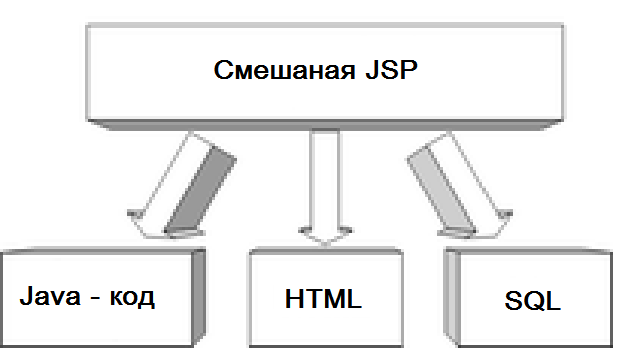


Рисунок 1.1.1 Разбиение представления на части

1.1 Технологии JEE

На текущий момент платформа JEE включает в себя технологии представленные в таблице 1

Таблица 1 Технологии JEE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Описание** | **Использованная версия** | | | |
| **J2EE 1.4** | **JEE 5** | **JEE 6** | **JEE 7** |
| **2003** | **2006** | **2009** | **2013** |
| EJB | Enterprise JavaBeans — спецификация технологии серверных компонентов, содержащих бизнес-логику | 2.1 | 3.0 | 3.1 | 3.2 |
| JPA | Java Persistence API | Нет | 1.0 | 2.0 | 2.1 |
| Сервлет | Обслуживание запросов веб-клиентов | 2.4 | 2.5 | 3.0 | 3.1 |
| JSP | JavaServer Pages — динамическая генерация веб-страниц на стороне сервера | 2.0 | 2.1 | 2.2 | 2.3 |
| JSTL | JavaServer Pages Standard Tag Library | Нет | 1.2 | 1.2 | 1.2 |
| JSF | JavaServer Faces — компонентный серверный фреймворк для разработки веб-приложений на технологии Java | Нет | 1.2 | 2.0 | 2.2 |
| JAX-WS | Java API for XML Web Services — создание веб-сервисов | 1.0 | 1.2 | 1.2 | 2.2 |
| JAX-RS | Java API for RESTful Web Services -- создание RESTful веб-сервисов | Нет | Нет | 1.1 | 2.0 |
| JNDI | Java Naming and Directory Interface — служба каталогов | 1.2 | 1.2 | 1.2 |  |
| JMS | Java Message Service — обмен сообщениями | 1.1 | 1.1 | 1.1 | 2.0 |
| JTA | Java Transaction API | 1.0.1B | 1.1 | 1.1 | 1.2 |
| JAAS | Java Authentication and Authorization Service — Java реализация PAM | 1.0 | 1.0 | 1.0 |  |
| JavaMail | Получение и отправка электронной почты | 1.2 | 1.4 | 1.4 | 1.5 |
| JACC | Java Authorization Contract for Containers | 1.0 | 1.1 | 1.1 | 1.4 |
| JCA | J2EE Connector Architecture | 1.5 | 1.5 | 1.6 | 1.6 |
| JAF | JavaBeans Activation Framework | 1.0 | 1.1 | 1.1 | 1.1 |
| StAX | Streaming API for XML | Нет | 1.0 | 1.0 | 1.0 |
| CDI | Context and Dependency Injection | Нет | Нет | 1.0 | 1.1 |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

1.2 Шаблоны J2EE

**Шаблоны J2EE** — набор шаблонов проектирования, описывающих архитектуру серверной платформы для задач средних и крупных предприятий. Эти шаблоны рекомендуется применять при проектировании приложений с использованием java-архитектуры J2EE.

Список шаблонов привиден ниже:

**Intercepting Filter** - Обеспечивает централизованную точку входа для управления обработкой запроса.

**Front Controller** - Комбинирует Dispatcher, Front Controller и View Helper, откладывая обработку сигналов.

**Dispatcher View** - шаблон описывает общую комбинацию контроллера и диспетчера с видами и помощниками.

**Service to Worker** описывает общую комбинацию контроллера и диспетчера с видами и помощниками.

**Composite View** - Создание составного визуального представления

**View Helper** - Обеспечивает предварительную и пост-обработку запроса.

**Business Delegate** - Прячет сложности поиска и создания бизнес-сервисов.

**Service Locator** - Управляет исполнением запросов, кэшированием результатов и их обработкой.

**Value List Handler** - Собирает составной Value Object из многих источников данных.

**Value Object Assembler** - Прячет сложность бизнес-объекта, централизует обработку workflow.

**Composite Entity** - Обеспечивает обмен данными между слоями, уменьшая сетевой трафик.

**Value Object** - Прячет сложность бизнес-объекта, централизует обработку workflow.

**Session Facade** - Разделяет презентационный и сервисный уровни, обеспечивает интерфейсы фасада и посредника для сервисов.

**Data Access Object** - Абстрагирует источник данных; обеспечивает прозрачный доступ к данным.

**Service Activator** - Обеспечивает асинхронную обработку для компонентов EJB.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

1.3 Приоритеты в Business Web Application

Предлагаемая, но не единственно верная, иерархия такова:

1. Функциональная адекватность - Приложение корректно выполняет свои функции в идеальных условиях (один пользователь в единицу времени, идеальное функционирование аппаратуры и т.д.).

2. Надежность в условиях ограниченной загрузки - Приложение устойчиво и надежно обслуживает небольшое фиксированное число одновременно работающих пользователей. (Типичные проблемы Интернет, такие, например, как сбой соединения, не нарушают работу.).

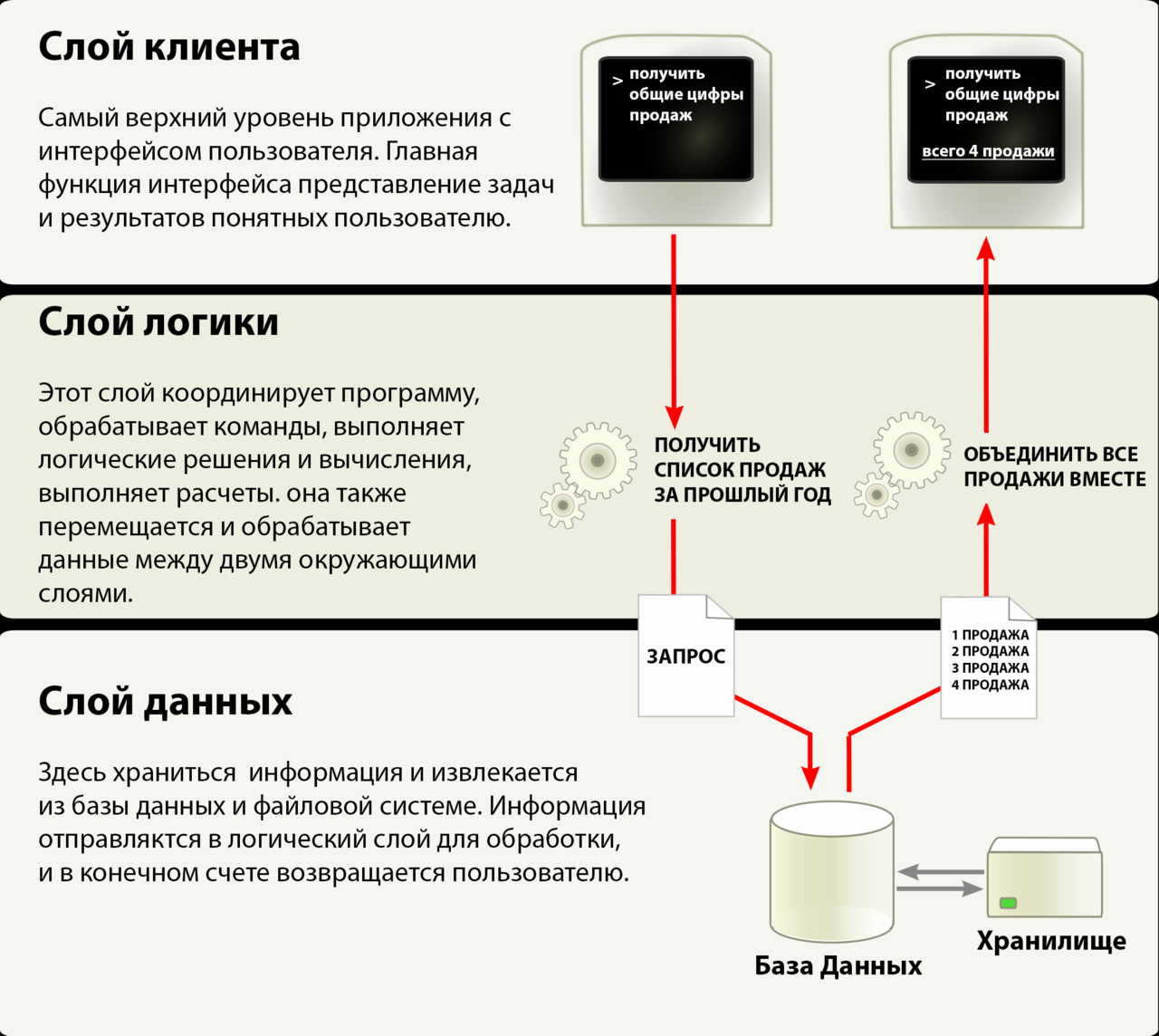
3. Удобство работы, интуитивно понятный интерфейс - пользователям удобно работать с системой;

4. Возможность обслужить максимальное расчетное количество одновременно работающих пользователей - система готова надежно обслужить всех потенциальных пользователей;

5. Высокая скорость работы - короткое время отклика системы;

6. Красивый дизайн - дизайн системы эстетически приятен большинству пользователей.

1.4 Распределенная архитектура Веб-приложений.

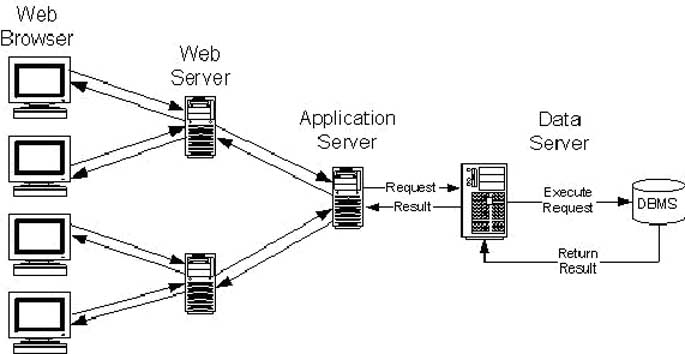


Интерпритация 3-х уровневой архитектуры

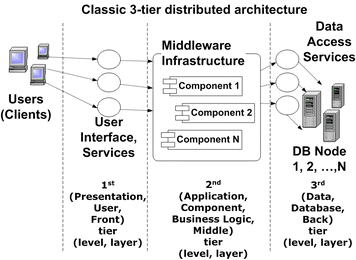
[Клиент](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_%28%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29) (слой клиента) — это интерфейсный (обычно [графический](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F)) компонент комплекса, предоставляемый конечному пользователю. Этот уровень не должен иметь прямых связей с базой данных (по требованиям безопасности и масштабируемости), быть нагруженным основной [бизнес-логикой](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0) (по требованиям [масштабируемости](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%88%D1%82%D0%B0%D0%B1%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D0%BE%D1%81%D1%82%D1%8C)) и хранить состояние приложения (по требованиям надёжности).

[Сервер приложений](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) (средний слой, связующий слой) располагается на втором уровне, на нём сосредоточена бо́льшая часть бизнес-логики. Вне его остаются только фрагменты, экспортируемые на клиента (терминалы), а также элементы логики, погруженные в базу данных (хранимые процедуры и триггеры).

Сервер баз данных (слой данных) обеспечивает хранение данных и выносится на отдельный уровень, реализуется, как правило, средствами [систем управления базами данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), подключение к этому компоненту обеспечивается только с уровня сервера приложений.



Интерпритация 3-х уровневой архитектуры



Интерпритация 3-х уровневой архитектуры

В этой схеме собраны почти все варианты названий трех уровней распределенного приложения. Чтобы избежать путаницы, будем именовать уровни так: 1st tier, 2nd tier и 3rd tier, по причине краткости написания.

Три уровня (с позиции программирования) - это **хранение**, **обработка** и **представление** информации. Идея заключается в том, чтобы не смешивать эти три составляющие. 3-х уровневый подход - это просто хороший стиль программирования. Его можно применять при разработке любых приложений. Новизна же и идея распределенных приложений в том, чтобы иметь возможность распределить эти три уровня физически на различных компьютерах, а также возможность иметь несколько взаимозаменяемых вариантов каждого уровня.

**Основные критерии идеальной распределенной архитектуры:**

**1. Каждый уровень распределенного приложения может взаимодействовать только со смежным уровнем.**

Это означает, что: 1st tier не должен иметь прямого доступа к 3rd tier и наоборот; 2nd и 3rd tiers не должны иметь прямого интерфейса с пользователем; обращение к источнику данных из 1st tier происходит только через объекты 2-nd tier;

**2. Вся сложная бизнес-логика находится в объектах 2nd tier.**

Как уже упоминалось, это условие часто нарушается, снижая возможность масштабирования. Хотя подобные решения иногда оправданы.

**3. Взаимодействие уровней организовано так, чтобы они могли взаимодействовать по сети, находясь физически на различных компьютерах.**

Это означает, что распределенная архитектура не зависит от способа развертывания (deployment) приложения - все уровни могут быть размещены физически как одном компьютере, так и на разных, в условиях заданной сетевой структуры;

**4. Сущности данных независимы от способа их хранения, уникально идентифицируемы по какому-либо ключу, и независимы от способа и места хранения других сущностей.**

Это, в частности, означает отсутствие в некоторых таблицах связей по внешним ключам, что позволяет хранить группы таблиц, имеющих отношение к одной сущности, в разных базах данных. Однако тогда необходимо отслеживать целостность данных дополнительным кодом.

Проблема JSP заключается в том, что смесь java-кода бизнес-логики, HTML и SQL представляет собой 2-х уровневую архитектуру (клиент-сервер), которая с задачами Web-приложений не справляется. Поэтому необходимо разделить JSP на три составляющие - JSP-код с бизнес-логикой, HTML и SQL.

Задание

Определите интересующую вас предметную область контрольной работы №1. Выделите сущности и детализируйте их. Разработайте архитектуру проекта для выбраной предметной области.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

**Тема 2.Основы Apache Maven**

2.1 Настройка Apache Maven для Windows

ApacheMaven – это программный продукт, предназначенный для упрощения сборки проектов, пришедший на смену ApacheAnt.Используя огромный потенциал Mavenа и его эффективность в решении ежедневных проблем, он стал очень популярным и широко используемым инструментом не только разработчиками, но и другими членами команд.

Официальный сайт Mavena <http://maven.apache.org>, основные его достоинства следующие:

* Делает процессы сборки приложения простым
* Предоставляет унифицированную систему сборки
* Предоставляет информацию о качестве проекта
* Предоставляет описание лучших практик в разработке
* Позволяет производить прозрачную и понятную миграцию новых компонентов.

***Maven*** – это программный продукт с интерфейсом командной строки и должен быть интегрирован с переменными среды Windows. Процесс установки достаточно прост и требует установленной Java SDK на компьютере.

