Развертывание проекта с помощью Maven

JSP & Servlets
2 уровень, 4 лекция

ОТКРЫТА

Используем maven-deploy-plugin

И еще одна очень интересная тема — автоматический deploy собранного пакета. Допустим, мы собрали собственную библиотеку с помощью Maven. Как нам автоматически выложить ее в локальный, корпоративный или центральный Maven-репозиторий?

Для этого у Maven есть специальный плагин **maven-deploy-plugin**. Пример:

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-deploy-plugin</artifactId>
    <version>2.5</version>
    <configuration>
      <file>${project.build.directory}\${project.artifactId}-src.zip</file>
      <url>${project.distributionManagement.repository.url}</url>
      <repositoryId>${project.distributionManagement.repository.id}</repositoryId>
      <groupId>${project.groupId}</groupId>
      <artifactId>${project.artifactId}</artifactId>
      <version>${project.version}</version>
      <packaging>zip</packaging>
      <pomFile>pom.xml</pomFile>
    </configuration>
</plugin>
```

С этими настройками ты можешь выложить собранную библиотеку в Maven-репозиторий как валидный пакет. Подробно разбирать этот процесс мы не будем, но кратко рассмотрим, что тут происходит:

Тег file задает файл, который будет отправлен в репозиторий Maven как новая библиотека.

Тег url — путь к репозиторию Maven (локальный/корпоративный/…).

Тег repositoryId задает идентификатор репозитория, в который будет производиться депллой.

Теги groupId, artifactId, version задают стандартную идентификацию пакета в репозитории Maven. Именно по этим трем параметрам можно однозначно идентифицировать библиотеку.

Тег packaging используется для того, чтобы результат был отправлен одним zip-файлом. Если его не указать, то будет один jar-файл, даже если у вас было несколько jar-файлов.

Тег pomFile опциональный и позволяет отправить в репозиторий другой pom.xml, который не содержит скрытых или служебных данных.

Развертывание web-приложения в Tomcat с помощью Maven

Самый популярный веб-сервер для веб-приложений на Java — это **Apache Tomcat**. И конечно же с помощью Maven можно деплоить war-файлы прямо в локальный или даже удаленный Tomcat-сервер.

Устанавливать и настраивать Tomcat мы научимся как-нибудь потом, сейчас же коснемся только темы автоматического деплоя нашего web-приложения.

Шаг первый. Мы должны дать доступ Maven'у к Tomcat-серверу. Для этого открываем файл **conf/tomcat-users.xml** в директории, где распакован Apache Tomcat, и добавляем роли **manager-gui** и **manager-script**:

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="manager-gui"/>
  <role rolename="manager-script"/>
  <user username="admin" password="admin" roles="manager-gui,manager-script" />
</tomcat-users>
```

Шаг второй. Разрешаем доступ Maven к Tomcat. Для этого открываем файл **$MAVEN_HOME/conf/settings.xml** и добавляем сервер:

```
<?xml version="1.0" encoding="UTF-8"?>
<settings ...>
  <servers>
    <server>
    <id>TomcatServer</id>
    <username>admin</username>
    <password>admin</password>
    </server>
  </servers>
</settings>
```

Шаг третий. Добавляем специальный плагин для автоматизированного деплоя нашего приложения в Apache Tomcat. Плагин называется **tomcat7-maven-plugin**. Кстати, создан он не разработчиками Maven, а разработчиками Tomcat, о чем можно догадаться по имени.

```
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.tomcat.maven</groupId>
                <artifactId>tomcat7-maven-plugin</artifactId>
                <version>2.2</version>
                <configuration>
                    <url>http://localhost:8080/manager/text</url>
                    <server>TomcatServer</server>
                    <path>/simpleProject</path>
                </configuration>
            </plugin>
        </plugins>
    </build>
```

В разделе configuration указываем:

- url — адрес, где запущен Tomcat, и путь к **manager/text**
- server — id сервера из файла **settings.xml**
- path — адрес, по которому будет доступно развертываемое приложение

