Деплой проекта на Tomcat с использованием Maven

Установка:

1. В папку /Users/alena/.m2 добавляется файл settings.xml

<?xml version="1.0" encoding="UTF-8"?>

<settings>

<servers>

<server>

<!--

id это просто идентификатор, используете удобное для вас

название!

Я называю так, чтобы не запутаться,

т.к. у меня есть еще сервера других версий и других целей.

-->

<id>tomcat</id>

<!-- имя пользователя, которое вы указали в tomcat-users.xml -->

<username>admin</username>

<!-- ну и естественно его пароль из tomcat-users.xml -->

<password>1234</password>

</server>

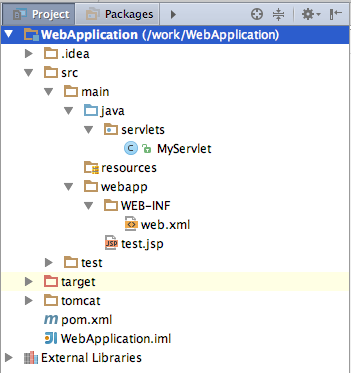
</servers>

</settings>

2. Скачивается Tomcat 7, в файл tomcat/conf/tomcat-users.xml добавляется пользователь с ролями, который указывается в settings.xml

<role rolename="tomcat"/>  
<role rolename="manager-gui"/>  
<role rolename="manager-script"/>  
<role rolename="admin-gui"/>  
<user username="admin" password="1234" roles="tomcat,manager-gui,admin-gui,manager-script"/>

3. Создается проект со следующей структурой:



4. В файле pom.xml указываются плагины tomcat7-maven-plugin, maven-compiler-plugin, maven-war-plugin и зависимости javax.servlet, javaee, javax, :

*<?*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>WebApplication</groupId>  
 <artifactId>WebApplication</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>war</packaging>  
  
 <dependencies>  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>servlet-api</artifactId>  
 <version>3.0-alpha-1</version>  
 <scope>provided</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>javaee</groupId>  
 <artifactId>javaee-api</artifactId>  
 <version>5</version>  
 </dependency>  
  
 <dependency>  
 <groupId>javax</groupId>  
 <artifactId>javaee-web-api</artifactId>  
 <version>6.0</version>  
 <scope>provided</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <configuration>  
 <source>1.5</source>  
 <target>1.5</target>  
 </configuration>  
 </plugin>  
  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-war-plugin</artifactId>  
 <version>2.1.1</version>  
 <configuration>  
 <failOnMissingWebXml>false</failOnMissingWebXml>  
 </configuration>  
 </plugin>  
  
 <plugin>  
 <groupId>org.apache.tomcat.maven</groupId>  
 <artifactId>tomcat7-maven-plugin</artifactId>  
 <configuration>  
 *<!-- id-шник из settings.xml -->* <server>tomcat</server>  
 <url>http://localhost:8080/manager/text</url>  
 <path>/app</path>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

5. В файле web.xml указываем следующее:

<web-app xmlns="http://java.sun.com/xml/ns/javaee"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
 http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"  
 version="3.0"  
 metadata-complete="true">  
 <display-name>Archetype Created Web Application</display-name>  
  
 <servlet>  
 <servlet-name>myServlet</servlet-name>  
 <servlet-class>servlets.MyServlet</servlet-class>  
 </servlet>  
 <servlet-mapping>  
 <servlet-name>myServlet</servlet-name>  
 <url-pattern>/myServlet</url-pattern>  
 </servlet-mapping>  
  
</web-app>

6. Далее с помощью maven выполняем следующие команды:

mvn clean

mvn install

(должен создаться .war файл проекта)

После запускаем Tomcat: выполняем команду в командной строке:

cd /work/WebApplication/tomcat/bin

sh startup.sh

(остановка сервера sh shutdown.sh)

mvn tomcat7:deploy (tomcat7:redeploy) – деплой проекта в Tomcat. В папке Tomcat /webapps должны появится папка с нашим проектом и .war файл.