Для проверки, установлена ли JavaSDK на компьютере, выполните следующую команду через командную строку Windows:

Java-version

*Результат может быть похожий на следующее:*

java version "1.6.0\_21"

Java(TM) SE Runtime Environment (build 1.6.0\_21-b07)

Java HotSpot(TM) Client VM (build 17.0-b17, mixed mode)

Если у вас не установлен **Java SDK**, тогда вы можете скачать последнюю версию на сайте http://www.oracle.com/technetwork/java/javase/downloads/index.html и установить.

|  |  |
| --- | --- |
|  | |
|  | |
|  | |
| **{?}** | Установлен Java SDK? |

Тогда пора установить ***apachemaven***. Скачать последнюю версию можно с сайта <http://maven.apache.org/download.html>.

После завершения скачивания Maven, распакуйте архив в любую папку. Для Windows, как правило, путь к папке не должен содержать пробелов. Я распаковал Maven на диск D: d:\opt\maven .

Для начала использования Maven необходимо настроить переменные среды Windows.**M2\_HOME** переменная должна быть установлена и **PATH** переменная должна быть модифицирована для включения папки, откуда запускается Maven.

Установить переменные среды вызовом «Настройки Системы» (System), используя «Панель Управления» (ControlPanel).

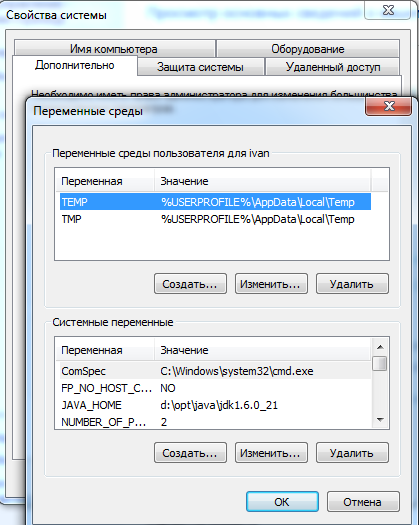


Рисунок 2.1.1Путь к изменению переменных среды.

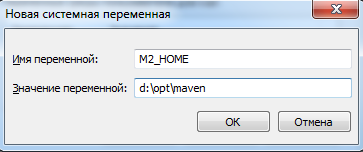


Рисунок 2.1.2 Установка переменной M2\_HOME

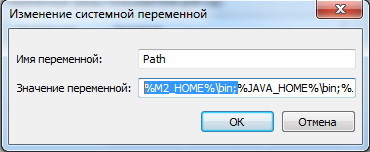


Рисунок 2.1.3 Изменение переменной PATH.

Создайте новую переменную среды M2\_HOME, указывающую путь к папке, куда был распакован Maven. Для моего компьютера значение для M2\_HOME указывает на **d:\opt\maven**.Переменная среды Path должна уже существовать. Она должна быть изменена, добавлением следующего текста в начало:

%M2\_HOME%\bin;

Apache Maven, теперь, готов к использованию. Он, также, доступен для интеграции с IDE и другими программными средствами, предназначенными для разработки.

Прежде чем использовать **Apache Maven**, вам необходимо **проверить,** как проверить установлен ли maven на вашем компьютере?

Это можно сделать с помощью следующей команды:

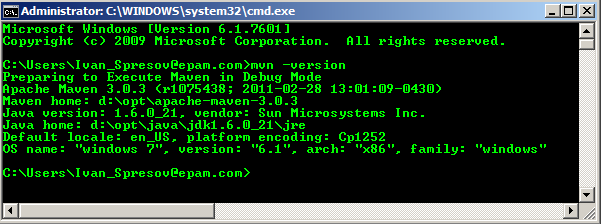


Рисунок 2.1.4. Проверка версии maven

Если результат похожий напредставленный выше, тогда вы можете быть уверенными, что maven установлен и готов к использованию. Если же ваша операционная система не может найти команду ***mvn***, тогда проверьте, что переменные среды ***path*** и ***M2\_HOME*** установлены корректно.

Задание 1

Возьмите установочный файл maven. Настройте environment variables и проверьте, что maven настроен корректно. Напишите версию Maven.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

2.2 Создание нового проекта

После того, как вы установили apachemaven, вам необходимо создать новый проект. Для этого необходимо сделать следующее:

1. Запустите консоль (cmd в windows).
2. Запустите следующую команду:

mvn archetype:generate

1. Если это первый раз, когда вы запускаете эту команду, вы увидите статус скачивания в командной строке.
2. Затем вы увидите большой список архетипов, у каждого есть номер, имя, и короткое описание, описывающие, что каждый из них представляет. Выберем архетип по умолчанию. Мы предполагаем, что это архетип номер 513, с именем maven-archetype-quickstart. Список со временем может меняться.

[INFO] No archetype defined. Using maven-archetype-quickstart

(org.apache.maven.archetypes:maven-archetype-quickstart:1.0)

Choose archetype:

1: remote -> docbkx-quickstart-archetype (-)

….

100: remote -> maven-archetype-profiles (-)

101: remote -> maven-archetype-quickstart (An archetype which

contains a sample Maven project.)

…

1222: remote -> us.fatehi:schemacrawler-archetype-plugin-lint (-)

Choose a number: 513:

Вас попросят выбрать **версию архетипа**. Архетип по умолчанию (default) – это последняя стабильная версия архетипа.

Затем, у вас спросят ввести параметры проекта, такие как ***groupId***, ***artifactId***, ***version*** и ***package***.

***GroupId*** параметр, используемый для определения иерархического расположения проекта в Maven ***репозитории***.

В этом случае, ***репозиторий*** – это локальный репозиторий Maven, расположенный на вашем компьютере. ***GroupId*** определяет типичный каталог верхнего уровня, и это может быть использовано в нескольких проектах в организации.

***ArtifactId*** идентифицирует ваш проект и ***version*** указывает версию проекта.***ArtifactId*** используется, когда артефакты распакованные в репозиториии используется как зависимость в других проектах.

|  |  |
| --- | --- |
|  | |
| **{?}** | Установлен Java SDK? |

1. По завершению вы должны увидеть следующее:

[INFO] -------------------------------------------

[INFO] BUILD SUCCESS

[INFO] -------------------------------------------

Мы используем ***Maven plugin***, для создания или генерации нового проекта.

Новый проект ***Apache Maven*** может быть создан «вручную» c pom.xml и папками, созданными или сгенеренными в соответствии с архитектурой Maven проекта.

|  |  |
| --- | --- |
| **{?}** | Что же maven создаст для нас? |

Ответ – следующее:

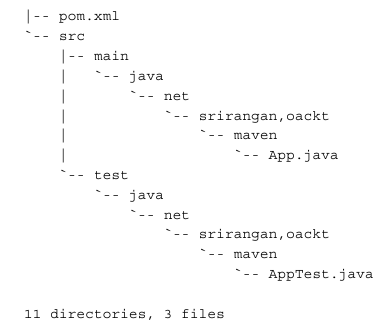


Рисунок 2.2.1 Структура файлов созданные maven.

Maven создал ***папку для проекта***, содержащую главный ***pom.xml*** и папку для исходных кодов ***src***, который содержит подпапки для приложения и папки для тестовых исходных кодов.

Задание 2

Создайте стандартный проект maven. Настройте Idea IDE. Произвеите сборку проекта. Напишите проблемы, которые у вас возникли.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

2.3 Компилирование и тестирование проекта

Для того, что бы откомпилировать и протестировать проект с помощью maven, необходимо сделать следующее:

1. Откройте консоль ***(cmd)***
2. Перейдите в проектную папку, содержащую ***pom.xml***
3. Запустите следующую команду:

mvn compile

Apachemaven начинает скачивать зависимости (артефакты), если они не доступны в локальном репозитории, и затем компилирует проект.

[INFO] -------------------------------------------

[INFO] BUILD SUCCESS

[INFO] -------------------------------------------

Если результат-***BuildSuccess***, это значит ***maven завершил компиляцию*** и сборку приложения.

***TestDrivenDevelopment (TDD)*** – популярная практика, рекомендуемая большинством самых знаменитых IT-компаний с мировым именем

Maven сделал тестирование частью стандартного жизненного цикла сборки.

Что в свою сделало TDD более простым для команд, которые пытаются внедрить эту практику в процессы разработки.

Когда используется стандартное соглашение maven, то проект включает src/test папку, которая содержит все тесты для кода. Для запуска тестов, вам необходимо запустить следующую команду:

mvn test

Эта команда запускает на выполнение тесты maven, находящиеся в ***../src/test*** папки. Когда выполнение команды закончится, вы увидите отчетна консоли:

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

--------------------------------------------------------

T E S T S

--------------------------------------------------------

Running net.srirangan.packt.maven.AppTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.037

sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

|  |  |
| --- | --- |
| **{?}** | Как же это все работает? |

Мы запускаем следующую команду:

mvn compile

Команда запустит Java компилятор, ассоциированный с проектом. Откомпилированный код кладется в target директорию. Target директория обычно содержит откомпилированные артефакты (это может быть jar файл по умолчанию) вместе с папками для откопилированных классов и тестов.Эта папка, также, содержит pom.properties файл, а также тестовые отчеты и временные файлы.

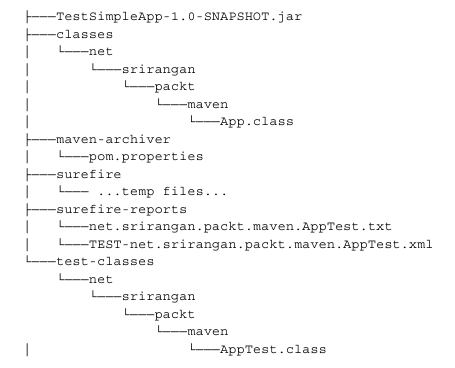


Рисунок 2.3.1 Target директория

|  |
| --- |
|  |

Задание 3

Измените проект, добавьте ресурсы. Скомпилируйте проект с помощью maven. Проверьте, что компиляция прошла успешна. Опишите проблемы, возникшие во время компиляции. Напишите JUnit-тесты. Проверьте работоспособность сборки приложения.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

2.4 Описание Объектной модели проекта (POM)

Каждый ApacheMaven проект содержит файл ***pom.xml***.

***Объектная модель*** проекта содержит проектную конфигурацию, детали системы учета дефектов, проектную организацию и лицензии, проектные пути, зависимости и т.д. Структура типового стандартного файла **pom.xml** выглядит следующим образом:

**<project … >**

**<modelVersion>4.0.0</modelVersion>**

<!--Базис -->

**<groupId>...</groupId>**

**<artifactId>...</artifactId>**

**<version>...</version>**

**<packaging>...</packaging>**

**<dependencies>...</dependencies>**

**<parent>...</parent>**

**<dependencyManagement>..**

**.</dependencyManagement>**

**<modules>...</modules>**

**<properties>...</properties>**

<!—Настройкисборки -->

**<build>...</build>**

**<reporting>...</reporting>**

<!—Метаданныепроекта -->

**<name>...</name>**

**<description>...</description>**

**<url>...</url>**

**<inceptionYear>...</inceptionYear>**

**<licenses>...</licenses>**

**<organization>...</organization>**

**<developers>...</developers>**

**<contributors>...</contributors>**

<!—Средаразработки -->

**<issueManagement>...</issueManagement>**

**<ciManagement>...</ciManagement>**

**<mailingLists>...</mailingLists>**

**<scm>...</scm>**

**<prerequisites>...</prerequisites>**

**<repositories>...</repositories>**

**<pluginRepositories>...</pluginRepositories>**

**<distributionManagement>...</distributionManagement>**

**<profiles>...</profiles>**

**</project>**

Минимальное число обязательных полей pom-файл авключает в себя - groupId, artifactId и version. Эти три поля определяют расположение артефакта в репозитории.

Представленный выше пример ***pom файла*** включает 4 основные секции:

1. ***Базис***. Эта секция содержит поля, определяющие уникальность артефакта (groupId, artifactId и version), управление зависимостями, а также детали наследования.
2. ***Настройки сборки***. Детали сборки располагаются в этой секции.
3. ***Метаданные проекта***. Эта секция включает специфичные для проекта параметры, такие как имя, организация, разработчики, адрес сайта организации, которая разрабатывает продукт, год поставки и т.д.
4. ***Среда разработки***. Эта секция содержит всю информацию, касающуюся среды разработки, включая детали о системе контроля версий, системе управления дефектами, системы непрерывной сборки, списки рассылок, репозиториях и т.д.

|  |
| --- |
|  |
|  |
|  |

2.5 Описание жизненного цикла сборки.

***Жизненный цикл сборки*** точно определяет процесс сборки, тестирования, выпуска артефактов, и это является сердцем каждого maven проекта.

Существует три встроенных жизненных циклов сборки: *стандартный, очистить и сайт.*

*Стандартный жизненный цикл* обрабатывает компиляцию проекта, тестирование и установка. Он содержит более чем 20 фаз, наиболее важные из которых перечислены ниже:

1. ***Проверка (validate)***: проверяет, что вся проектная информация доступна и корректна.
2. ***Компилирование (compile):*** компилирует исходный код.
3. ***Тестирование (test)***: запускает модульные тесты.
4. ***Упаковка (package)***: упаковывает откомпилированный код в соответствующий выходной формат.
5. ***Интеграционное тестирование (integration-test)***: запускает интеграционные тесты на тестовой среде.
6. ***Проверка упаковки (verify)***: Запускает проверки на правильность упаковки.
7. ***Установка (Install)***: устанавливает упакованный артефакт на локальный репозиторий.
8. ***Установка на удаленный репозиторий (deploy):*** установка финального упакованного артефакта на удаленный репозиторий.

|  |
| --- |
|  |
|  |
|  |

Как бы вы не запускали на выполнение фазу сборки, все предыдущие фазы выполняются последовательно. Например, выполняя команду

mvn integration-test, вы запускаете проверку, компиляцию, тестирование и упаковку, до того момента пока выполнится фаза интеграционного тестирования.

***Жизненный цикл очистки***отвечает за процесс очистки проекта от предыдущих файлов сборкии состоит из следующих фаз:

1. ***Подготовка к очистке (pre-clean):*** выполняет процессы необходимые до очистки предыдущих файлов сборки проекта.
2. ***Очистка (clean):*** удаляет все файлы полученные при предыдущей сборки.
3. ***Фаза после очистки (post-clean):*** выполняет процессы необходимые для завершения очистки файлов предыдущей сборки проекта.

***Жизненный цикл сайт*** отвечает за генерацию и установку документации проекта, состоит из следующих фаз:

1. Подготовка к генерации документации (pre-clean): выполняет процессы необходимые до генерации документации проекта.
2. Генерация документации (site): генерация проектной документации.

2.6 Описание профайлов сборки.

Проекты в maven портируемы. Портируемость достигается посредством ***конфигурации POM***, обходя все особенности файловой системы, и сильно зависима от локального репозитория, в котором сохранены необходимые метаданные.

Однако не всегда возможно достичь портируемости, благодаря внутренним взаимозависимостям конфигурации и файловой системы. В таких случаях maven представляет концепцию ***профайлов сборки***.