Команды для управления деплоем:

mvn tomcat7:deploy	Выполнить деплой приложения в Tomcat
mvn tomcat7:undeploy	Удалить приложение из Tomcat
mvn tomcat7:redeploy	Передеплоить приложение

Деплой с помощью плагина Cargo Plugin

Еще один полезный и универсальный плагин для деплоя веб-приложений — **Cargo Plugin**. Он умеет работать с разными типами веб-серверов. Вот как произвести деплой с его помощью в Apache Tomcat:

```
<build>
    <plugins>
        <plugin>
            <groupId>org.codehaus.cargo</groupId>
            <artifactId>cargo-maven2-plugin</artifactId>
            <version>1.9.10</version>
            <configuration>
                <container>
                    <containerId>tomcat8x</containerId>
                    <type>installed</type>
                    <home>Insert absolute path to tomcat 7 installation</home>
                </container>
                <configuration>
                    <type>existing</type>
                    <home>Insert absolute path to tomcat 7 installation</home>
                </configuration>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Чтобы выполнить установку веб-приложения в локальный Tomcat, нужно просто выполнить команды:

```
mvn install
mvn cargo:deploy
```

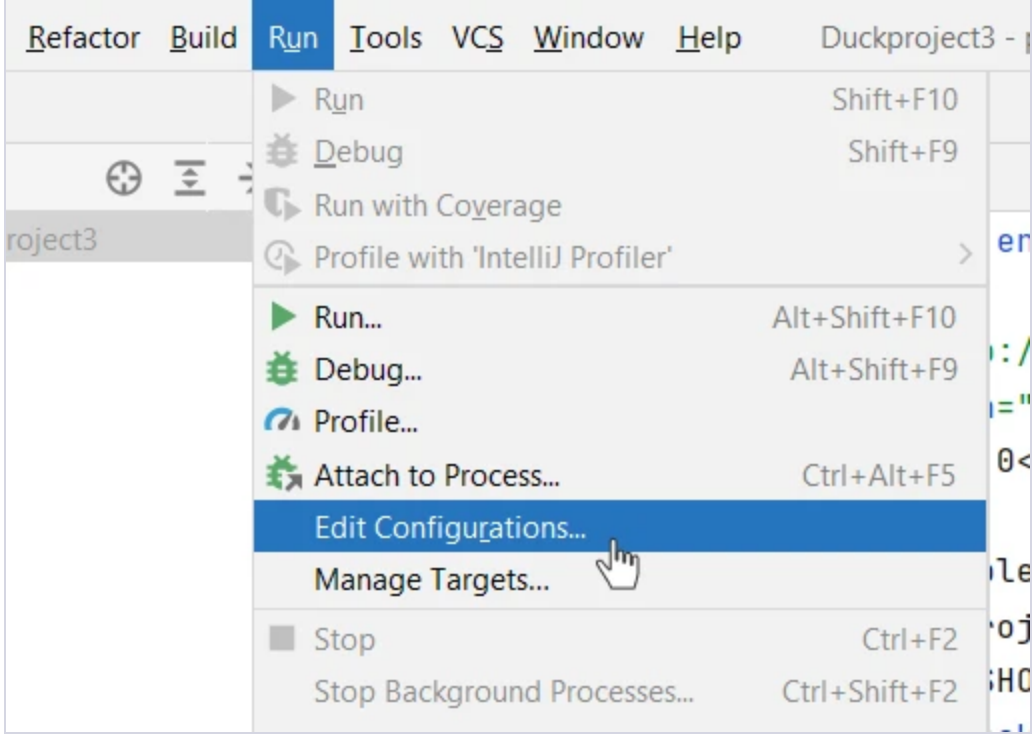
Если мы хотим выполнить деплой на удаленный веб-сервер, то тут уже придется настроить права доступа к этому серверу. Для этого их нужно просто прописать в **pom.xml**:

