## Семинар №1

# Использование Maven и Gradle для разработки Java приложений

1. **Инструментарий:**

* Java Development Kit (JDK): [Скачать](https://www.oracle.com/java/technologies/javase-jdk14-downloads.html)
* Apache Maven: [Скачать](https://maven.apache.org/download.cgi)
* Gradle: [Скачать](https://gradle.org/releases/)
* IntelliJ IDEA: [Скачать](https://www.jetbrains.com/idea/download/)

1. **Цели семинара:**

* Получить практический опыт работы с Maven и Gradle
* Закрепить теоретический материал, полученный на лекции

По итогам семинара слушатель должен **знать** :

* Основы работы с Maven и Gradle
* Создание и настройка проекта с использованием Maven и Gradle
* Управление зависимостями с помощью Maven и Gradle

По итогам семинара слушатель должен **уметь** :

* Создавать и настраивать проекты с использованием Maven и Gradle
* Работать с зависимостями в проектах Maven и Gradle
* Создавать собственные задачи и плагины для Maven и Gradle

1. **План Содержание:**

|  |  |  |
| --- | --- | --- |
| **Этап урока** | **Тайминг, минуты** | **Формат** |
| Знакомство со студентами | 5 | Модерирует преподаватель |
| Вопросы по лекции, подготовка к семинару | 10 | Преподаватель зачитывает вопросы и разъясняет ответы |
| Блок 1: Создание и настройка проекта с использованием Maven | 30 | Практическое занятие |
| Блок 2: Создание и настройка проекта с использованием Gradle | 30 | Практическое занятие |
| Блок 3: Управление зависимостями с помощью Maven и Gradle | 30 | Практическое занятие |
| Обсуждение результатов, домашнее задание | 15 | Модерирует преподаватель |
| **Длительность:** | **120** |  |

### Блок 1. Создание и настройка проекта с использованием Maven

Тайминг:

Объяснение правил – 10 минут

Практическая работа – 20 минут

Задание:

Создать новый Maven проект и настроить его с использованием файла pom.xml.

Пример решения:

1. Создайте новый Maven проект с помощью команды:

mvn archetype:generate -DgroupId=com.example -DartifactId myapp -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

1. Откройте файл pom.xml в вашем проекте и добавьте следующие зависимости:

<dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.1</version>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>1.7.30</version>  
 </dependency>  
</dependencies>

1. Создайте пакет com.example.myapp и в нем простой класс HelloWorld с методом main:

package com.example.myapp;  
  
public class HelloWorld {  
 public static void main(String[] args) {  
 System.out.println("Hello, World!");  
 }  
}

Часто встречающиеся ошибки:

* Ошибки в структуре проекта
* Некорректное указание зависимостей в файле pom.xml
* Ошибки в импорте пакетов

### Блок 2. Создание и настройка проекта с использованием Gradle

Тайминг:

Объяснение правил – 10 минут

Практическая работа – 20 минут

Задание:

Создать новый Gradle проект и настроить его с использованием файла build.gradle.

Пример решения:

1. Создайте новый Gradle проект с помощью команды:

gradle init --type java-application

1. Откройте файл build.gradle в вашем проекте и добавьте следующие зависимости:

dependencies {  
 implementation 'org.slf4j:slf4j-api:1.7.30'  
 testImplementation 'junit:junit:4.13.1'  
}

1. В пакете src/main/java создайте простой класс HelloWorld с методом main:

package com.example.myapp;  
  
public class HelloWorld {  
 public static void main(String[] args) {  
 System.out.println("Hello, World!");  
 }  
}

Часто встречающиеся ошибки:

* Ошибки в структуре проекта
* Некорректное указание зависимостей в файле build.gradle
* Ошибки в импорте пакетов

### Блок 3. Управление зависимостями с помощью Maven и Gradle

Тайминг:

Объяснение правил – 10 минут

Практическая работа – 20 минут

Задание:

Добавить в проекты, созданные в предыдущих блоках, новую зависимость и использовать ее.

Пример решения:

1. Для Maven проекта добавьте следующую зависимость в pom.xml:

<dependency>  
 <groupId>com.google.guava</groupId>  
 <artifactId>guava</artifactId>  
 <version>30.1-jre</version>  
</dependency>

1. Для Gradle проекта добавьте следующую зависимость в build.gradle:

dependencies {  
 implementation 'com.google.guava:guava:30.1-jre'  
}

1. Используйте Guava в вашем классе HelloWorld:

package com.example.myapp;  
  
import com.google.common.base.Joiner;  
  
public class HelloWorld {  
 public static void main(String[] args) {  
 String[] words = {"Hello", "World"};  
 String message = Joiner.on(", ").join(words);  
 System.out.println(message);  
 }  
}

Часто встречающиеся ошибки:

* Некорректное указание зависимостей в файлах pom.xml или build.gradle
* Ошибки в импорте пакетов
* Использование неправильной версии библиотеки

**Домашнее задание**

Условие:

Создать проект с использованием Maven или Gradle, добавить в него несколько зависимостей и написать код, использующий эти зависимости.

Пример решения:

1. Создайте новый Maven или Gradle проект, следуя инструкциям из блока 1 или блока 2.
2. Добавьте зависимости org.apache.commons:commons-lang3:3.12.0 и com.google.code.gson:gson:2.8.6.
3. Создайте класс Person с полями firstName, lastName и age.
4. Используйте библиотеку commons-lang3 для генерации методов toString, equals и hashCode.
5. Используйте библиотеку gson для сериализации и десериализации объектов класса Person в формат JSON.

Рекомендации для преподавателей по оценке задания:

* Проверьте правильность структуры проекта
* Убедитесь в корректности указания зависимостей в файлах pom.xml или build.gradle
* Оцените правильность использования библиотек в коде
* Проверьте работоспособность кода