***Профайл сборки*** *–* это настройки, описанные в pom.xml, которые могут быть запущены когда необходимо.

Существуют следующие варианты запуска профайлов:

1. Запуск профайла через командную строку.
2. Запуск профайла в зависимости от настроек maven.
3. Запуск профайла в зависимости от среды разработки.

Профайл может быть запущен через ***командную строку*** при помощи параметра –P. Список профайлов, разделенных запятыми, которые должны быть активированы, должны быть перечислены после флага –P.

mvn install -P profile-1,profile-2

|  |
| --- |
|  |
|  |
|  |

В этом случае, только *профайлы*, указанные в команде, все же другие сконфигурированные *профайлы*, будут ***неактивны***.

Обратное действие также ***возможно***. Вы можете указать, какие профайлы не должны быть активированы.

mvn install -P !profile-1,!profile-2

|  |
| --- |
|  |
|  |
|  |
|  |

Настройки maven могут активировать профайлы, если они указанны в секции

<activeProfiles>файла settings.xml.

<settings>

...

<activeProfiles>

<activeProfile>profile-1</activeProfile>

<activeProfile>profile-2</activeProfile>

</activeProfiles>

...

</settings>

Профайлы, указанные в настройках maven активируются по умолчанию каждый раз и их не нужно явно указывать в командной строке.

Профайлы могут быть запущены в зависимости от среды разработки. Среда, в которой профайл должен быть активирован, определяется в секции <profiles> файла pom.xml.

<profiles>

<profile>

<activation>

<property>

<name>environment</name>

<value>dev</value>

</property>

</activation>

</profile>

</profiles>

В коде, указанном выше, профайл может быть активирован в dev среде разработке. Пример типичной команды maven, в которой среда разработки указана явно, выглядит так:

mvn groupId:artifactId:goal -Denvironment=dev

|  |
| --- |
|  |
|  |
|  |

Задание 4

Выполните сборку проекта с различными фазами жизненного цикла сборки.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Тема 3.Инженерные техники при применении Apache Maven.**

3.1 Автоматизация сборки

***Автоматизация сборки*** *–* это реализованные на языке программирования задачи, которые разработчики выполняют ежедневно для запуска и тестирования написанного кода.

Эти задачи включают:

1. Компиляцию исходного кода в исполняемый код.

|  |
| --- |
|  |
|  |
|  |

1. Упаковка исполняемого кода.

|  |
| --- |
|  |
|  |
|  |

1. Запуск тестов.

|  |
| --- |
|  |
|  |
|  |

1. Установка упакованного исполняемого кода на удаленные системы.

|  |
| --- |
|  |
|  |
|  |

1. Создание документации к релизу и продукту.

|  |
| --- |
|  |
|  |
|  |

Автоматизация сборки представляет ряд преимуществ, включая ускорение сборки, ограничение неудачных сборок, стандартизация и унификация в командах и организациях, увеличение эффективности и улучшение в продуктовом качестве. Сегодня, эта практика необходима для ежедневного использования.

Если вам необходим ***готовый maven проект***. Если у вас нет готового, запустите следующую команду в командной строке, для создания простого Java проекта:

mvn archetype:generate -DgroupId=com.epam.maven

-DartifactId=MySampleApp

Команда archetype:generate создал Apache Maven проект для нас. Если мы выберем maven-archetype-quickstartархетип из списка, наша структура проекта будет выглядеть похожей на рисунок 3.1.1.

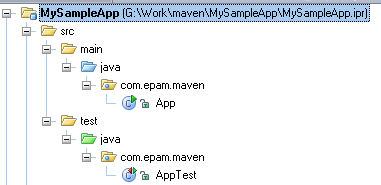


Рисунок 3.1.1 Структура проекта

В каждом ApacheMaven проект, включая тот, который мы сгенерировали, сборка запускает автоматически стандартный жизненный цикл сборки.

|  |
| --- |
|  |
|  |
|  |

Тоже самое вы можете выполнить вручную:

mvn validate

...

mvn compile

...

mvn package

...

mvntest

...

Вы можете включить некоторые фазы жизненного цикла сборки индивидуальной командой. Maven позволяет вам ***автоматизировать запуск всех фаз сборки в правильном порядке***.

Запустите команду mvninstall и вы запустите часть стандартного жизненного цикла сборки, включая компилирование, тестирование, упаковку и установку артефакта в локальный репозиторий.

Автоматизация сборки стандартизирует структуру проекта, что может упростить разработку и тестирование продуктов.

|  |
| --- |
|  |
|  |

3.2 Модульность проекта

Предположим, что вы собираете большое приложение, требующее взаимодействие со старой базой данных, работающее с существующими сервисами, предоставляющие современные графические интерфейсы как для web- и прикладных-составляющих, и предоставляющих API другим приложениям заказчиков. Такая сборка подразумевает разделение этого достаточно большого проекта на ***подпроекты или модули****.*

Apache maven предоставляет верную и точную поддержку, для такой проектной организации, через ***Apache Maven мультимодульные проекты***. Мультимодульные проекты состоят из ***«родительского проекта»***, который включает ***«дочерние проекты»*** или модули.

|  |
| --- |
|  |
|  |
|  |

Родительский pom файл содержит ссылки на все дочерние модули. Каждый может быть разного типа, с разным упаковочным расширением.

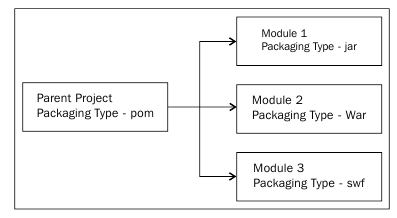


Рисунок 3.2.1 Мультимодульный проект.

Мы начинаем с создания родительского модуля. Не забудьте поставить значение ***packaging* в *pom***, как представлено в следующем ***примере***:

<?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>com.epam.maven</groupId>

<artifactId>TestModularApp</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>**pom**</packaging>

<name>MyLargeModularApp</name>

</project>

Это ***базовый родительский pom-файл*** для нашего проекта MyLargeModularApp. Он не содержит никакие дочерние модули на данный момент.

|  |
| --- |
|  |
|  |
|  |

Для создания нашего ***первого дочернего модуля***, запустите консоль командной строки (cmd), перейдите в директорию, где лежит родительский pom- файл и запустите следующую команду:

mvn archetype:generate

Данная команда покажет список всех архетипов на ваш выбор. Вы можете выбрать ***архетип maven-archetype-quickstart***, который сгенерирует базовый Java модуль. Команда *archetype:generate* требует заполнения параметров maven проекта: groupId, artifactId, package, version и др.

|  |
| --- |
|  |
|  |
|  |

После генерации модуля, просмотрите родительский pom файл, и вы увидите, что следующий блок был добавлен в секцию модулей:

<modules>

<module>moduleJar</module>

</modules>

Дочерний модуль, который мы создали, автоматически был добавлен в проект к родительскому модулю. Эта процедура работает достаточно просто – ***никакого внешнего вмешательства не требуется***.

Сейчас мы создадим другой дочерний модуль. Пришло время создать web-модуль, путем вызова следующей команды:

mvn archetype:generate -DarchetypeArtifactId=maven-archetype-webapp

Давайте, снова, посмотрим на родительский pom файл, и мы увидим, что оба дочерних модуля добавлены в секцию модули:

<modules>

<module>moduleJar</module>

<module>moduleWar</module>

</modules>

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

Компиляция и установка (еще бывает и распаковка на сервере) ***обоих дочерних модулей*** (в правильном порядке, в случае, если дочерние модули взаимозависимы) – это ***главное*.** Данная операция может быть сделана, путем запуска командной строки, перехода в директорию, где располагается родительский pom файл, и запуска следующей команды:

mvn clean install

Данная команда, запуская процесс сборки для родительского модуля, так же автоматически запускает сборку всех дочерних модулей в правильном порядке.

|  |
| --- |
|  |
|  |
|  |

Задание 5

Создайте многомодульный проект. Одни модуль - jar, другой модуль – war. Создайте parent модуль. Проверьте процесс сборки и опишите ваши вопросы.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

3.3 Управление зависимостями

***Управление зависимостями*** заслуженно считается одной из лучших возможностей ApacheMaven. В мультимодульных проектах, где количество зависимостей достигает десятков или даже сотен, ApacheMaven отличается от других систем сборки, тем, что позволяет сохранять высокий уровень контроля и стабильности системы.

***Управление зависимостями в ApacheMaven***– это прозрачный процесс. Это значит, что maven автоматически находит артефакты, от которых зависит ваш проект. Эта возможность стала доступной, начиная с версии Maven 2, и это стало особенно удобно для многих проектов с открытым исходным кодом (opensource projects) и для их зависимостей, которые мы используемв сегодняшних проектах.

Зависимости в maven имеют 6 возможных областей действий:

1. Зависимость, необходимая для компилирования (***compile***): Эта область действия по умолчанию. Зависимость доступна в classpath.

|  |
| --- |
|  |
|  |
|  |

1. Зависимость, представлена во времени выполнения (***provided***): Эта область действия предполагает, что JDK или среда разработки предоставляет зависимость во времени выполнения.

|  |
| --- |
|  |
|  |
|  |

1. Зависимость, представлена во время выполнения (***runtime***): Зависимость необходима во время выполнения и определена в runtime classpath.

|  |
| --- |
|  |
|  |
|  |

1. Зависимость, необходима для тестирования (***Test***): Зависимости необходимы для компиляции и выполнения тестов.

|  |
| --- |
|  |
|  |
|  |

1. Системная зависимость (***System***): Зависимость всегда доступна, и в любом случае наличие JAR обеспечено.

|  |
| --- |
|  |
|  |
|  |

1. Импортрируемая зависимость (***Import***):Имортрируемая зависимость представлена в POM, включая <dependencyManagement/> секцию.

|  |
| --- |
|  |
|  |
|  |

Зависимости для ApacheMaven проекта описаны в проектном pom-файле. Пока мы глубже начали рассматривать, как работать с зависимостями, мы рассмотрим ApacheMaven плагин управления зависимостями (***ApacheMaven dependency plugin***).

*«Плагин управления зависимостями предоставляет возможность манипулирования артефактами. Он может копировать или распаковывать артефакты из локального или удаленного репозитория в указанное место на жестком диске»*

|  |
| --- |
|  |
|  |

В соответствии с <http://maven.apache.org/plugins/maven-dependency-plugin/>:

Это плагин, который предоставляет ряд удобных задач (goals), которые заключаются в следующем:

1. ***mvn dependency:analyze.*** Анализирует зависимости (используемые, неиспользуемые, декларируемые или не декларируемые зависимости).
2. ***mvn dependency:analyze-duplicate***. Определяет дублирующие друг друга зависимости.
3. ***mvn dependency:resolve***. Определить все зависимости.
4. ***mvn dependency:resolve-plugin***. Определитьвсеплагины.
5. ***mvn dependency:tree***. Определить дерево зависимостей.

Большинство maven проектов зависимы от других артефактов (других проектов, библиотек и инструментов). Управление и интеграция зависимостями является одной из основных особенностей maven. Зависимости проекта указаны в pom-файле.

<dependencies>

<dependency>

<groupId>...</groupId>

<artifactId>...</artifactId>

<version>...</version>

<scope>...</scope>

</dependency>

</dependencies>

В мультимодульных проектах, зависимости могут быть определены в родительскомpom-файле и могут впоследствии унаследованы в дочерних pom- файлах, когда это необходимо. Расположение всех зависимостей в одном источнике делает версионность зависимостей более простой, что делает большие проектные зависимости более управляемыми и масштабируемыми во времени.

|  |
| --- |
|  |
|  |

Следующий ***пример*** показывает мультимодульный проект, обладающий зависимостью от MySql.

Родительский pom будет содержать полное описание зависимости:

<dependencyManagement>

<dependencies>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.2</version>

</dependency>

<dependencies>

</dependencyManagement>

|  |
| --- |
|  |
|  |
|  |

Все дочерние модули требующие MySql будут включать упрощенное определение зависимости:

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

При таком описании не будет конфликта версий между многочисленными дочерними модулями, имеющие те же зависимости.

Область действия и тип зависимости по умолчанию – это ***compile и JAR***. Однако, они могут быть переопределены по требованию:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.8.2</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>...</groupId>

<artifactId>...</artifactId>

<version>...</version>

<type>war</type>

</dependency>

Системные зависимости не ищутся в репозитории. Мы должны указывать путь к таким зависимостям сами:

<dependencies>

<dependency>

<groupId>sun.jdk</groupId>

<artifactId>tools</artifactId>

<version>1.5.0</version>

<scope>system</scope>

<systemPath>${java.home}/../lib/tools.jar</systemPath>

</dependency>

</dependencies>

|  |
| --- |
|  |
|  |

Однако очень ***рекомендовано избегать системных зависимостей***, потому что это противоречит общей концепции управления зависимостями ApacheMaven.

В идеале, разработчик должен иметь возможность взять код из системы контроля версий и запустить ApacheMaven командой. После этого, ApacheMaven должен отвечать за управление зависимостями.

Задание 6

Настройте зависимость многомодульного проекта на MySQL database driver. Выполните сборку проекта и проверьте, что она прошла успешно.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

3.4 Автоматизация установки проекта.

***MavenDeploy плагин*** используется для добавления артефактов на удаленный репозиторий, в процессе фазы установки в жизненном цикле сборки.

Deploy плагин представляет две задачи:

1. ***deploy:deploy***: Установка проекта и всех его артефактов.
2. ***deploy:deploy-file***: Установка одного простого артефакта.

Установка в репозиторий означает не только копирование артефакта в директорию, но для обновления метаданных, так же как и артефактов. Это требует:

1. ***targetrepository:*** Целевой репозиторий – это где артефакты будут установлены. Это необходимое расположение, протокол доступа (FTP, SCP, SFTP) и специфическая пользовательская информация о профиле.
2. ***targetartifacts***: Целевые артефакты – это артефакты, которые будут установлены. GroupId, artifactId, version, packaging, и classifier информация артефакта необходимы.
3. ***deployermethod***: Методустановки, с помощью которого происходит установка. Метод может быть реализован либо в кросс- платформенном, либо в платформенном виде.

Проектный pom файл должен иметь <distributionManagement/> секцию с <repository/> элементами внутри, в которых описаны расположение и параметры удаленного репозитория.

<distributionManagement>

<repository>

<id>by.park</id>

<name>MyPrivateRepository</name>

<url>...</url>

</repository>

</distributionManagement>

Дляэтого, вам необходимо описать информацию о сервере в вашем settings.xml (<USER\_HOME>/.m2/settings.xmlили<M2\_HOME>/conf/settings.xml) для предоставления идентификационной информации для репозиториев.

<server>

<id>by.park</id>

<username>ivan</username>

<password>myTopSecretPassword</password>

</server>

Установка может быть выполнена из командной строки, перейдя в директорию проекта и вызвав следующую команду:

mvn deploy

|  |
| --- |
|  |
|  |
|  |

Snapshots и releases могут быть разделены определением <snapshotRepository/> элемента.

<distributionManagement>

<repository>

<uniqueVersion>false</uniqueVersion>

<id>corp1</id>

<name>Corporate Repository</name>

<url>scp://repo/maven2</url>

<layout>default</layout>

</repository>

<snapshotRepository>

<uniqueVersion>true</uniqueVersion>

<id>propSnap</id>

<name>Propellors Snapshots</name>

<url>sftp://propellers.net/maven</url>

<layout>legacy</layout>

</snapshotRepository>

...