```
<configuration>
    <container>
        <containerId>tomcat8x</containerId>
        <type>remote</type>
    </container>
    <configuration>
        <type>runtime</type>
        <properties>
            <cargo.remote.username>admin</cargo.remote.username>
            <cargo.remote.password>admin</cargo.remote.password>
            <cargo.tomcat.manager.url>http://localhost:8080/manager/text</cargo.tomcat.manager.url>
        </properties>
    </configuration>
</configuration>
```

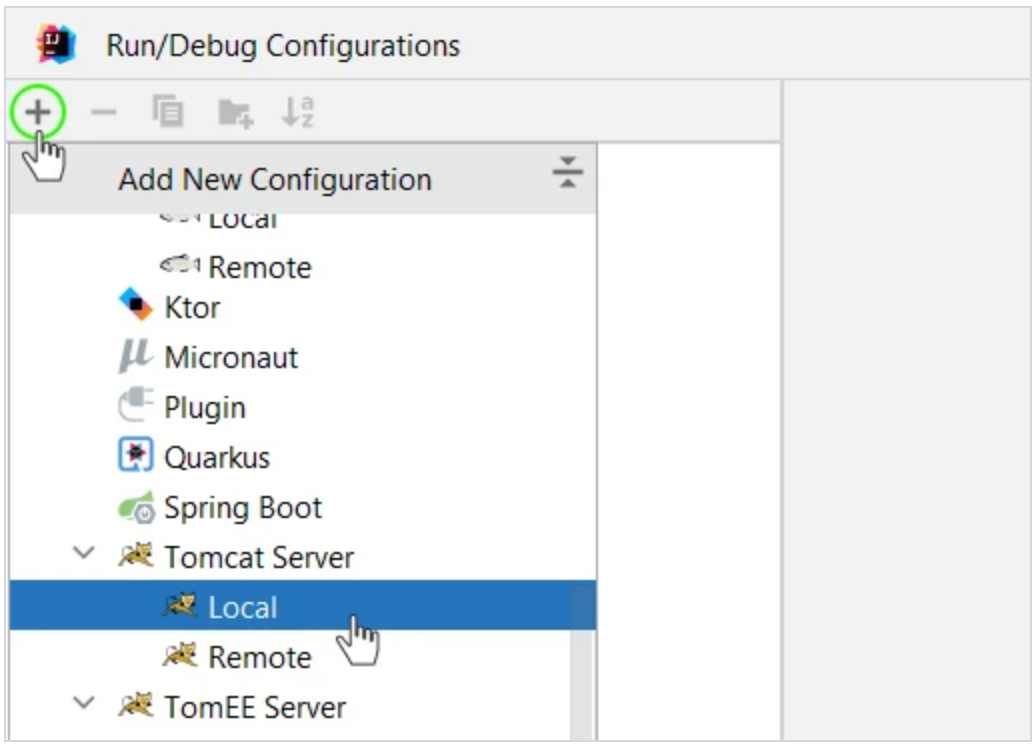
Деплой с помощью IntelliJ IDEA

Intellij IDEA делает всю работу сама, нужен только установленный Tomcat.

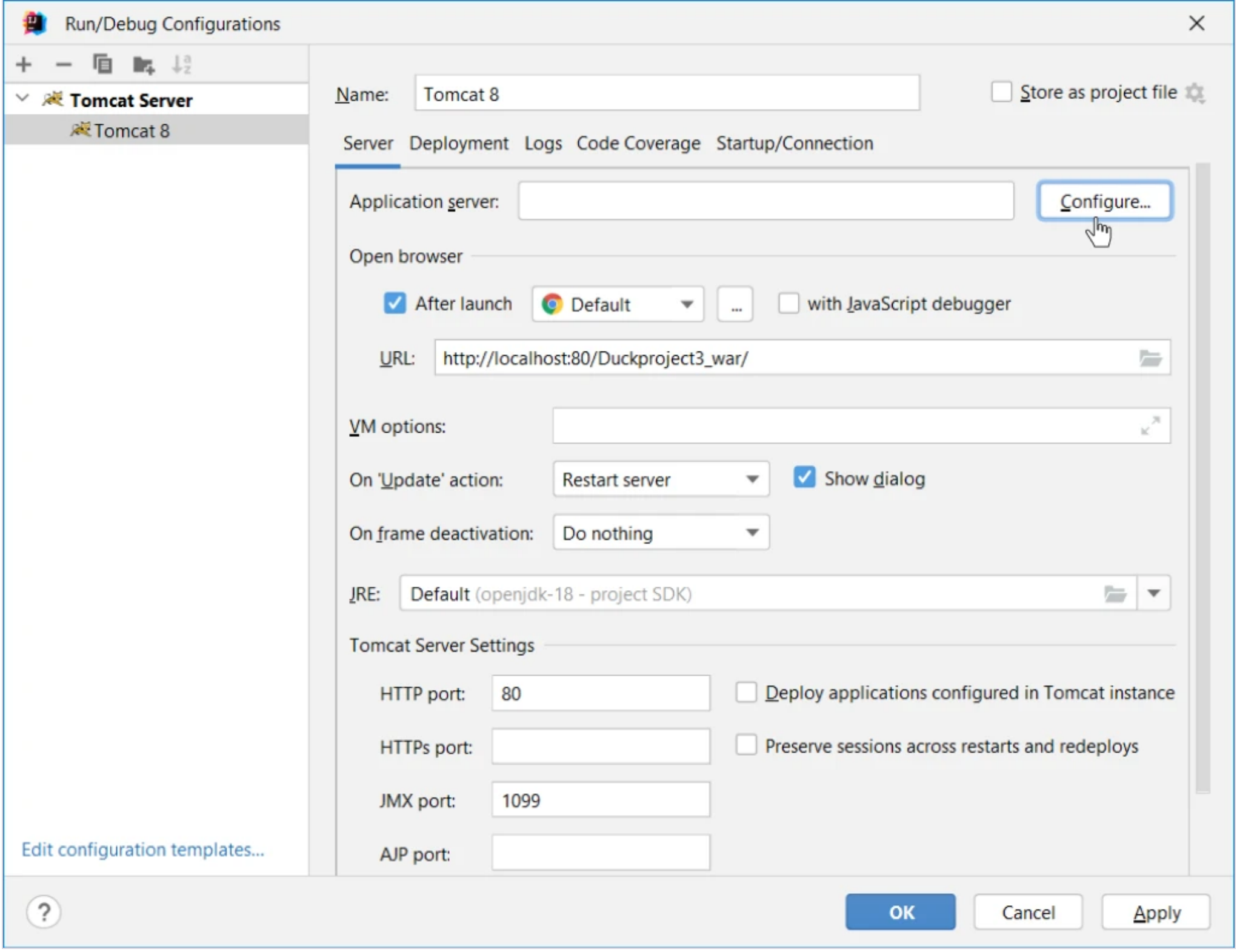
Шаг первый. Создаем локальную конфигурацию Tomcat:



Шаг второй. Затем выбираем локальный Tomcat:



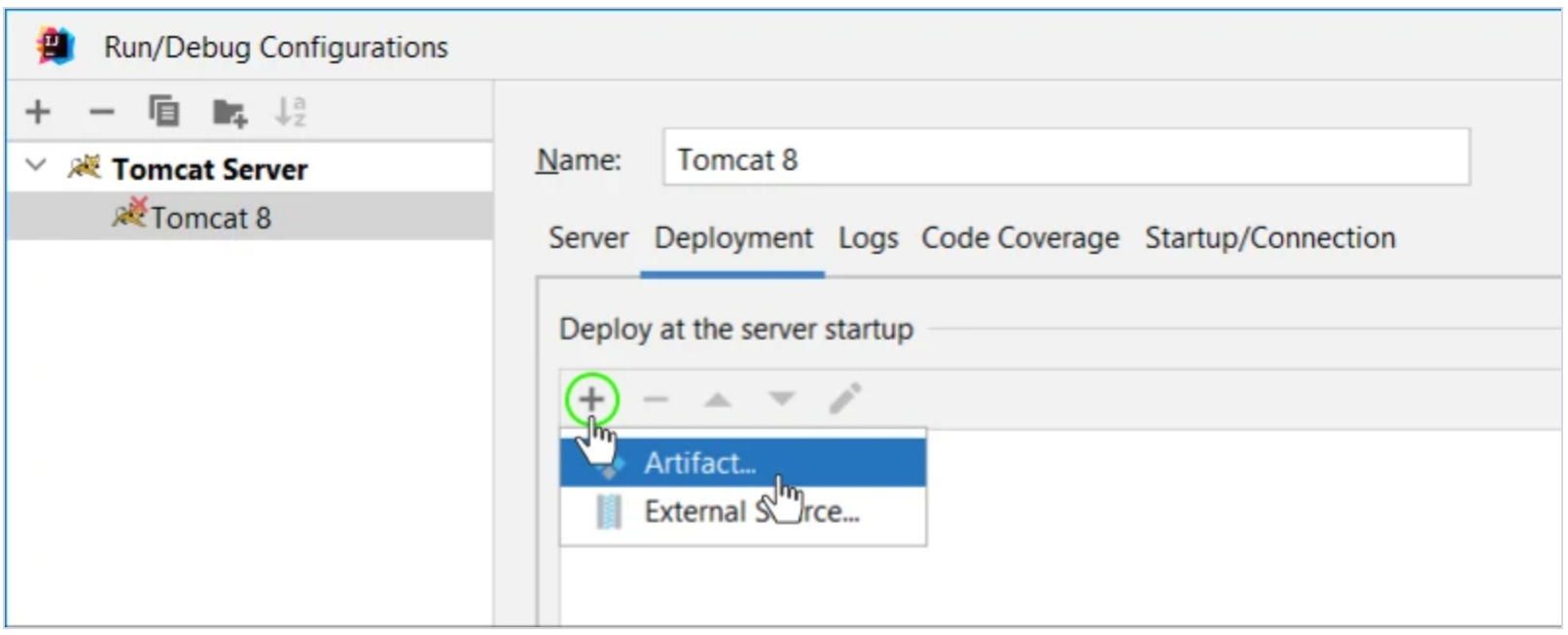
Шаг третий. Конфигурируем Tomcat:



