**Лабораторная работа №2**

**«Установка среды разработки. Внедрение зависимостей»**

**Цель работы:** познакомиться с корпоративными средствами создания и сборки приложений. Создать веб-приложение посредством Maven и импортировать его под IntelliJ IDEA.

**ПРАКТИЧЕСКАЯ ЧАСТЬ**

I. Настройка инструментариев и среды для разработки приложений на JAVA

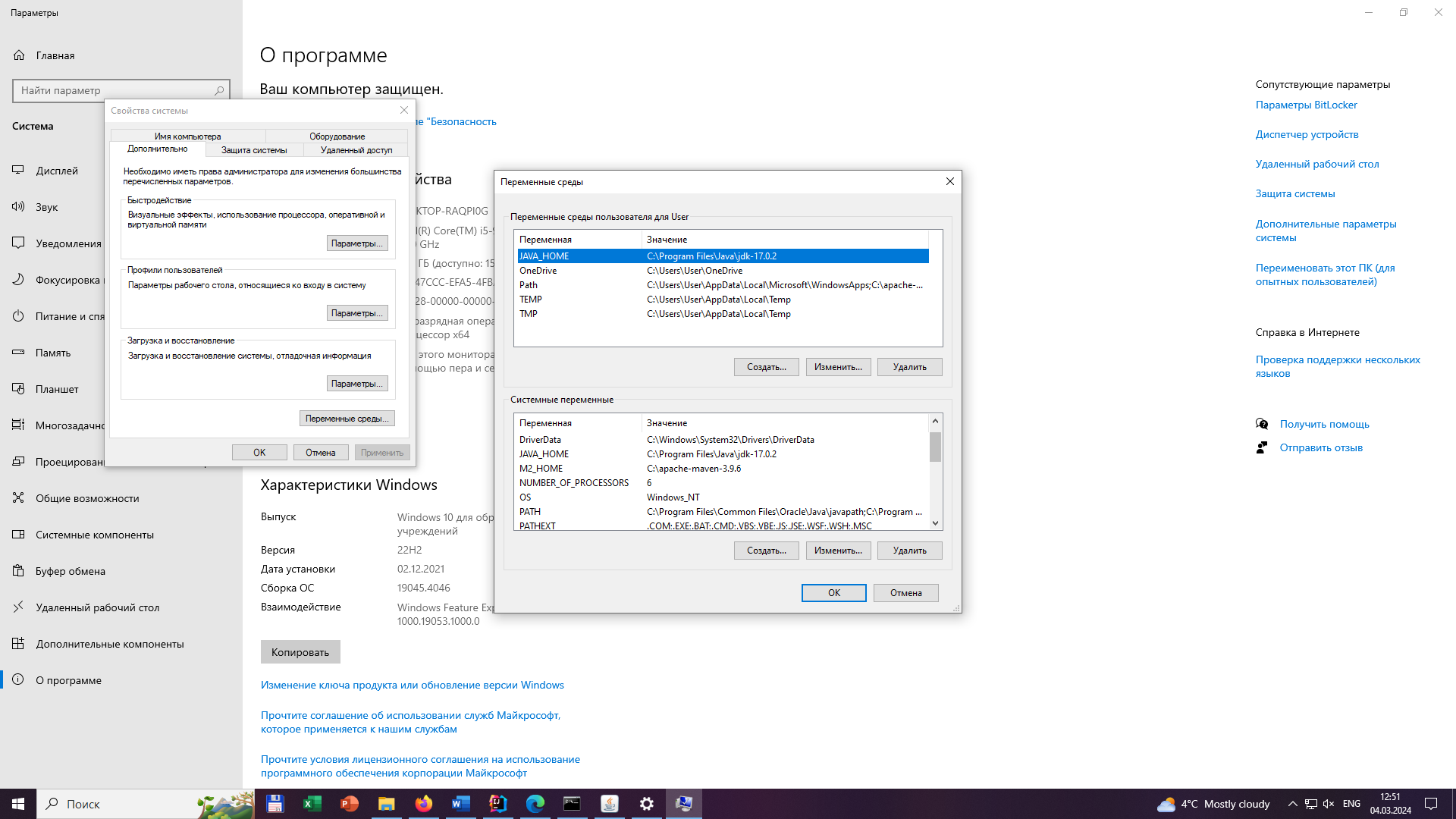


Рисунок 1 – Настройка среды

Проверить корректность установки, набрав в командной строке mvn –version.