</distributionManagement>

Секция <distributionManagement/> наследуется всеми дочерними модулями. Действительная установка происходит на основе протоколов, определенных в репозитории. Чаще всего используют FTP и SSH протоколы. Wagon-ftp и wagon-ssh-external – основные провайдеры для этих протоколов.

|  |
| --- |
|  |
|  |
|  |
|  |

Если удаленный репозиторий доступен через FTP протокол, то <build/> секция должна включать определение wagon-ftp.

<distributionManagement>

<repository>

<id>sri-ftp-repository</id>

<url>ftp://...</url>

</repository>

</distributionManagement>

<build>

<extensions>

<extension>

<groupId>org.apache.maven.wagon</groupId>

<artifactId>wagon-ftp</artifactId>

<version>1.0-beta-6</version>

</extension>

</extensions>

</build>

Если удаленный репозиторий доступен через SSH, определяемое по artifactId и URL, тогда<build/> секция включает определение wagon-ssh-external.

<distributionManagement>

<repository>

<id>sri-ssh-repository</id>

<url>scpexe://....</url>

</repository>

</distributionManagement>

<build>

<extensions>

<extension>

<groupId>org.apache.maven.wagon</groupId>

<artifactId>wagon-ssh-external</artifactId>

<version>1.0-beta-6</version>

</extension>

</extensions>

</build>

Идентификационная информация будет храниться в settings.xml.

|  |
| --- |
|  |
|  |
|  |

3.5 Сборка web-приложения.

|  |  |
| --- | --- |
| **{?}** | Как же создать и собрать web-приложение с использованием maven? |

Мы начнем с концепции ***POWA*** (простое web приложение) – это web проект, состоящий из сервлетов и нескольких JSP. Шаблон, который может использоваться для генерации POWA называется org.sonatype.mavenbook.simpleweb.

Откройте командную строку и запустите следующую команду:

mvn archetype:generate -DarchetypeArtifactId=maven-archetype-webapp -DartifactId=testWebApp -DgroupId=com.epam.maven -Dversion=1.0-SNAPSHOT -Dpackage=com.epam.maven

Вы должны увидеть следующее в консоли:

[INFO] ------------------------------------------------------------------

[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-webapp:1.0

[INFO] ------------------------------------------------------------------

[INFO] Parameter: groupId, Value: com.epam.maven

[INFO] Parameter: packageName, Value: com.epam.maven

[INFO] Parameter: package, Value: com.epam.maven

[INFO] Parameter: artifactId, Value: testWebApp

[INFO] Parameter: basedir, Value: G:\Work\maven

[INFO] Parameter: version, Value: 1.0-SNAPSHOT

[INFO] project created from Old (1.x) Archetype in dir: G:\Work\maven\testWebApp

[INFO] ------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------

|  |
| --- |
|  |
|  |

Комнада archetype:generate создала новую проектную директорию testWebApp, которая содержит следующий pom-файл:

<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/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.epam.maven</groupId>

<artifactId>testWebApp</artifactId>

<packaging>war</packaging>

<version>1.0-SNAPSHOT</version>

<name>testWebApp Maven Webapp</name>

<url>http://maven.apache.org</url>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<finalName>testWebApp</finalName>

</build>

</project>

|  |
| --- |
|  |
|  |
|  |

Сейчас, мы добавим buildplugin, для компилирования проекта с помощью JDK 1.6. <build/> секция pom файла должна иметь похожий вид:

..

<build>

<finalName>testWebApp</finalName>

<plugins><plugin>

<artifactId>maven-compiler-plugin</artifactId>

<configuration><source>1.6</source><target>1.6</target>

</configuration>

</plugin></plugins>

</build>

..

Apache Maven web проект будет работать, как многие другие maven проекты. Структура проекта изображена на рисунке 3.5.1.

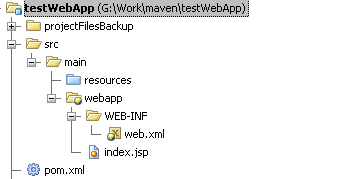


Рисунок 3.5.1 Структура maven web проекта.

Вы можете выполнить следующие команды для компилирования и тестирования проекта:

1. mvn compile
2. mvn test
3. mvn install

По окончанию установки, выходной артефакт будет упакован и установлен в локальный репозиторий, какдругие maven проекты. Он также может быть установлен на удаленный репозиторий.

Если вы посмотрите, вы можете найти проектную директорию src/main/webapp, в которой находится файл index.jsp. Вы можете изменить этот файл, как вам необходимо, например вот так:

<html>

<body>

<h2>Maven is very useful tool</h2>

</body>

</html>

После успешной сборки проекта командой mvn install, ***target*** директория была создана. В target директории вы можете найти WAR файл с именем ***testWebApp.war***. Давайте посмотрим на pom файл, упакованный в данном архиве.

...

<artifactId>testWebApp</artifactId>

**<packaging>war</packaging>**

<version>1.0-SNAPSHOT</version>

<name>testWebApp Maven Webapp</name>

...

Как мы можем видеть, что packaging тип установлен в WAR и выходной артефакт отражает это свойство.

Задание 7

Создайте новое web приложение. Добавьте туда исходные файлы, которые вы делали для учебного проекта. Скомпилируйте и соберите готовое веб приложение.

|  |
| --- |
|  |
|  |
|  |
|  |

**Тема 4. Apache Tomcat Сервер**

4.1 Описание Apache Tomcat.

***ApacheTomcat сервер*** – это основанный на платформе Java контейнер для веб-приложений. Сервер создан для запуска веб-приложений, основанных на сервлетах и JavaServerPages (JSP). Он был создан как подпроект ***Apache-Jakarta***; однако, благодаря своей популярности, сейчас, существует как отдельный Apache проект, который поддерживается группой волонтеров сообщества JAVA.

JVM совместимость зависит от выбранной версии сервера Tomcat.

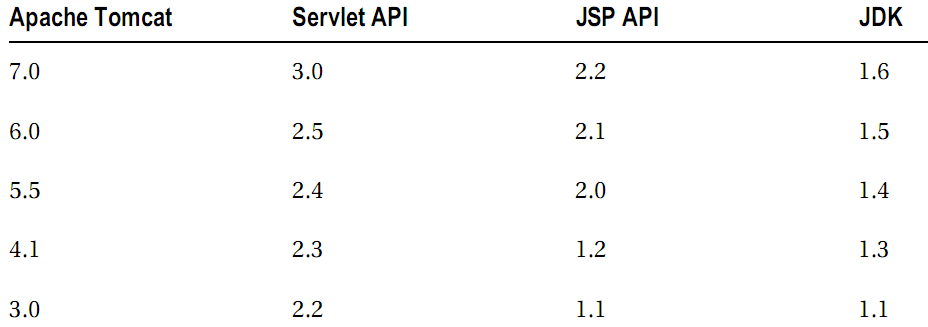


Рисунок 4.1.1 Версии Tomcat сервера и соответствующие им версии Servlet, JSP API и JDK.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

4.2 Архитектура Tomcat

***Сервер Tomcat*** – это высокоуровневый компонент в иерархии контейнеров Tomcat. Только одна запущенная сущность Tomcat сервера может работать в рамках JVM. Такой подход делает работу всех других Java приложений, запущенных на том же компьютере, что и сервер Tomcat, ***безопасной***, в случае краха JVM или Tomcat.

|  |
| --- |
|  |
|  |

Сервер Tomcat состоит из группы ***контейнеров приложений***, которые существуют в строго определенной иерархии. Ключевой компонент иерархии – это ***Catalina***. ***Catalina***– это Java сервлет контейнер, реализованный в соответствии со спецификацией JavaServletAPI.

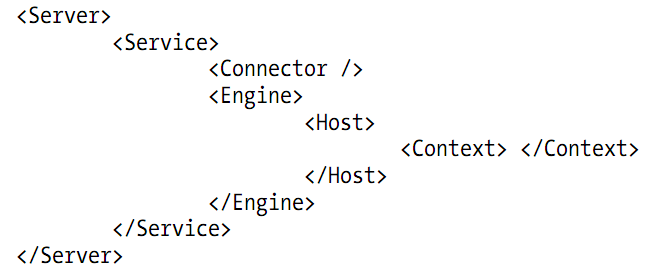


Рисунок 4.2.1 XML представление архитектуры компонентов Tomcat.

Эта сущность может быть разбита на серию ***контейнеров***, включающих *сервер, сервис, соединитель, ядро, хост, и контекст*.

|  |
| --- |
|  |
|  |
|  |
|  |

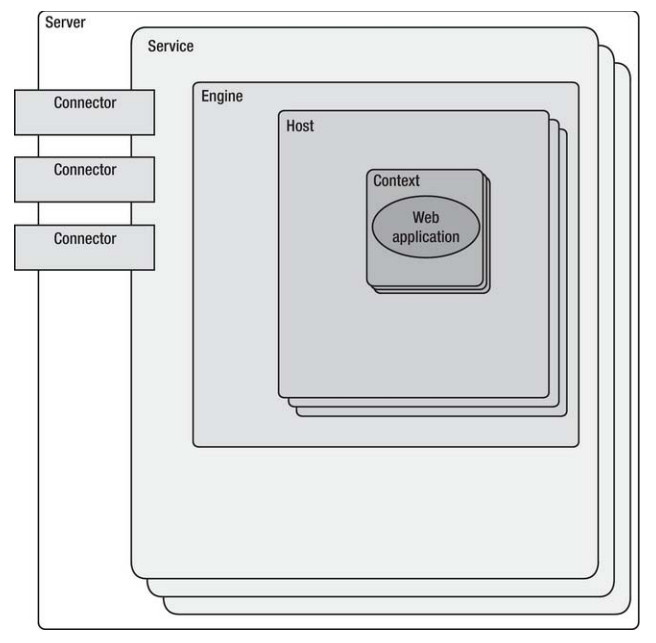


Рисунок 4.2.2 Архитектура Tomcatс основными компонентами.

По умолчанию, каждый из этих контейнеров конфигурируется, используя ***server.xml***.

***Сервер***

Первый контейнерный элемент, представленный на рисунке 4.2.1 как **<Server>.** Он представляет собой ***цельный Catalina сервер***и используется как высокоуровневый элемент в отдельной сущности Tomcat. Элемент <Server> может содержать одну или более <Service> контейнеров.

|  |
| --- |
|  |
|  |

***Сервис***

Следующий контейнерный элемент***<Service>***, который владеет коллекцией одного или более <Connector> элементов, которые делят одиночный<Engine> элемент. N-число<Service> элементов могут быть вложенными в отдельный <Server> элемент.

|  |
| --- |
|  |
|  |

***Соединитель***

Следующий тип элемента – это *<****Connector>*** элемент, который определяет класс, фактически обрабатывающий запросы и ответы от вызывающего клиентского приложения.

|  |
| --- |
|  |
|  |

***Ядро***

Третий контейнерный элемент – это ***<Engine>*** элемент. Каждый установленный <Service> может содержать тока один <Engine> элемент, и этот один <Engine> компонент обрабатывает все запросы, полученные всеми установленными <Connector> компонентами, определенные родительским сервисом.

|  |
| --- |
|  |
|  |

***Хост***

***<Host>***элемент определяет виртуальные хосты, которые содержатся в каждой сущности ***Catalina<Engine>***. Каждый<Host> может быть родительским контейнером для одного или более веб-приложений, с каждой из которых представлен <Context>компонент.

|  |
| --- |
|  |
|  |

***Контекст***

***<Context>*** элемент- это наиболее часто используемый контейнер в сущности Tomcat. Каждый <Context> элемент представляет индивидуальное веб-приложение, которое запускается с определенным <Host>. Нету ограничений, какое число контекстных элементов может быть определено в рамках <Host> элемента.

|  |
| --- |
|  |
|  |
|  |

4.3 Установка и конфигурирование Tomcat

Для установки и настройки Tomcat, прежде скачайте **установочный пакет** с http://tomcat.apache.org .

Сначала вы должны установить ***JavaDevelopmentKit (JDK)***. Так как мы будем использовать ***Tomcat 7***, то вы должны установить ***JDK 1.6*** или более поздние версии.

Послеустановки JDK, мы будем устанавливать ***apache-tomcat-7.0.29.exe***. Tomcat Windows установщик очень похож на другие Windows установщики.

**Следуйте шагам, выбирайте, куда вы хотите установить сервер, и установщик распакует и скопирует файлы в корректную директорию, и сконфигурирует переменные среды и сервисные свойства.**



Рисунок 4.3.1 Установка ApacheTomcat.

После завершения установки, вы можете проверить, что переменная среды ***CATALINA\_HOME*** была установлена. Если она не была установлена, то установите ***вручную***.

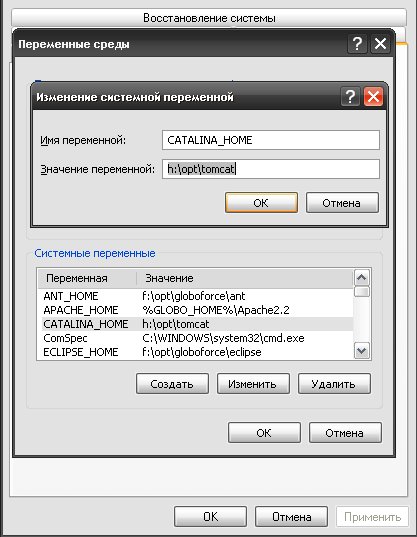


Рисунок 4.3.2 Установка переменной среды CATALINA\_HOME

Windows установщик установит Tomcat7 как Windows Сервис автоматически, так что Tomcat может быть запущен автоматически при старте windows, и работать в фоновом режиме как Windows Сервис.

|  |
| --- |
|  |

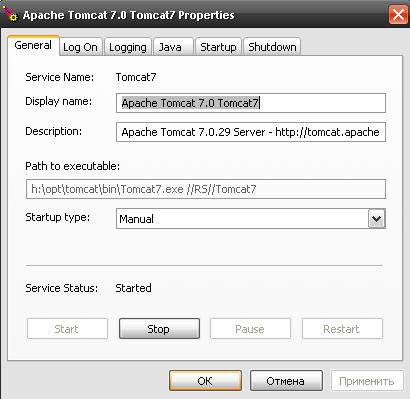


Рисунок 4.3.3 Утилита для запуска и остановки ApacheTomcat, как Windows Сервиса.

|  |
| --- |
|  |
|  |
|  |

4.4 Тестирование установки сервера Tomcat

Для тестирования установки сервера Tomcat, вам необходимо для начала **запустить** его.

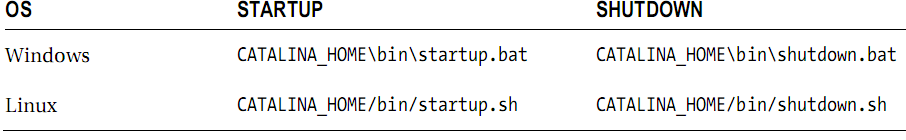


Рисунок 4.4.1 Команды запуска и остановки сервера Tomcat.

Вы также можете запускать и останавливать Tomcat через утилиту для запуска и остановки ApacheTomcat, в случае использования ***установщика Windows***.