Шаг четвертый. Добавляем путь к папке Tomcat.

Шаг пятый. Добавляем наш проект как артефакт к Tomcat.

Для этого переходим на вкладку Deployment и нажимает справа кнопку `+`.



Вот и все.

Кстати, если тебе нужно деплоить на удаленный сервер, просто выбери во втором шаге Remote Tomcat.

+16

Комментарии (6) популярные новые старые

JavaCoder

Введите текст комментария

Роман Уровень 51, Москва, Russian Federation 26 июля, 04:01

Тег packaging используется для того, чтобы результат был отправлен одним zip-файлом. Если его не казать, то будет один jar-файл, даже если у вас было несколько jar-файлов.

тут явно не казать должно быть написано,поправте пожалуйста

Ответить +1

Mentor-02 Backend Developer в **JavaRush** MENTOR 31 октября, 18:40

Поправили

Ответить 0

Artur May Software Developer в **UA-GIS** 15 июля, 13:18

Тут напевно потрібно закрити тег </configuration>

```
<configuration>
                    <url>http://localhost:8080/manager/text</url>
                    <server>TomcatServer</server>
                    <path>/simpleProject</path>
<configuration>
```

Ответить 0

Mentor-02 Backend Developer в **JavaRush** MENTOR 31 октября, 18:40

Виправили, дякуємо)

Ответить 0

Андрей Уровень 61, Красноярск, Россия 14 июля, 07:26

Добрый день,
1 - Шаг пятий - название пункта надо поправить
2 - в конце по ходу ошибка - "Вот и все.
Кстати, если тебе нужно деплоить на удаленный сервер, просто выбери в первом шаге Remote Tomcat."
-
remote мы вроде на втором шаге выбираем

Ответить 0

Mentor-02 Backend Developer в **JavaRush** MENTOR 31 октября, 18:40

Спасибо, поправили

Ответить 0

RUSH

JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

Русский

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ

ДОСТУПНО В Google Play

Загрузите в App Store