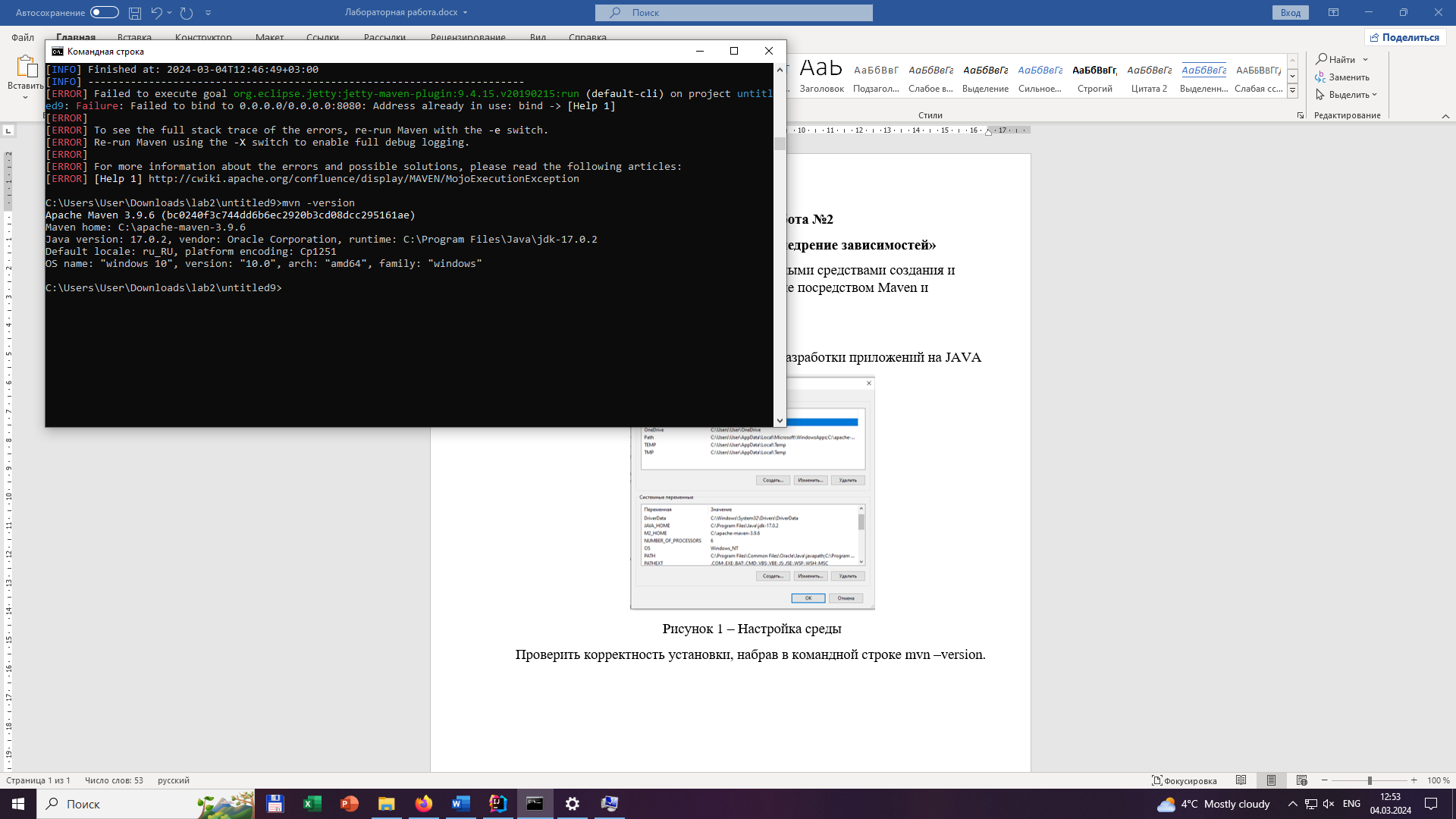


Рисунок 2 – Проверка работы Maven

II. Создание веб-приложения

1. Выполнить команду для генерации шаблона приложения и дальнейшей

разработки.