Когда Tomcat запустился, откройте в вашем браузере следующий URL:

<http://localhost:8080/>

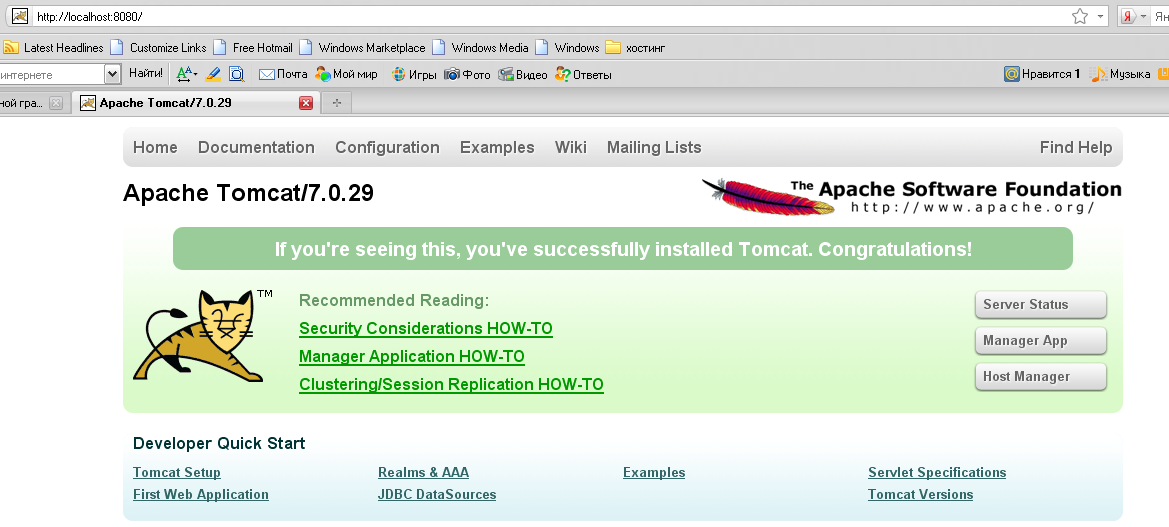


Рисунок 4.4.2 Домашняя страница Tomcat сервера.

|  |
| --- |
|  |
|  |
|  |

***Следующий шаг***, это проверить, что JSP страницы корректно обрабатываются сервером Tomcat. Вы можете сделать это, запустив один из JSP примеров, предоставляемый сервером Tomcat.

Для запуска примера JSP, откройте в браузере страницу <http://localhost:8080/examples/jsp/> и выбрать ***JSP пример***.

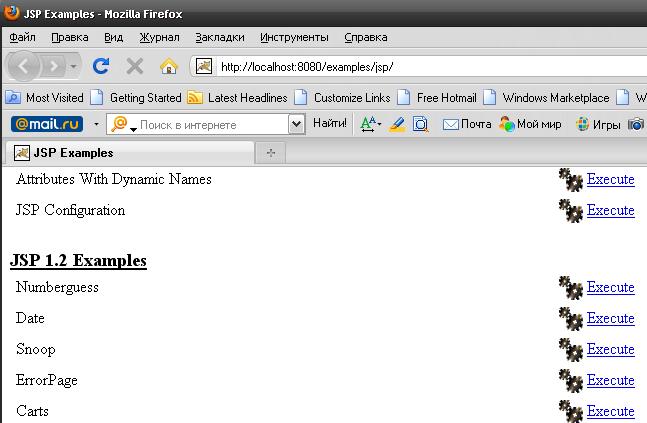


Рисунок 4.4.3 JSP примеры сервера Tomcat.

Сейчас выберите пример ***Date*** и нажмите на ***Execute*** ссылку. Если все было установлено правильно, то вы увидите страницу с датой.

|  |
| --- |
|  |
|  |

Если вы не увидели страницу с датой, проверьте, что значение вашей переменной среды ***JAVA\_HOME*** совпадает с местом, где установлена ваша ***JDK***.

|  |
| --- |
|  |
|  |

4.5 Структура директорий сервера Tomcat

Для того чтобы ***установки веб-приложения на сервер*** необходимо сначала рассмотреть структуру директорий сервера Tomcat, а затем перейти к установки веб-приложения.

Вся ***структура директорий*** сервера Tomcat лежит в установочной директории ***CATALINA\_HOME***.

***/bin***–Содержит скрипты запуска и остановки сервера как для ОС Windows, так и для Linux. Jar файлы с классами, необходимыми для запуска сервера, тоже находятся здесь.

|  |
| --- |
|  |
|  |

***/conf*** – Содержит основные конфигурационные файлы сервера Tomcat.Два наиболее важных конфигурационных файлов – это ***server.xml*** и ***web.xml*** (глобальный).

|  |
| --- |
|  |
|  |

***/lib*** – СодержитTomcatjava архив (jar) файлы, разделенные среди всех компонентов Tomcat. Все веб-приложения, установленные на сервер Tomcat, могут получить доступ к библиотекам, расположенным в этой директории. Она включает Servlet API иJSP API библиотеки.

|  |
| --- |
|  |
|  |

***/logs*** – Содержитlog файлы сервера Tomcat.

***/temp*** – Содержит временные файлы сервера Tomcat.

|  |
| --- |
|  |
|  |

***/webapps*** – Директория, куда все веб-приложения устанавливаются, и куда вы поместите ваш WAR файл, когда он будет готов к установке.

***/work*** -Рабочая директория сервера Tomcat, куда Tomcat копирует все сервлеты, которые были сгенерированы из JSP страниц. Если вы хотите увидеть, как существующие JSP страницы превращаются в сервлеты, загляните в эту папку.

Задание 8

Скопируйте приложение, которое вы создали при помощи maven в директорию webapps. Запустите Tomcat и проверьте статус установки приложения в Tomcat с помощью log файлов.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

4.6 Выполнение скриптов сервера Tomcat

Все исполняемые файлы Tomcat находятся в ***CATALINA\_HOME/bin*** директории. Исполняемые файлы как для Windows, так и для Linux располагаются в этой директории, поэтому вы увидите парные скрипты с одним и тем же именем и разными расширениями (\*.bat для Windows и \*.sh для Linux).

Наиболее ***важные исполняемые файлы*** в этой директории – это ***Catalina скрипты*** (catalina.sh на Linux или catalina.bat для Windows). Этот скрипт отвечает за запуск и остановку сервера Tomcat, и он принимает разные параметры командной строки для разных операций. Например, для старта сервера Tomcat, вы можете просто выполнить команду catalinastart в командной строке, иcatalinastop -для остановки сервера Tomcat.

Скрипты catalina *доступны* только, если вы скачали *zip дистрибутив* сервера Tomcat. Если вы установили Tomcat с использованием установщика сервиса Windows, то по умолчанию, вы не можете использовать скрипты Catalina. В этом случае вы можете использовать tomcat7w.exe приложение для управления сервисом Tomcat или скопировать скрипты из zip дистрибутива.

Основные команды, которые вы можете использовать, когда запускаете скрипты Catalina.

***catalina start –*** запускает Tomcat как новый процесс. В ОС Windows, Tomcat будет запущен в новом окне состояния командной строки.

***catalina stop -*** останавливает Tomcat.

***catalina run –*** запускаетTomcat в текущем окне. В ОС Windows, вывод истории запуска Tomcat будет добавлен к текущему окну состояния командной строки.

***catalina debug –*** запуск Tomcat в режиме отладки.

***catalina version –*** выводверсииTomcat.

***catalina configtest –*** проверяет, что текущая конфигурация Tomcat правильна. Скрипт пытается запустить Tomcat и загрузить все конфигурационные файлы, а затем аккуратно выйти, если сервер настроен, верно. Если какие-либо файлы конфигурации настроены не верно, то скрипт выдаст отчет об ошибки.

***catalina jpdastart -***Запускает Tomcat в Java Platform Debugging Architecture (JPDA) отладочном режиме.

В дополнение, вы можете расширить опцией ***– security*** любую команду скриптов catalina. С Помощью этой опции Tomcat запускается в режиме включенного менеджера безопасности (securitymanagerenabled). ***Режим включенного менеджера безопасности –*** это механизм защиты ресурсов, управляемых JVM от несанкционированного доступа.

Директория bin также содержит два подходящих скрипта для запуска и остановки работы Tomcat: ***startup.bat и shutdown.bat***.

|  |
| --- |
|  |
|  |
|  |

4.7 Передача конфигурационных опций времени выполнения скриптам Catalina.

Когда запускаются Java приложения, то часто требуется передать различные опции *виртуальной машине java*(JavaVirtualMachine - JVM), такие как настройки памяти, настройки кодировки файлов, порты, адреса серверов и т.д. Эти опции передаются в скрипты catalina через стандартные переменные среды Java и Tomcat.

JAVA\_OPTS переменные среды содержат опции доступные всем Java процессам, запущенным на компьютере, включая Tomcat.

Для того, что бы быть уверенным, что JVM читает все файлы используя кодировку UTF-8, мы можем добавить стандартный параметр ***file.encoding*** к переменной JAVA\_OPTS: **set JAVA\_OPTS="-Dfile.encoding=utf-8"**

Если вы запустите Tomcat с установленными JAVA\_OPTS, то это гарантирует, что JVM будет использовать UTF-8 кодировку для чтения файлов не только для Tomcat процесса, но и для всех других Java процессов, запущенных на компьютере.

Для передачи специфических параметров только для Tomcat процесса, необходимо использовать CATALINA\_OPTS.

Типичный сценарий – это добавление больше памяти Tomcat процессу.

set CATALINA\_OPTS=" –Xms256m -Xmx1g -XX:MaxPermSize=256m"

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Тема 5. Конфигурирование сервера Tomcat**

Директория conf содержит все конфигурационные файлы Tomcat. Основной конфигурационный файл – это ***server.xml***. Основные Tomcat элементы ядро (engines), хосты (hosts) и контексты (contexts) расположены здесь. Context.xml конфигурация содержит настройки по умолчанию, которые могут быть использованы всеми контекстами Tomcat. К дополнению к глобальным настройкам контекста, которые расположены в server.xml и context.xml, каждый контекст может иметь свой собственный конфигурационныйфайл. Расположение конфигурационного файла зависит от имени ядра и хоста, в которых контекст определен. Соглашение, что контекст с именем CONTEXT\_NAME, настроенный с хостом HOST\_NAME, имя ядра которого ENGINE\_NAME сохраняют настройки по следующему пути:

*CATALINA\_HOME/conf/ENGINE\_NAME/HOST\_NAME/CONTEXT\_NAME.xml*

Например, если вы устанавливаете веб приложение в контекст it-academy, используя хост по умолчанию с именем localhost и стандартное ядро TomcatCatalina, конфигурационный файл будет располагаться по следующему пути: ***CATALINA\_HOME/conf/Catalina/localhost/.xml***

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

5.1 Установка WAR-архива

***WAR (Веб архив) –*** это архив, содержащий все ресурсы веб-приложение, и может быть установлен в Tomcat.

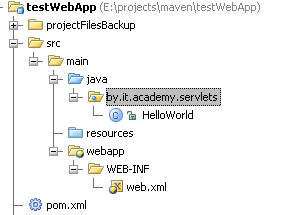


Рисунок 5.1.1 Структура web-проекта

После вызова команды mvncleaninstall происходит сборка WAR файла, который находится в target\testWebApp.war.

Если разархивировать testWebApp.war, то мы получим следующую структуру файлов.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

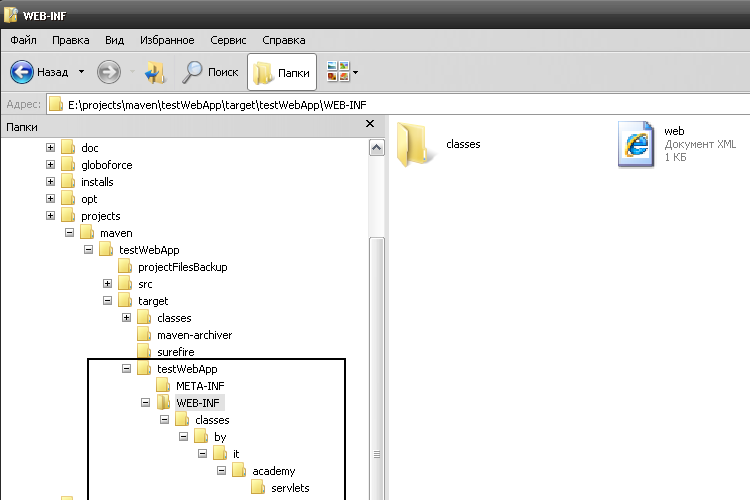


Рисунок 5.1.2Структура WAR

Для установки WAR веб-приложения вы должны скопировать WAR в ***CATALINA\_HOME/webapps***

При установке Tomcat возьмет testWebApp.war, распакует и установит приложение автоматически.

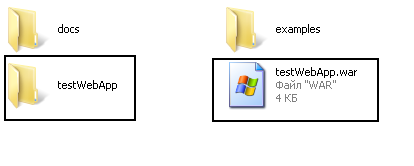


Рисунок 5.1.3Распаковка WAR после установки

После запуска Tomcat, вы можете открыть окно браузера и ввести <http://localhost:8080/testWebApp/hello>.

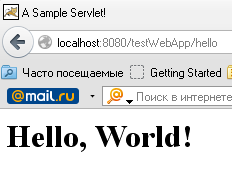


Рисунок 5.1.4 Установленное веб-приложение.

Tomcat распаковывает WAR по причинам производительности, т.к. это значительно быстрее загружать файлы из файловой системы, чем распаковывать файлы из архива каждый раз, когда они необходимы.

Существуют способ настройки Tomcat, при котором он не может распаковывать архив WAR при установке. Для этого необходимо установить значение атрибута unpackWAR<Context> элемента вашего веб-приложения. Следующий пример конфигурации иллюстрирует эту возможность:

<Context path="/testWebApp" docBase="testWebApp" unpackWar="true">

Задание 9

Установите web приложение на Tomcat сервер. Настройте контекст приложения таким образом, что бы war файл автоматически распаковывался при запуске сервером приложений. Проверьте работоспособность вашего приложения.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

5.2 Конфигурация хостов

В архитектуре Tomcat, каждое установленное приложение представляет собой контекст в рамках хоста. Даже без специализированной конфигурации, у Tomcat конфигурации существуют значения по умолчанию, благодаря которым мы могли устанавливать приложения без беспокойства об хостах и контекстах.

***Хосты*** настраиваются в CATALINA\_HOME/conf/server.xml. Если вы посмотрите на этот файл, то увидите, что один хост #1 уже настроен по умолчанию. Имя этого хоста – ***localhost***

<Host#1

name="localhost"#2

appBase="webapps"#3

unpackWARs="true"#4

autoDeploy="true"#5 >

Каждый хост #1 должен иметь имя *name* (#2). Имя хоста обычно указывает на сетевое имя компьютера (сервера), на котором запущен Tomcat, например *www.it-academy.by*. Хост по умолчанию имеет имя *localhost*.

Атрибут *appBase* настраивает *базовую директорию для приложений,* из которой Tomcat загружает и устанавливает веб-приложения (#3).