7. После в браузере по адресу <http://localhost:8080/manager/html> можно увидеть запущенные приложения, наше приложение: <http://localhost:8080/app/myServlet>

8. Поскольку говно из пункта 6 выполнять руками долго, создаем .sh файл возле pom.xml для деплоя проекта

sudo sh /Applications/'IntelliJ IDEA 15 CE.app'/Contents/plugins/maven/lib/maven3/bin/mvn clean  
sudo sh /Applications/'IntelliJ IDEA 15 CE.app'/Contents/plugins/maven/lib/maven3/bin/mvn install  
sudo sh /work/WebApplication/tomcat/bin/startup.sh  
sudo sh /Applications/'IntelliJ IDEA 15 CE.app'/Contents/plugins/maven/lib/maven3/bin/mvn tomcat7:redeploy

9. Дебаг сервлета: в файле catalina.sh добавляем

export JPDA\_ADDRESS=8000  
export JPDA\_TRANSPORT=dt\_socket

Теперь вместо файла запуска tomcat startup.sh используем catalina.sh jpda start

Немного теории

* Сервлеты – классы на Java, которые позволяют обрабатывать клиентские запросы и возвращать ответы.
* JSP – файл, который после преобразования представляет из себя сервлет. Содержит статическую (html) и динамическую (код) части.
* В адресной строке после ? можно указывать параметры для сервлета:

/myServlet?param=value

HTML

Основные элементы:

1. <!DOCTYPE> предназначен для указания типа текущего документа — DTD (document type definition, описание типа документа). Это необходимо, чтобы браузер понимал, как следует интерпретировать текущую веб-страницу, ведь HTML существует в нескольких версиях, кроме того, имеется XHTML (EXtensible HyperText Markup Language, расширенный язык разметки гипертекста), похожий на HTML, но различающийся с ним по синтаксису.
2. <html> определяет начало HTML-файла, внутри него хранится заголовок (<head>) и тело документа (<body>)
3. <meta> является универсальным и добавляет целый класс возможностей, в частности, с помощью метатегов можно изменять кодировку страницы, добавлять ключевые слова, описание документа и многое другое. Например, механизмы поисковых систем обращаются к метатегам для получения описания сайта, ключевых слов и других данных.
4. <title> определяет заголовок веб-страницы, это один из важных элементов предназначенный для решения множества задач.
5. <h1> представляет собой наиболее важный заголовок первого уровня, а тег <h6> служит для обозначения заголовка шестого уровня и является наименее значительным.
6. <p> определяет абзац (параграф) текста. Если закрывающего тега нет, считается, что конец абзаца совпадает с началом следующего блочного элемента.

Типы тегов: теги верхнего уровня; теги заголовка документа;блочные элементы; строчные элементы; универсальные элементы; списки; таблицы; фреймы.

1. Теги верхнего уровня (html, head, body) – определяют структуру документа.
2. Теги заголовка документа(head – title, meta, link)

<link> - с помощью этого тега можно подключить таблицу стилей

<link rel="stylesheet" href="style.css">

1. Блочные элементы характеризуются тем, что занимают всю доступную ширину, высота элемента определяется его содержимым, и он всегда начинается с новой строки.

<blockquote> Предназначен для выделения длинных цитат внутри документа. Текст, обозначенный этим тегом, традиционно отображается как выровненный блок с отступами слева и справа (примерно по 40 пикселов), а также с пустым пространством сверху и снизу.

<div> относится к универсальным блочным контейнерам и применяется в тех случаях, где нужны блочные элементы без дополнительных свойств. Также с помощью тега <div> можно выравнивать текст внутри этого контейнера с помощью атрибута align.

<h1>,...,<h6> Эта группа тегов определяет текстовые заголовки разного уровня, которые показывают относительную важность секции, расположенной после заголовка.

<hr> Рисует горизонтальную линию, которая по своему виду зависит от используемых атрибутов. Линия всегда начинается с новой строки, а после нее все элементы отображаются на следующей строке.

<p> Определяет параграф (абзац) текста.

<pre> Задает блок предварительно форматированного текста. Такой текст отображается обычно моноширинным шрифтом и со всеми пробелами между словами. В HTML любое количество пробелов идущих в коде подряд на веб-странице показывается как один. Тег <pre> позволяет обойти эту особенность и отображать текст как требуется разработчику.

1. Строчные элементы - являются непосредственной частью другого элемента, например, текстового абзаца. В основном они используются для изменения вида текста или его логического выделения.

<a>предназначен для создания ссылок. В зависимости от присутствия атрибутов name или href тег <a> устанавливает ссылку или якорь.

<b> Определяет жирное начертание шрифта.