mvn archetype:generate -DgroupId={project-packaging} — DartifactId={project-

name} -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false

II. Создание веб-приложения



Необходимо собрать созданный проект и проверить его корректность

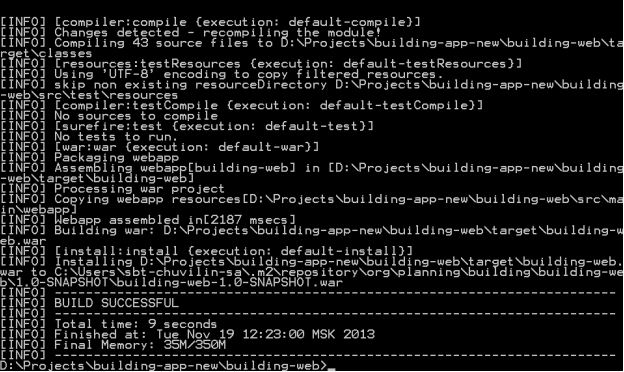
сборки. Из корня приложения запустить команду mvn clean install. Результат

должен быть таким, как показано на рисунке ниже, со статусом BUILD

SUCCESS. Это означает, что приложение правильно создано и успешно

компилируется в готовую сборку, которая является конечным этапом разработки

приложения.

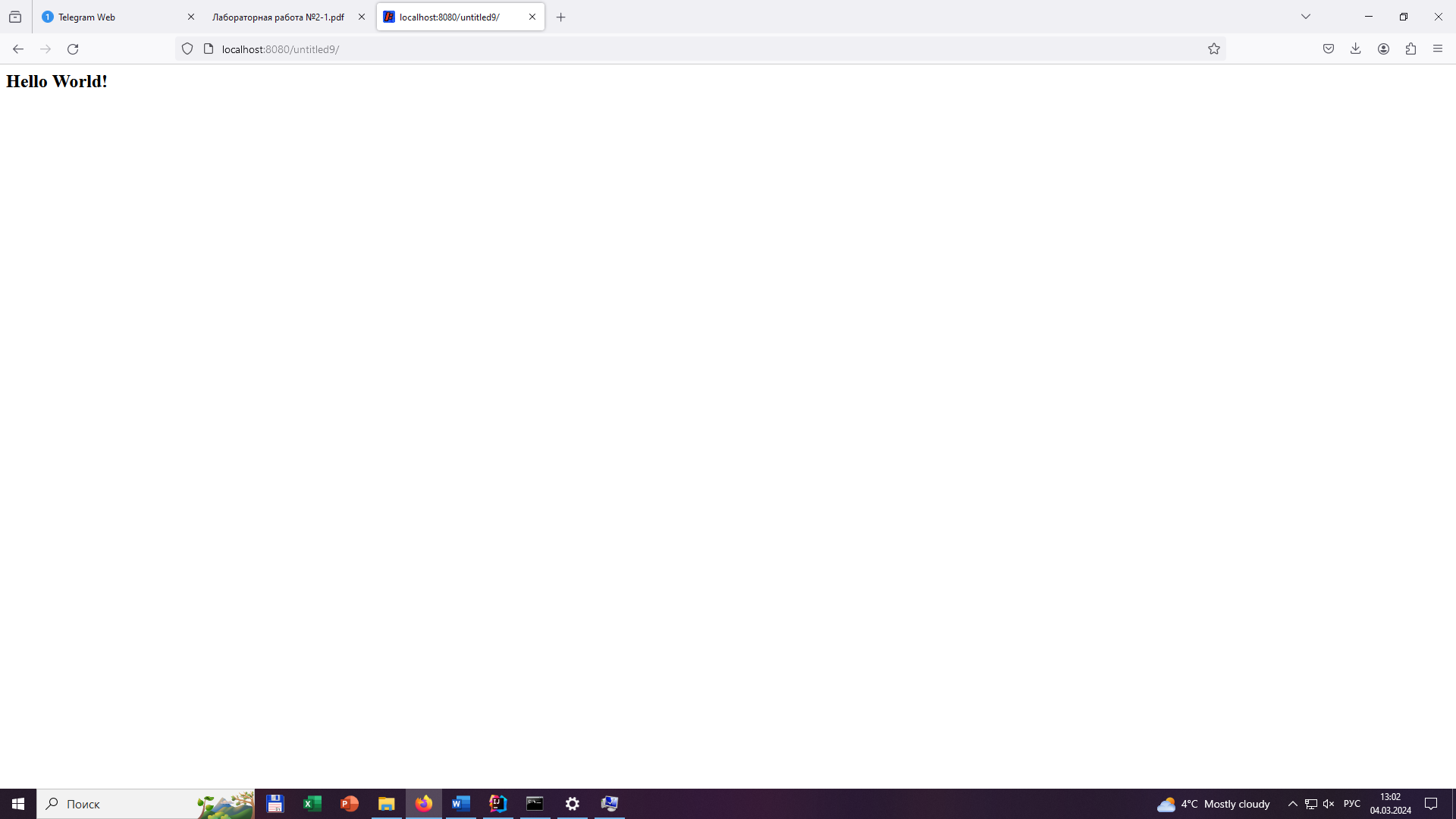


IV. Разработка и наполнение проекта

1. Код нашего сайта

<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>org.example</groupId>  
 <artifactId>untitled9</artifactId>  
 <packaging>war</packaging>  
 <version>1.0-SNAPSHOT</version>  
 <name>untitled9 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>untitled9</finalName>  
 <plugins>  
 <plugin>  
 <groupId>org.mortbay.jetty</groupId>  
 <artifactId>maven-jetty-plugin</artifactId>  
 <version>6.1.10</version>  
 <configuration>  
 <scanIntervalSeconds>10</scanIntervalSeconds>  
 <connectors>  
 <connector  
 implementation="org.mortbay.jetty.nio.SelectChannelConnector">  
 <port>8080</port>  
 <maxIdleTime>60000</maxIdleTime>  
 </connector>  
 </connectors>  
 <stopKey/>  
 <stopPort/>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

1. Результат



**Вопросы для самопроверки**

1. Чем отличаются зависимости от плагинов в Maven?

Зависимости от плагинов в Maven отличаются по своей функциональности. Некоторые плагины используются для компиляции кода, другие - для создания документации, а еще другие - для упаковки и развертывания приложения.

1. В чем особенность этапа жизненного цикла clean и чем он отличается от запуска плагина eclipse:clean?