Следующий атрибут unpackWARs (#4) настраивает возможность автоматического разархивирования WAR при установке (само собой при запуске Tomcat).

Аттрибут autoDeploy (#5), по умолчанию его значение true, настраивает Tomcat на периодическую проверку обновлений WAR файлов и распакованных директорий (но не их контекстов) в базовой директории для приложений.

|  |
| --- |
|  |
|  |

6.3 Конфигурация контекстов

Каждое установленное веб-приложение в Tomcat представляет контекст во внутренней архитектуре Tomcat. Контексты относятся к сконфигурированным хостам, и они настраиваются используя XML элемент (#1).

<Context#1

path="/testWebApp"#2

docBase="testWebApp"#3

reloadable="true"#4 />

*path* указывает контекстный путь (путь для запроса к серверу) для веб-приложения (#2). Этот параметр должен быть установлен, если контекст настроен в server.xml; во всех остальных случаях контекстный путь берется из имени директорий и файлов, где установлено веб-приложение. Контекстный путь должен быть уникальным в рамках хоста.

URI – это набор символов, определяющих ресурс в Интернете. Например, [www.mycompany.com:80/testWebApp](http://www.mycompany.com:80/testWebApp) или

[http:/localhost:8080/testWebApp/index.html?page=2](http://localhost:8080/testWebApp/index.html?page=2)

Часть URI после имени сервера и порта используется для определения контекстного пути в Tomcat.

*docBase* определяет корневой контекст, который является путем к контексту веб-приложения (#3).

*reloadable* (#4). Если он установлен в true, Tomcat будет проверять WEB-INF/classes и WEB-INF/lib директории и корневого контекста, и переустанавливать веб-приложение, если какие-либо изменения обнаружены. Настройка по умолчанию для данного атрибута – false.

Контекст веб-приложения может быть настроен следующим образом:

1. В файле server.xml
2. В каталоге конфигурации хоста:

CATALNA\_HOME/conf/ENGINE\_NAME/HOST\_NAME/CONTEXT\_PATH.xml

1. Как часть кода веб-приложения, в файле /META-INF/context.xml в WAR или распакованной директории
2. Неявно, где Tomcat создает контекст для веб-приложения, для которого ни одна из вышеупомянутых возможностей не подходит.

Когда настраиваете контекст веб-приложения в файле server.xml,вы должны указать все атрибуты, указанные выше.

Вторая опция – это настройка контекста в отдельном xml файле и сохранить его в специальной директории настройки хоста в рамках CATALINA\_HOME/conf. Имя хml файла будет представлять контекстный путь, так что атрибут path устанавливать не нужно. Атрибут docBase должен быть установлен только в том случае, если корневая директория контекста находится за приделами корневой директории хоста.

Третья возможность для конфигурации контекста – это настроить его в/META-INF/context.xml файле в WAR файле или распакованной директории. Этот вариант подходит лишь тогда, когда первые два варианта не настроены. Файлcontext.xml выглядит точно так же как указанный выше.

Четвертый вариант – это вариант по умолчанию. Он относится к веб-приложениям, автоматически установленным в корневом каталоге для веб-приложений ((CATALINA\_HOME/webapps директория в настройках Tomcat по умолчанию). Все распакованные каталоги, установленные в корневом каталоге для веб-приложений, обладают именем контекста, совпадающим с именем каталога, корневой контекстный путь устанавливается путем к каталогу.

|  |
| --- |
|  |
|  |
|  |

5.4 Менеджер веб-приложений Tomcat

***Менеджер веб-приложений*** – это программа, предоставляющая базовую функциональность для управления веб-приложениями, запущенными на сервере Tomcat.Некоторая из представленной функциональности включает возможностьпо установке, запуску, остановке и удалению веб-приложения. Используя менеджер веб-приложений, вы можете легко установить веб-приложение на локальный или удаленный сервер, без необходимости FTP доступа, все команды вызываются с помощью HTTP протокола – используя ваш любимый браузер или командную строку.

Для дополнительного доступа, когда безопасность критична, менеджер веб-приложений может быть доступен через ***Java Management Extension proxy (JMX)***.

***JMX –*** это стандартная Java технология, созданная для управления приложений, устройств, или сетей, используястандартныйJavaAPI.

Существуют следующие контекстные пути для доступа к менеджеру веб- приложений в Tomcat:

1. Веб-приложение на основе html интерфейсе - ***/manager/html***

|  |
| --- |
|  |
|  |

1. Тестовое приложение на основе сценария - ***/manager/text***

|  |
| --- |
|  |
|  |

1. JMX proxy доступ - ***/manager/jmxproxy***

|  |
| --- |
|  |
|  |

1. Запрос статуса - ***/manager/status***

|  |
| --- |
|  |
|  |

5.5 Получение доступа к Менеджеру веб-приложений

Прежде чем вы сможете использовать менеджер веб-приложений, вы должны создать нового пользователя с соответствующими привилегиями для доступа к веб-приложению. Если вы попробуете получить доступ к любой части веб-приложения без установки привилегий безопасности, вы будете переадресованы на страницу для ошибок Tomcatсо стандартным HTML статус-кодом 403-запрещено.

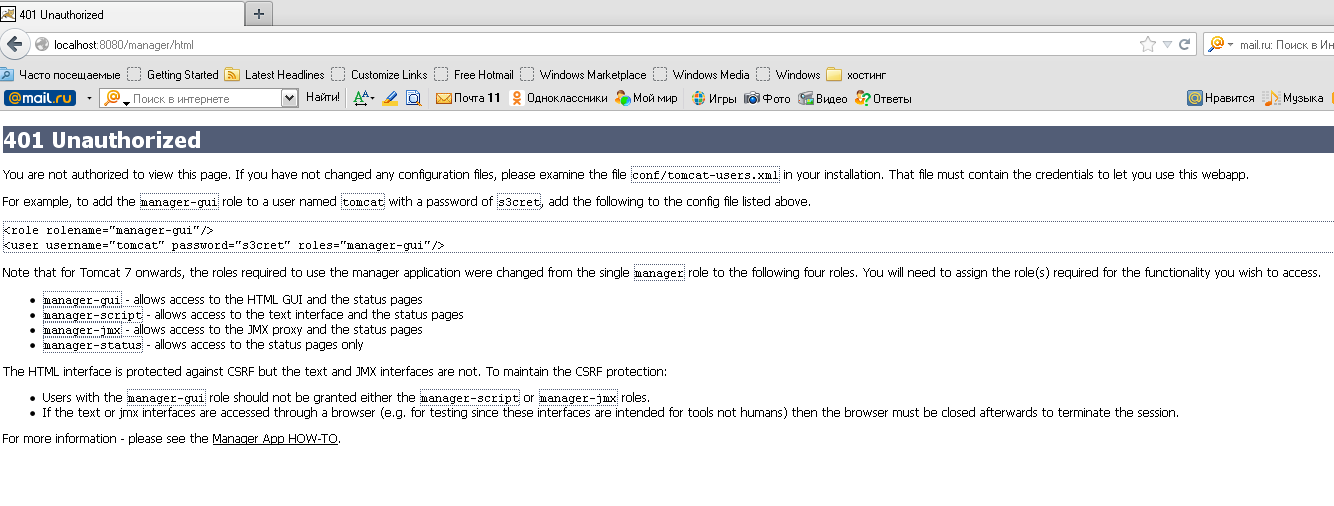


Рисунок 5.5.1 Страница для ошибок, ответ пользователю на попытку доступа к менеджеру веб-приложений без необходимых привилегий.

До Tomcat версии 7.0, была одна роль, необходимая для доступа к менеджеру веб-приложений, которая просто называлась «manager». С 7 версии Tomcat представляет 4 различные роли для доступа менеджера веб-приложений – одна для каждого компонента, упомянутого выше.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

|  |  |  |
| --- | --- | --- |
| ***Компонент*** | ***Требуемая пользовательская роль*** | ***Описание*** |
| Веб-приложение на основе html интерфейсе | manager-gui | Разрешаетдоступ только к HTML интерфейсу и статус странице |
| Тестовое приложение на основе сценария | manager-script | Разрешает доступ только к текстовому скриптовому движку и статус странице |
| JMX proxy доступ | manager-jmx | Разрешает доступтолько к JMX консоле и статус странице |
| Запрос статуса | manager-status | Разрешает доступ только кстатус странице |

Сейчас нам необходимо создать несколько пользователей с необходимыми ролями, так что мы сможем продемонстрировать, что может быть сделано с помощью Tomcat менеджера веб-приложений.

Имена пользователей, пароли и роли сохранены в файле ***CATALINA\_HOME/conf/tomcat-users.xml***. Если вы откроете этотфайл, вы увидите, что он *пуст*. Так мы должны добавить несколько строк вtomcat-users.xml, такмы можем создать пользователей, которые имеют доступ к менеджеру веб-приложений.

<?xml version='1.0' encoding='utf-8'?>

<user username="managerGui" password="abc123" roles="manager-gui"/>#1

<user username="managerScript" password="abc123" roles="manager-script"/>#2

</tomcat-users>

В строчке, отмеченной как #1, мы определили пользователя с именем ***managerGui*** и ролью ***manager-gui***. В следующей строчке (#2), мы создали пользователя ***managerScript*** с ролью ***manager-script***. Пользователь ***managerGui*** имеет доступ к менеджеру веб-приложений через HTML интерфейс, и пользователь имеет доступ к командам, используятестовое приложение на основе сценария. Как было упомянуто выше, оба имеют доступ кстатус странице.

5.6 Доступ к менеджеру веб-приложений, используя веб-интерфейс.

Наиболее распространённый способ доступа менеджера веб-приложений - это использования веб интерфейса. Так как у нас есть пользователь ***managerGui***, с соответствующими привилегиями, вы можете открыть домашнюю страницу менеджера веб-приложений***.***

Сейчас, давайте введем имя пользователя и пароль для пользователя ***managerGui*** с ролью ***manager-gui***. После ввода корректных имени пользователя и пароля мы увидим домашнюю страницу менеджера веб-приложений.

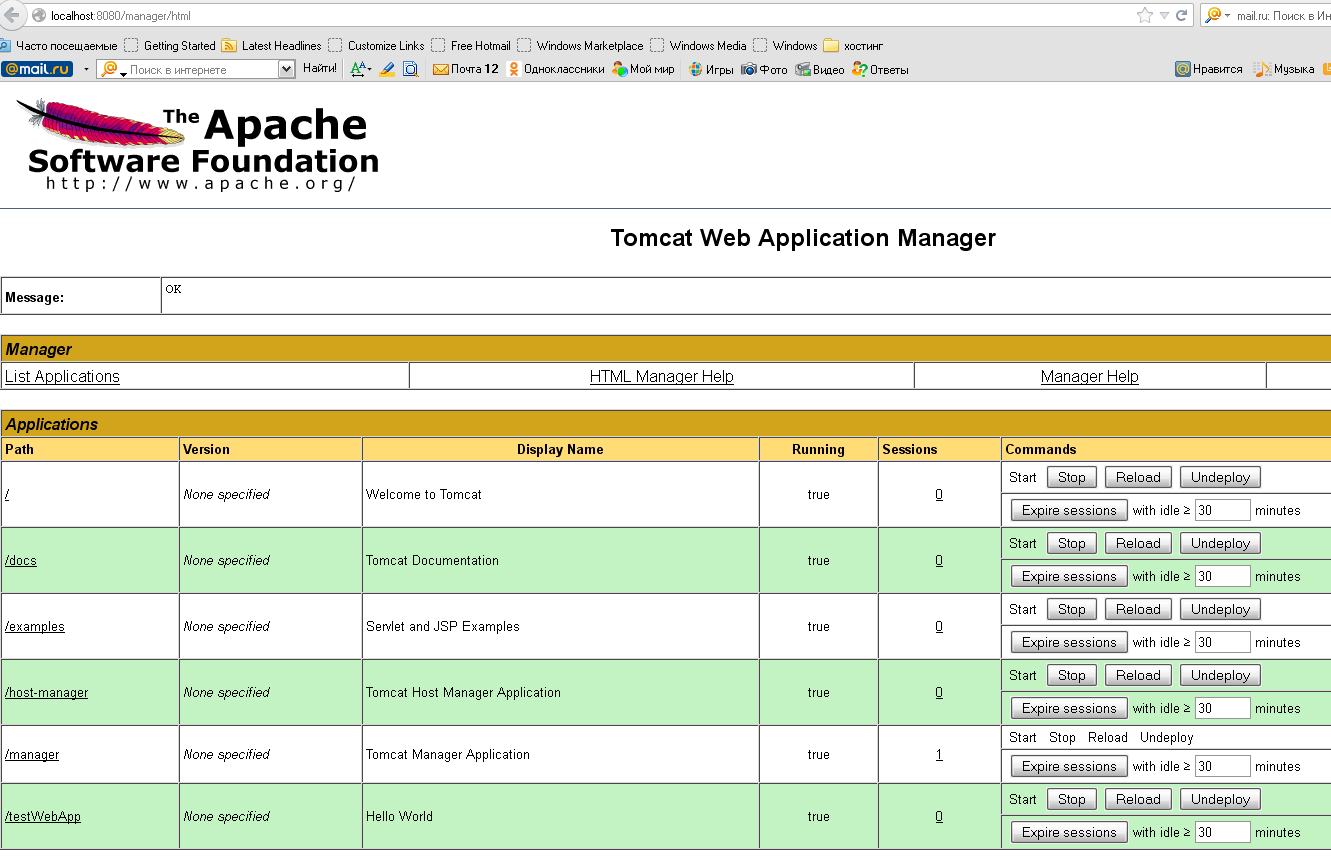


Рисунок 5.6.2 Домашняя страница менеджера веб-приложений

Чуть ниже логотипа Tomcat, находитсяполе с заголовком «Message» и текстом «ОК». В этом поле, вы можете увидеть ответ Tomca tпосле каждого действия. Каждое действие или команда, обработанная менеджером веб-приложения Tomcat,предоставитответ. Если все работает как надо, то ответ – «ОК». В случае ошибки, поле отображает сообщение об ошибки, для быстрого определения проблемы. Когда используете, менеджер веб-приложений, проверьте поле.

Под полем для сообщения, вы увидите таблицу с четырьмя ссылками: списки ссылок приложений, две ссылки на странички помощи, ссылка на страницу статуса.

|  |
| --- |
|  |
|  |
|  |
|  |

5.7 Прослушивание установленных веб-приложений

Список ссылок приложений производят список команд на Tomcat сущности. Результаты команд в списке всех приложений, установленных на сервере. Список команд вызывается по умолчанию, когда используем веб-интерфейс менеджера Tomcat. Когда мы загружаем домашнюю страничку менеджера веб-приложений, мы в действительности вызываем список команд.

Все приложения, установленные на Tomcat, показываются на домашней странице менеджера веб-приложений. На домашней странице мы видим:

1. Список веб-приложений, установленных на сервере со списком команд.
2. Проверка серверного статуса, используя команду статуса.
3. Установка нового веб-приложения в Tomcat, используя командуустановку.
4. Перезагрузка выбранного веб-приложения, используя команду перезагрузки.
5. Проверка серверных сессий для каждого веб-приложения, используя команду сессии.
6. Запустить веб-приложение, используя команду запуска.
7. Остановить запущенное веб-приложение, используя команду остановку.
8. Удаление веб-приложение из сервера, используя команду удаления.