<big> увеличивает размер шрифта на единицу по сравнению с обычным текстом. В HTML размер шрифта измеряется в условных единицах от 1 до 7, средний размер текста, используемый по умолчанию, принят 3. Таким образом, добавление тега <big> увеличивает текст на одну условную единицу.

<br> устанавливает перевод строки в том месте, где этот тег находится. В отличие от тега параграфа <p>, использование тега <br> не добавляет пустой отступ перед строкой.

<em> предназначен для акцентирования текста. Браузеры отображают такой текст курсивным начертанием.

<i> Устанавливает курсивное начертание шрифта.

<img> предназначен для отображения на веб-странице изображений в графическом формате GIF, JPEG или PNG. Если необходимо, то рисунок можно сделать ссылкой на другой файл, поместив тег <img> в контейнер <a>. При этом вокруг изображения отображается рамка, которую можно убрать, добавив атрибут border="0" в тег <img>.

<small> уменьшает размер шрифта на единицу по сравнению с обычным текстом. По своему действию похож на тег <big>, но действует с точностью до наоборот.

<span> Универсальный тег, предназначенный для определения строчного элемента внутри документа.

<strong> предназначен для акцентирования текста. Браузеры отображают такой текст жирным начертанием.

<sub> Отображает шрифт в виде нижнего индекса. Текст при этом располагается ниже базовой линии остальных символов строки и уменьшенного размера — H2O.

<sup> Отображает шрифт в виде верхнего индекса. По своему действию похож на <sub>, но текст отображается выше базовой линии текста — м2.

1. Универсальные элементы - могут использоваться как блочные или встроенные элементы.

<del> используется для выделения текста, который был удален в новой версии документа. Подобное форматирование позволяет отследить, какие изменения в тексте документа были сделаны. Браузеры обычно помечают текст в контейнере <del> как перечеркнутый.

<ins> предназначен для акцентирования вновь добавленного текста и обычно применяется наряду с тегом <del>. Браузеры помечают содержимое контейнера <ins> подчеркиванием текста.

1. Теги для списков

<ol> устанавливает нумерованный список, т.е. каждый элемент списка начинается с числа или буквы и увеличивается по нарастающей.

<ul> Устанавливает маркированный список, каждый элемент которого начинается с небольшого символа — маркера.

<li> определяет отдельный элемент списка. Внешний тег <ul> или <ol> устанавливает тип списка — маркированный или нумерованный.

<dd>, <dt>, <dl> Тройка элементов предназначена для создания списка определений. Каждый такой список начинается с контейнера <dl>, куда входит тег <dt> создающий термин и тег <dd> задающий определение этого термина. Закрывающий тег </dd> не обязателен, поскольку следующий тег сообщает о завершении предыдущего элемента. Тем не менее, хорошим стилем является закрывать все теги.

1. Теги для таблиц

<table> Служит контейнером для элементов, определяющих содержимое таблицы. Любая таблица состоит из строк и ячеек, которые задаются с помощью тегов <tr> и <td>.

<td> Предназначен для создания одной ячейки таблицы. Тег <td> должен размещаться внутри контейнера <tr>, который в свою очередь располагается внутри тега <table>.

<th> предназначен для создания одной ячейки таблицы, которая обозначается как заголовочная. Текст в такой ячейке отображается браузером обычно жирным шрифтом и выравнивается по центру.

<tr> служит контейнером для создания строки таблицы.

1. Теги для фреймов

*Фреймы* разделяют окно браузера на отдельные области, расположенные вплотную друг к другу. В каждую из таких областей загружается самостоятельная веб-страница определяемая с помощью тега <frame>. С помощью фреймов веб-страница делится на два или более документа, которые обычно содержат навигацию по сайту и его контент. Механизм фреймов позволяет открывать документ в одном фрейме, по ссылке, нажатой в совершенно другом фрейме. Допустимо также использовать вложенную структуру элементов, это позволяет дробить фреймы на мелкие области.

<frame> определяет свойства отдельного фрейма, на которые делится окно браузера.

<frameset> заменяет собой элемент <body> на веб-странице и формирует структуру фреймов.

<iframe> создает плавающий фрейм, который находится внутри обычного документа, он позволяет загружать в область заданных размеров любые другие независимые документы.

1. Якоря

Якорем называется закладка с уникальным именем на определенном месте веб-страницы, предназначенная для создания перехода к ней по ссылке. Якоря удобно применять в документах большого объема, чтобы можно было быстро переходить к нужному разделу.

<a name="top">

<a href="#top">Наверх</a>