Этап жизненного цикла clean в Maven предназначен для очистки проекта от сгенерированных ранее файлов и папок. Он удаляет все файлы, созданные во время предыдущих этапов сборки проекта. Плагин eclipse:clean, с другой стороны, предназначен для очистки проекта в среде разработки Eclipse

1. Разобрать состав проектного файла pom.xml (ссылка). Какие теги за что отвечают?

Файл pom.xml - это основной файл проекта в Maven. В нем содержится информация о проекте, его зависимостях, плагинах и настройках. В файле pom.xml можно найти теги, такие как <groupId>, <artifactId> и <version>, которые определяют уникальные идентификаторы проекта, а также <dependencies>, который определяет зависимости проекта

1. Что такое локальный и удаленный репозиторий (maven repository)?

Локальный репозиторий - это место на локальной машине, где Maven хранит загруженные зависимости. Удаленный репозиторий - это удаленное хранилище, где Maven ищет зависимости, которых нет в локальном репозитории. Maven загружает зависимости из удаленного репозитория и сохраняет их в локальном репозитории для последующего использования.

1. Какие есть особенность и параметры и плагина mvn archetype:generate?

Плагин mvn archetype:generate используется для создания нового проекта на основе шаблона (архетипа). Он позволяет выбрать тип проекта, настроить его параметры и создать базовую структуру проекта. Параметры плагина mvn archetype:generate могут включать groupId, artifactId, version и другие настройки, которые определяют особенности создаваемого проекта.

1. Разобрать в настройке дополнительных параметров плагина maven jetty. Как применяются дополнительные параметры и на что они влияют?

Дополнительные параметры в настройке плагина maven jetty позволяют настроить различные аспекты работы сервера Jetty. Например, параметр <scanIntervalSeconds> определяет интервал сканирования файлов для автоматической перезагрузки при изменениях, а параметр <webApp> указывает путь к директории с веб-приложением. Дополнительные параметры позволяют настроить поведение сервера Jetty в соответствии с требованиями проекта.