Задание 10

Проверьте функциональность менеджера веб приложений: список веб-приложений, проверка серверного статуса, перезагрузка выбранного веб приложения, проверка серверных сессий, запустить веб-приложений и удаление веб-приложений.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

5.8 Установка нового веб-приложения

Для установки нового веб-приложения используя веб-интерфейс, вы можете использовать HTML форму домашней страницы менеджера веб-приложений. Две HTML формы доступны: одна – для установки веб-приложения для файла или директории, уже доступной на сервере Tomcat, и другой, позволяющий вам установить веб-приложение из WAR файла, расположенного на вашем локальном компьютере.

Наиболее частое использование менеджера веб-приложений для установки на Tomcat – это удаленная установка, где располагается ваше веб-приложение, упакованное в WAR файл, которое вы хотите установить на удаленный сервер Для того, что бы достичь этого, мы будем использовать вторую форму в секции установки на домашней странице менеджера веб-приложений, обозначенной «WAR файл на установку». Все что вы должны сделать – это выбрать файл на вашем диске (нажимая на «Обзор…»), и нажать на кнопку установки под полем выбора. Веб-приложение будет установлено по контекстному пути, соответствующему имени WAR файла, например testWebApp.WAR, контекстный путь - /testWebApp .

Когда выполнение команды завершится, то страница менеджера веб-приложений перезагрузится, и вы увидите сообщение, что операция завершена успешна (просто будет написано «ОК») в поле для сообщений. В дополнение, вы увидите новое, установленное сообщение в списке приложений.

|  |
| --- |
|  |
|  |
|  |

Вы можете также использовать менеджер веб-приложений для установки веб-приложения, расположенного на файловой системе сервера. Это может быть сделано с помощью верхней части формы установки, которая содержит три поля: контекстный путь, адрес (URL) конфигурационного файла XML, адрес (URL) WAR или директории. Первый параметр, контекстный путь, это имя, рекомендуемое для установки. Если вы знаете расположение WAR или адрес директории на вашей локальной системе, то вы должны внести адрес в «WAR или адрес директории» и оставить поля «адрес XML конфигурационного файла XML». Если же у вас есть адрес конфигурационного файла XML, то вы должны сделать наоборот.

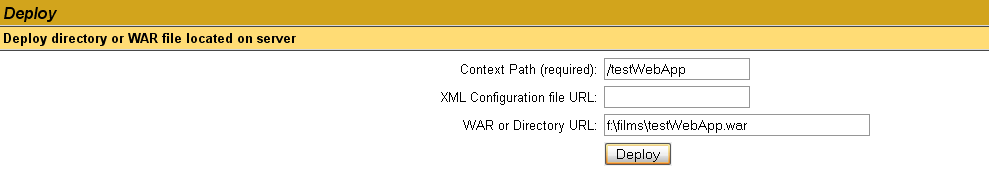


Рисунок 5.8.1 Установка с помощью адреса веб-приложения на файловой системе сервера.

Задание 12

Установите ваше веб приложение, собранное maven, на Tomcat сервер при помощи менеджера веб приложений. Проверьте статус установки при помощи менеджера веб-приложений и Log файлов.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

5.9 Перезагрузка, остановка и запуск существующего веб-приложения

Для перезагрузки существующего приложения, необходимо вызвать *reload* команду. Данная команда перегружает все веб-компоненты, включая сервлеты, JSP, и зависимые классы, ассоциированные с именем веб приложения.



Рисунок 5.9.1 Перезагрузка веб-приложения.

После вызова команды ***reload***, в строке сообщений появится *«OK–Reloadedapplicationatcontextpath /testWebApp»*, что обозначает, что приложение было перегружено успешно.



Рисунок 5.9.2 Сообщение об успешной перезагрузки.

Вlog-файлахдолжнопоявитсяследующаязапись: *«INFO: Reloading Context with name [/testWebApp] is completed»*.

Для того чтобы ***остановить*** веб-приложение необходимо вызвать команду ***stop***.

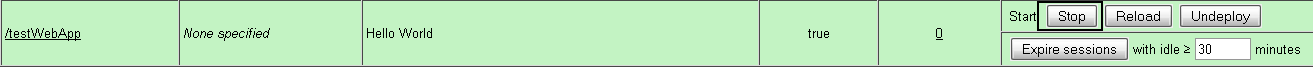


Рисунок 5.9.3 Остановка существующего веб-приложения.

В результате вызова этой команды, в строке сообщения появится *«OK - Stoppedapplicationatcontextpath /testWebApp»*.



Рисунок 5.9.4 Сообщение об успешной остановки.

Если вы попробуете открыть ваше приложение в браузере, например <http://localhost:8080/testWebApp/>, то увидите 404 ошибку (страница не найдена).

|  |
| --- |
|  |

Для ***старта*** веб-приложения необходимо вызвать команду ***start***.



Рисунок 5.9.5 Запусквеб-приложения

Послезапускаприложения, встрокесообщенияпоявится*«OK - Started application at context path /testWebApp»*.



Рисунок 5.9.6 Сообщение об успешном запуске веб-приложения.

Если вы откроете веб-приложение, то увидите, что оно запустилось успешно.

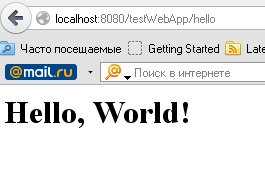


Рисунок 5.9.7 Приложение запустилось успешно.

|  |
| --- |
|  |
|  |

5.10 Управление сессиями веб-приложения

Tomcat связывает каждого клиента с ***sessionid***, и сохраняет параметры в сессии.

Вы можете увидеть число связанных пользовательских сессий для каждого веб-приложения на домашней странице менеджера веб-приложений. Эта информация представлена числом в колонке сессии списка установленных веб-приложений.

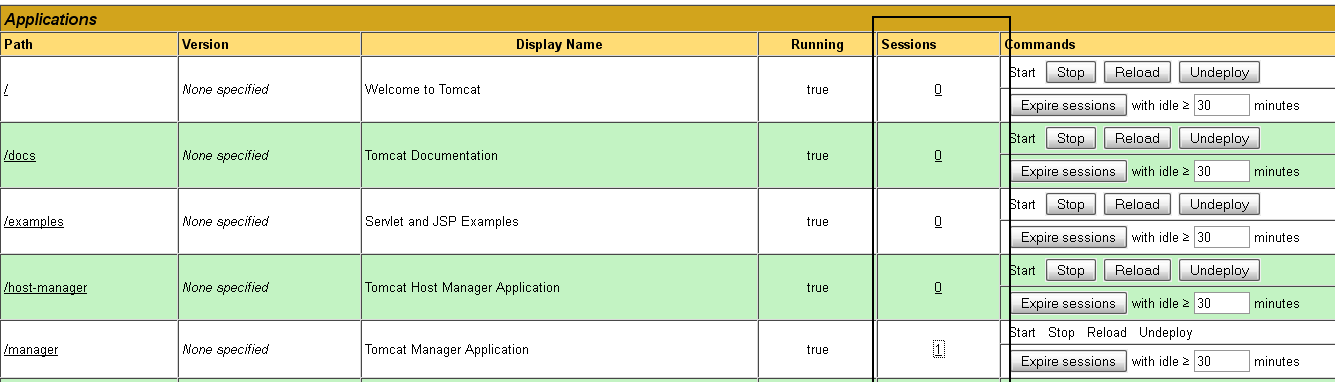


Рисунок 5.10.1 Колонка сессий веб-приложений.

Вы можете изменить время, в течении которого сессия «жива», установив значение в поле рядом с кнопкой «Expiresessions».

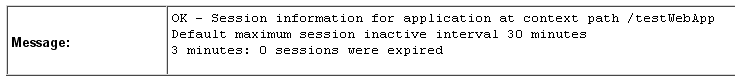


Рисунок 5.10.2 Изменение времени деактивации сессии.

Вы можете получить детальную информацию о каждой сессии, нажав на ссылку (количество сессий) в колонке сессий.



Рисунок 5.10.3 Детальная информация о сессиях.

Если же вы хотите посмотреть полную информацию о конкретной сессии, нажмите на ссылку с id сессии.

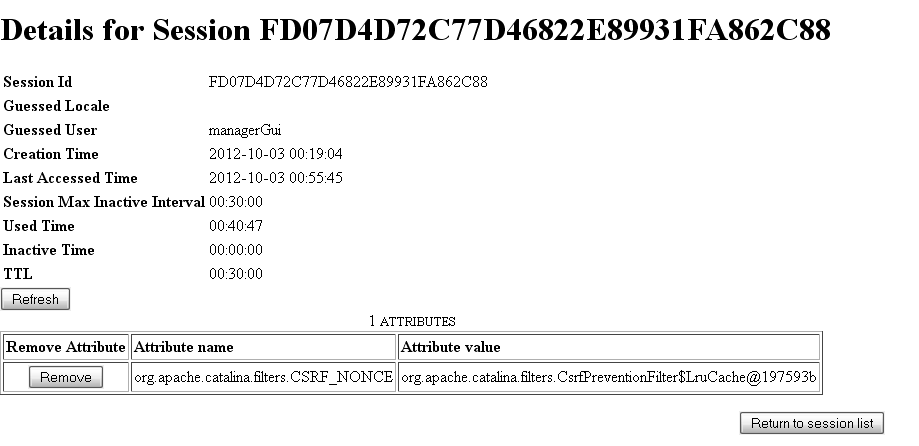


Рисунок 5.10.4 Детальная информация о конкретной сессии.

На странице детальной информации о сессии, находится следующая информация:

1. ***SessionId*** – идентификатор сессии, уникальная строка для сущности Tomcat.
2. ***GuessedLocale –*** пользовательская локаль, если она может быть определена из атрибута запроса.
3. ***GuessedUsername –*** Имя вошедшего в систему пользователя, если он может быть взят из запроса.
4. ***CreationTime –*** Дата и время, когда сессия создана.
5. ***LastAccessedTime –*** Дата и время последнего доступа сессии.
6. ***UsedTime –*** Период времени между созданием и временем последнего использования.
7. ***Inactive –*** Период времени с момента последнего обращения.
8. ***TTL–***Время жизни сессии.

5.11 Tomcat Maven plugin.

Менеджерский текстовый интерфейс может быть вызван напрямую из браузера, путем ввода команды в форме URL, и получения текстового ответа прямо в браузере. Однако, на практике, текстовый доступ к менеджеру веб-приложений Tomcat осуществляется с помощью ***maven***.

Tomcat maven плагин предоставляет возможности для установки и удаления веб-приложения на сервер Tomcat при помощи текстового интерфейса доступа к менеджеру веб-приложений.

Существуют следующие команды работы с этим плагином:

1. ***Переустановка веб-приложения:*** ***mvn tomcat:redeploy***.
2. ***Удаление веб-приложения из сервера Tomcat******: mvn tomcat:undeploy***
3. ***Запуск веб-приложения на сервере Tomcat:*** ***mvn tomcat:start***.

Для того, что бы команда ***start*** выполнилась успешно, необходимо, чтобы веб-приложение было установлено на сервере и остановлено.

1. ***Остановка веб-приложения на сервере Tomcat:*** ***mvn tomcat:stop***.
2. ***Вывод всех текущих сессий веб-приложения:*** ***mvn tomcat:sessions***.
3. ***Вывод статики об установленных веб-приложениях на сервер Tomcat:*** ***mvn tomcat:list***.
4. ***Вывод информации о сервере Tomcat*** ***mvn tomcat:info***

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

Задание 11

Соберите ваше веб приложение при помощи maven. Установите, переустановите, посмотрите статус сервера при помощи Tomcat maven plugin. Опишите проблемы, возникшие при использовании, и попробуйте их решить.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

**тема 6. Основы Subversion**

6.1 Основные понятия управления версиями

***Subversion*** - это централизованная система для совместной работы. В её основе лежит ***хранилище***, которое содержит данные в форме ***дерева файловой системы*** - обычной иерархии файлов и папок.

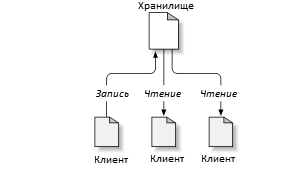


Рисунок 6.1.1 Система клиент-сервер

|  |
| --- |
|  |
|  |

*Subversion запоминает* каждое внесённое изменение, когда-либо записанное в хранилище: любое изменение любого файла, и даже изменения в самом дереве каталогов, такие как добавление, удаление и перемещение файлов и каталогов.Когда клиент просто читает данные из хранилища, он получает *последнюю версию* дерева файловой системы. Но, помимо этого, клиент имеет возможность посмотреть *предыдущие* состояния файловой системы.

Существуют различные модели обмена информацией между пользователями, не мешая друг другу.

***Проблема совместного использования файлов***. Два пользователя редактируют один и тот же файл из репозитория и затем сохраняют его. В результате, история изменений сохранится и в последней версии этого файла в репозитории будут изменения пользователя, который сохранял изменения последним.

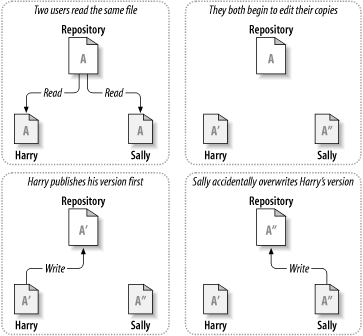


Рисунок 6.1.2 Проблема совместного использования файлов.

Для того что бы избежать данной проблемы, были разработаны следующие модели:

1. ***Модель Блокирование-Изменение-Разблокирование.***

Эта модель реализована следующим образом. Если два пользователя пытаются внести изменения в один и тот же файл в хранилище, то хранилище разрешает вносить изменения только одному человеку за раз. Для того, чтобы пользователь мог внести изменения в хранилище, он должен сначала его ***заблокировать***. При блокировки другой пользователь не сможет внести изменение в хранилище, пока первый не закончит правку. После завершения работы первого пользователя, блокировка снимается и файл доступен для редактирования вторым пользователем (и соответственно его блокировке).

Проблема с моделью блокирование-изменение-разблокирование в ее жестокости, она может создать следующие неудобства пользователям:

1. Блокирование может вызвать административные проблемы.
2. Блокирование может вызвать излишнюю поочерёдность.
3. Блокирование сильно ухудшает производительность.

|  |
| --- |
|  |
|  |

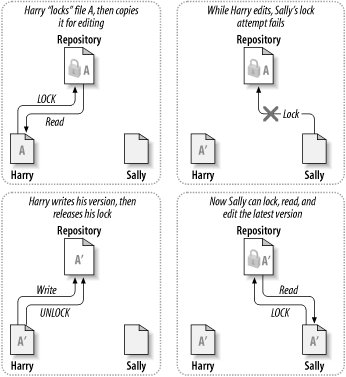


Рисунок 6.1.3 Модель Блокирование-Изменение-Разблокирование

|  |
| --- |
|  |
|  |

1. ***Модель Копирование-Изменение-Слияние.***

Subversion, CVS и другие системы управления версиями используют модель *копирование-изменение-слияние* вместо *блокирования*. В этой модели клиент каждого пользователя считывает из хранилища проект и создаёт персональную рабочую копию - локальное отражение файлов и каталогов хранилища. После этого пользователи работают, одновременно изменяя свои личные копии. В конце концов, личные копии сливаются в новую, финальную версию. Обычно система управления версиями выполняет слияние автоматически, но в общем случае необходимо присутствие человека.

|  |
| --- |
|  |
|  |
|  |

6.2 Установка сервера subversion

Для того, что бы установить сервер subversion, необходимо сделать следующее:

1. Скачиваем файл VisualSVN-Server.msi и запускаем установку
2. В мастере установки указываем, использовать ли для доступа HTTPS, либо просто HTTP. Указываем порт для прослушивания по выбранному протоколу и способ аутентификации. Так же указываем каталог, в котором будут храниться репозитории.
3. После установки открываем ManagementConsole (через Пуск, например) и создаем пользователей и репозитории.

Для управления репозиторием и svn аккаунтами можно применить ***Visual SVN Management Console***.

Необходимо создать нового пользователя. Для этого выберем папку **Users** в левой части окна менеджера. При нажатии на правую кнопку мыши появится контекстное меню, в котором можно выбрать пункт «создать пользователя».

1. Далее вы вводите username, password и confirm password.

Для создания репозитория необходимо в левой части окна менеджера выбрать пункт ***«Repositories»*** и нажать на правую кнопку мыши. В появившемся контекстном меню нужно выбрать ***«Create New Repository»***.

1. Создания репозитория.

Нужно ввести имя нового репозитория и установить галочку на пункте «создать структуру по умолчанию» для стандартизации структуры проекта с общепринятым подходом.

После завершения операции, репозиторий будет доступен либо через svn-клиент, либо через браузер.

Задание 12

Установите subversion сервер. Создайте новое хранилище. Создайте пользователей. Проверьте доступ к хранилищу через ваш браузер.

|  |
| --- |
|  |
|  |
|  |

6.3 Установка клиента subversion и его использование

В качестве клиента subversion мы будем использовать TortoiseSVN. ***TortoiseSVN*** — это бесплатный Windows-клиент с открытыми исходным кодом для системы управления версиями ApacheSubversion. То есть TortoiseSVN управляет файлами и директориями во времени.

Tortoise SVN поставляется в виде простого в использовании установочного файла. Для запуска установки необходимо выполнить этот файл.

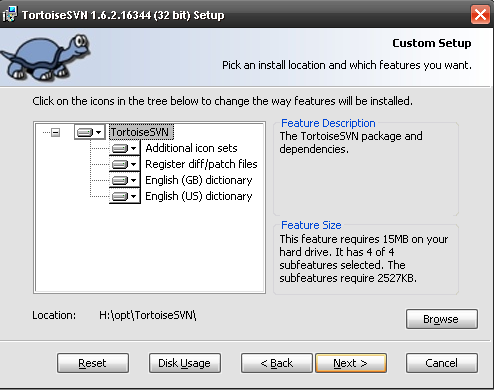


Рисунок 6.3.1 Задание пути установки TortoiseSVN

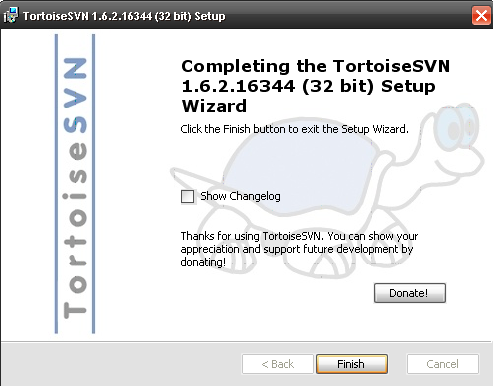


Рисунок 6.3.2 Окончание установки TortoiseSVN

После завершения установки вам **необходимо** будет ***перезагрузить*** ваш компьютер.

Для начала работы с клиентом вам необходимо открыть проводник Windows, а затем, нажимая правую клавишу мыши, откроется контекстное меню, в котором вы можете работать с функциями клиента TortoiseSVN.

|  |
| --- |
|  |
|  |
|  |

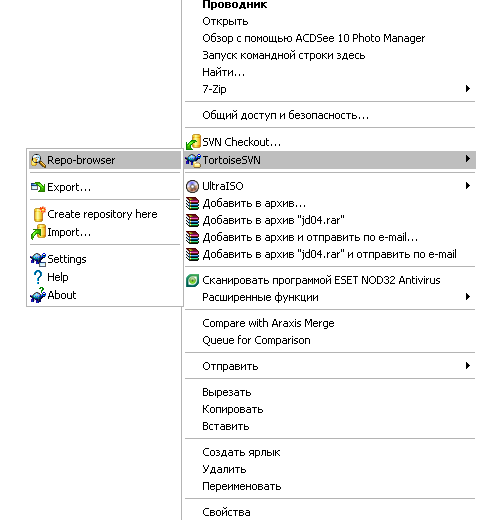


Рисунок 6.3.3 Доступ к функциям клиента.

У нас есть хранилище <https://ivan:8443/svn/ivelon/trunk/>, но оно совершенно пустое в данный момент. У нас есть проект testWebApp, который мы хотим поместить в это хранилище для совместного пользования.

|  |
| --- |
|  |
|  |

Что нужно, что бы поместить testWebApp под управление subversion?

Перейдите к папке testWebApp в проводнике и вызовите ***TortoiseSVN>Import***.

|  |
| --- |
|  |
|  |

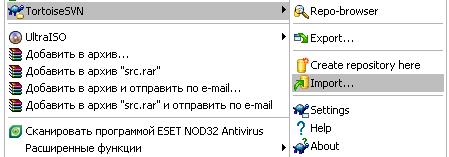


Рисунок 6.3.5 Функция импорта кода в репозиторий.

Далее откроется диалог импорта, в котором нужно в строке адреса репозитория внести адрес используемого хранилища <https://ivan:8443/svn/ivelon/trunk/>, а в строке сообщения – поясняющее сообщение, что бы потом понять для чего вносились файлы в хранилище.

При проведении операции, клиент спросит имя и пароль к хранилищу. Вам необходимо будет ввести имя пользователя и пароль, а также поставить галочку напротив пункта меню *«сохранить пароль»,* что бы не вводить их каждый раз.

В результате вы получите сообщение о помещенных файлах в репозиторий.

|  |
| --- |
|  |
|  |
|  |

6.4 Сохранение своих изменений в репозиторий

Если Вы хотите внести свои изменения в скачанный репозиторий и у Вас есть на это право (право коммита в репозиторий), то просто начинайте работать с локальной рабочей копией как с обычной папкой: создавайте, редактируйте, удаляйте файлы и/или папки.

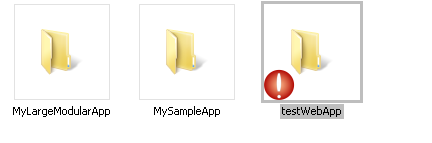


Рисунок 6.4.1 Измененные файлы.

Теперь чтобы внести изменения в хранилище, нужно:

1. Заходим в папку, в которой находятся файлы скачанной рабочей копии в Проводнике Windows. Внимание! Рабочая копия должна быть скачана от логина пользователя, у которого есть право на коммит изменений (коммит-флаг).
2. Нажимаем правой кнопкой по свободному месту в папке, выбираем «SVNCommit…».
3. В открывшемся окне «Commit» в секции «Message» вводим краткое описание своих изменений, но описание можно и оставить пустым (не рекомендуется, т.к. другие разработчики или пользователи не смогут понять что именно было изменено).
4. В секции «Changesmade (double-clickonfilefordiff)» Вы увидите список всех изменённых, добавленных и удалённых Вами файлов локальной рабочей копии. У изменённых файлов будут автоматически стоять флажки, у остальных — нет.
5. После выделения нужных файлов или всех сразу («Select / deselectall»), нажмите кнопку «OK» и дождитесь окончания загрузки изменений на сервер. Будет открыто окно со списком добавленных, изменённых и/или удалённых файлов, а также указан текущий номер ревизии.



Рисунок 6.4.2 Результат операции «commit».

Задание 13

Скопируйте веб-проект в хранилище, созданное ранее. Затем скачайте (checkout) изменения в отдельный каталог. Попробуйте изменить что-либо в вашем проекте ипосмотрите на статус папок в проводнике Windows.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

Задание 14

Выполните задание 13 для другой локальной папки, сделайте изменения кода в двух папках, произведите операции обновления и фиксирования изменений локальных папок и разрешите возникший конфликт.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

6.5 Работа с репозиторием напрямую на сервере

Иногда требуется работать с svn-репозиторием напрямую на сервере не скачивая его на локальный компьютер. Для этих целей существует утилита «SVN Repo-browser», также входящая в состав TortoiseSVN.

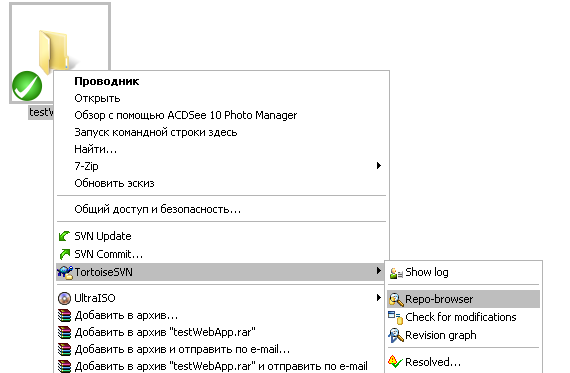


Рисунок 6.5.1 Операция «Repo-browser»

1. Запускаем проводник Windows, заходим в любую папку, нажимаем правой кнопкой по свободному месту и выбираем «SVNRepo-browser».
2. В открывшемся окне «URL» вводим полный URL нужного нам svn-репозитория и нажимаем «OK».

Работать в этом окне можно так же, как в Проводнике Windows. Разрешено перетаскивание объектов (DragandDrop). Через контекстное меню файла/папки можно:

1. удалять «Delete…», переименовывать «Rename» файлы и каталоги (папки);
2. добавить файл («Addfile…») или папку («Addfolder…») в репозиторий;
3. создать пустую папку («Createfolder») в репозитории;
4. скачать («Checkout…») репозиторий;
5. извлечь файл или папку из репозитория без служебных каталогов («Export…»)

Извлечённая папка отличается от рабочей копии тем, что её нельзя обновлять и она не содержит служебных скрытых каталогов .svn;

1. создать ветку или тег («Copyto…»);
2. изменять свойства файла или папки («Showproperties»).

Любое изменение требует ввести текст комментария (разрешается не вводить и оставить поле пустым) и увеличивает номер ревизии на единицу. Изменения применяются немедленно.

1. Нажмите кнопку «OK» для выхода из RepositoryBrowser.

Задание 15

Выполните задание 13 для другой локальной папки, сделайте изменения кода в двух папках, произведите операции обновления и фиксирования изменений локальных папок и разрешите возникший конфликт.

|  |
| --- |
|  |
|  |
|  |

6.6 Откат изменений в рабочей копии

Если работая с локальным репозиторием Вы совершили какую-либо ошибку, например, повредили какой-то файл, неправильно отредактировали и т.д., то это легко исправить:

1. Заходим в папку, в которой находятся файлы скачанной рабочей копии в Проводнике Windows.
2. Нажимаем правой кнопкой по свободному месту и выбираем «TortoiseSVN» — «Revert... »
3. В открывшемся окне Вы увидите список всех изменённых файлов Вашей рабочей копии. Установите флажок у того файла, который Вы хотите откатить до версии из репозитория, либо выберите сразу все файлы («Select / deselectall»). Двойной щелчок по файлу покажет его отличия от оригинального файла.

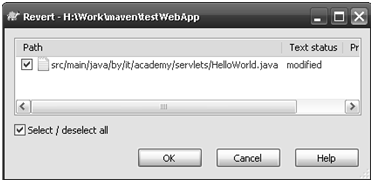


Рисунок 6.6.1 Откат изменений.

1. Нажмите «OK«. Выбранные файлы будут восстановлены, а изменённые будут отправлены в Корзину Windows.

Задание 16

Внесите изменения в ваш веб проект, например, измените текст на jsp. И затем выполните откат изменений и проверьте статус выполнения операции.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

6.7 Работа с утилитой LogViewer

Утилита LogViewer, запускаемая после нажатия на кнопку «Showlog…», либо при нажатии правой кнопкой по свободному месту в локальной рабочей копии — «TortoiseSVN» — «Showlog». По умолчанию утилита показывает последние 50 ревизий и их описаний (если во время коммита они были введены). Все основные действия выполняются из контекстного меню ревизии:

1. «Compare with woking copy» — сравнитьтекущую(выделенную) ревизию с локальной рабочей копией. Список изменений будет показан в TortoiseMerge.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

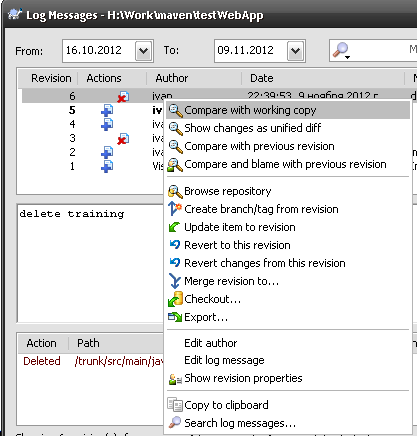


Рисунок 6.7.1 – Контекстное меню утилиты LogViewer

1. «Show changes as unified diff» — показать изменения выделенных ревизий как унифицированный патч-файл (unifieddiff). Если выделена только одна ревизия, то она будет сравниваться с предыдущей.
2. «Compare with previous revision» — показать изменения выделенной и предыдующей ревизии. Будет отображено окно с различающимися файлами, двойной щелчок по файлу откроет его в TortoiseMerge для просмотра изменений.
3. «Compare and blame with previous revision» — то же, что и предыдущее, но здесь отображается – кто и когда внёс какое-либо изменение в файл построчно.
4. «Browse repository» — открывает описанный выше RepositoryBrowser для прямой работы с репозиторием.
5. «Create branch/tag from revision» — позволяет создать ветку или тег из выделенной ревизии (см. выше).
6. «Update item to revision» — обновляет до выделенной ревизии.
7. «Revert to this revision» — позволяет откатиться до выделенной ревизии.
8. «Revert changes from this revision» — позволяет откатить все изменения, сделанные в выделенной ревизии.
9. «Merge revisions to…» — позволяет объединить несколько ревизий в рабочую копию.
10. «Checkout…» — скачать выделенную ревизию.
11. «Export…» — извлечь выделенную ревизию без служебных каталогов.
12. «Edit author» — редактировать имя автора выделенной ревизии (нужны права администратора).
13. «Edit log message» — позволяет редактировать текст описания выделенной ревизии (нужны права администратора).
14. «Show revision properties» — показать свойства выделенной ревизии.
15. «Copy to clipboard» — скопировать номер выделенной ревизии, автора, описание и список изменённых файлов в буфер обмена Windows.
16. «Search log messages» — запустить поиск в описаниях ревизий.

Кнопка «Statistics» отображает различную статистическую информацию: наиболее активный и наименее активный автор, количество изменений в репозитории в неделю, общее количество изменений и т.д.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |