**Лабораторная работа № 3**

**УСТАНОВКА TOMCAT. НАСТРОЙКА СБОРЩИКА MAVEN, СОЗДАНИЕ И ЗАПУСК ВЕБ-ПРИЛОЖЕНИЯ С ПОМОЩЬЮ MAVEN**

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

**Программное обеспечение:**

1.IntelliJ IDEA

2.Maven 3

3.JDK 1.8

**Необходимая теоретическая подготовка:**

• Представление о разработке на языке Java;

• Умение скачать и произвести установку JDK;

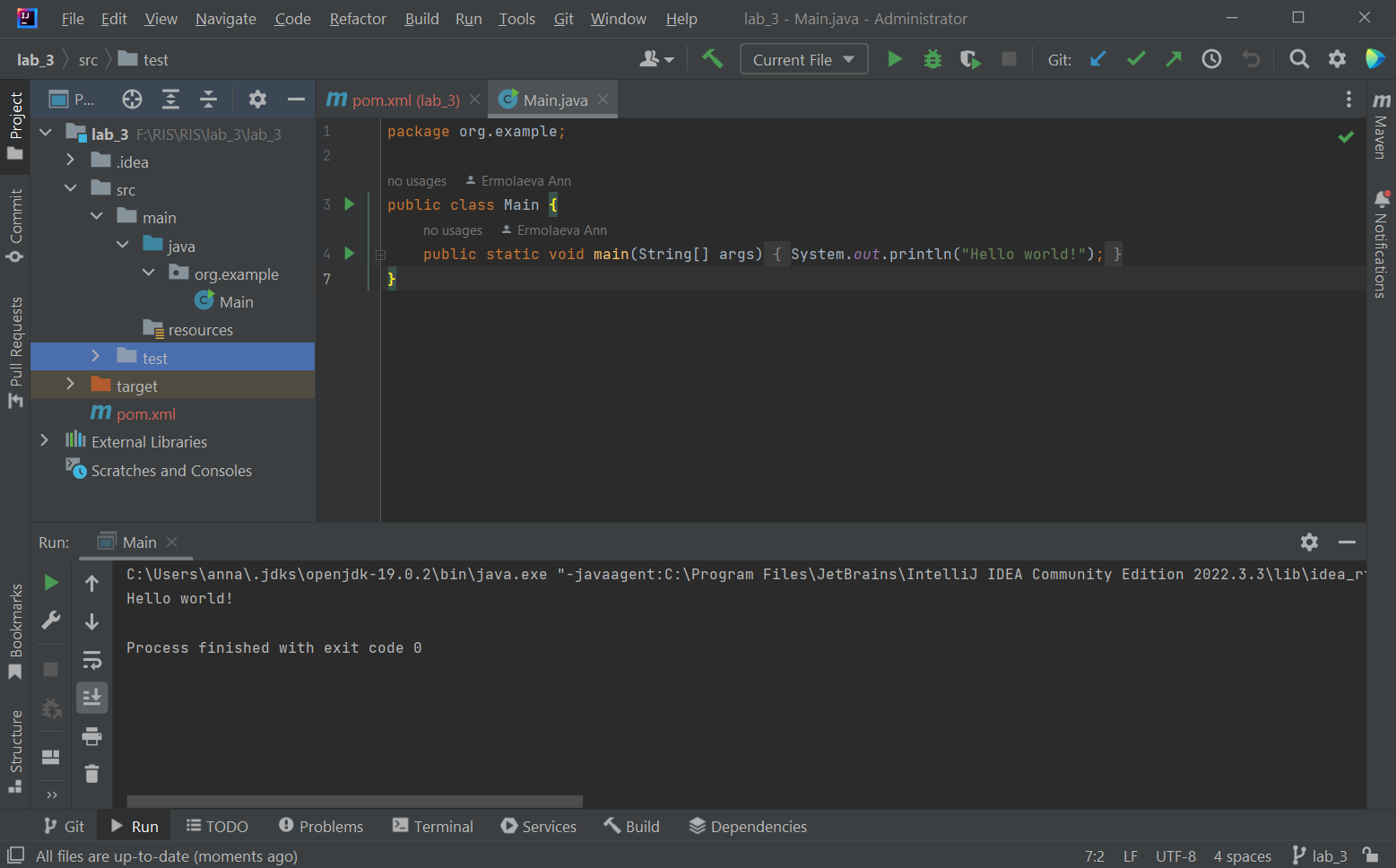
• Умение скачать и распаковать IntelliJ IDEA;

• Умение скачать и распаковать maven.

**Задание:** Настройка инструментариев и среды для разработки приложений на JAVA.

**Методика выполнения работы**

1. Создание нового проекта в папке репозитория «RIS».

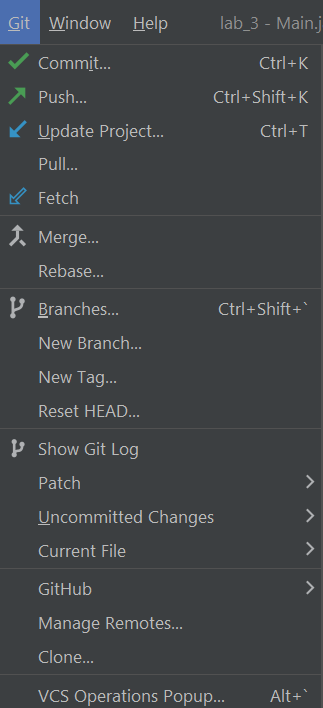


**Рисунок 1** – Результат программы, выводящей «Hello world!»

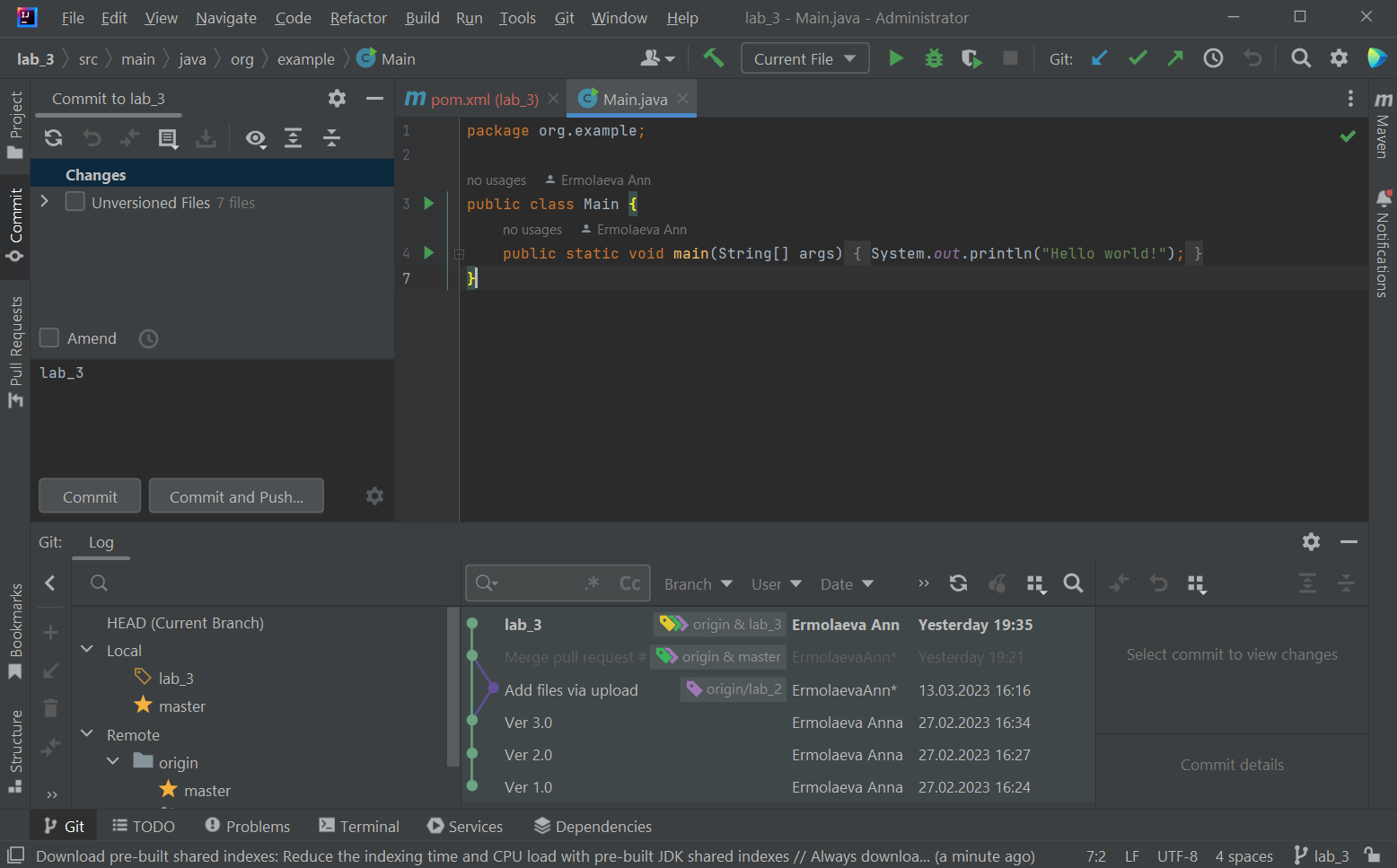
2. В разделе Git нажимаем New Branch, чтобы создать новую ветку lab\_3 в данном репозитории.

3. Нажимаем Commit в этом же разделе Git.

4. Далее выбираем Push.



**Рисунок 2** – Раздел Git



**Рисунок 3** – Результат выполненной лабораторной работы

**Контрольные вопросы:**

1. Чем отличаются зависимости от плагинов в Maven?

Зависимости предоставляют необходимый функционал для вашего приложения, а плагины позволяют настраивать процесс сборки и выполнять дополнительные задачи в процессе сборки.

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

Этап жизненного цикла clean в Maven отвечает за очистку проекта перед сборкой. В процессе выполнения этого этапа, Maven удаляет все файлы, созданные в результате предыдущих сборок проекта, включая скомпилированные классы, сгенерированные исходные коды, ресурсы и т.д.

Запуск плагина eclipse:clean выполняет очистку проекта в среде разработки Eclipse. Он удаляет все файлы, созданные Eclipse в процессе разработки, включая скомпилированные классы, временные файлы, файлы настроек и т.д.

Основное отличие между этими двумя очистками заключается в том, что этап clean в Maven удаляет все файлы, созданные в результате предыдущих сборок проекта, в то время как плагин eclipse:clean удаляет только файлы, созданные средой разработки Eclipse. Таким образом, если вы работаете в среде разработки Eclipse, то необходимо запускать плагин eclipse:clean, чтобы очистить все временные файлы, созданные Eclipse. Если же вы работаете в другой среде разработки или через командную строку, то необходимо использовать этап clean в Maven для очистки проекта перед следующей сборкой.

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

Тег project является базовым и содержит всю информацию о проекте. В заголовке указана информация, необходимая Maven для понимания файла pom.xml. Тег modelVersion указывает на текущую версию POM. Эти два тега обычно генерируются автоматически, менять их не нужно.

Затем идёт информация, формирующая уникальный идентификатор проекта, теги groupId и artifactId. Её задают при генерации из архетипа. Тег version тоже входит в эту группу. Он обычно генерируется и обновляется автоматически. После номера версии идёт суффикс -SNAPSHOT. Это означает, что проект находится в стадии разработки. В процессе выпуска ПО фреймворк уберёт этот суффикс, а если разработка продолжится – автоматически увеличит номер версии. Вместе эти три тега позволяют однозначно идентифицировать артефакт.

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

Локальный репозиторий Maven – это хранилище библиотек и зависимостей, которые загружаются локально на компьютер разработчика при первой загрузке. Он находится на локальном диске и используется для хранения всех зависимостей, которые Maven загружает из центрального репозитория или других удаленных репозиториев в процессе сборки проекта.

Удаленный репозиторий Maven – это хранилище библиотек и зависимостей, которые расположены на удаленном сервере. Когда Maven не находит необходимые зависимости в локальном репозитории, он ищет их в удаленном репозитории.

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

Плагин mvn archetype:generate в Maven используется для генерации проекта на основе шаблона (архетипа).

Особенности плагина mvn archetype:generate включают:

* Генерация проекта на основе заданного шаблона (архетипа).
* Создание структуры каталогов для проекта в соответствии с выбранным шаблоном.
* Подстановка стандартных настроек, файлов и зависимостей в созданный проект.

Некоторые из параметров плагина mvn archetype:generate включают:

* DgroupId - идентификатор группы проекта, например, com.example.
* DartifactId - идентификатор проекта, например, my-project.
* DarchetypeGroupId - идентификатор группы архетипов, например, org.apache.maven.archetypes.
* DarchetypeArtifactId - идентификатор архетипа, например, maven-archetype-quickstart.
* DarchetypeVersion - версия архетипа, например, 1.4.
* DinteractiveMode - включение интерактивного режима для выбора настроек проекта.
* Dpackage - базовый пакет проекта, например, com.example.myproject.

Параметры плагина могут варьироваться в зависимости от выбранного шаблона.

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

Плагин Jetty используют в Maven для быстрой разработки и тестирования. По желанию он может периодически сканировать проект на предмет изменений и автоматически развертывать веб-контейнеры, если они обнаружены. Это делает цикл разработки более продуктивным, устраняя этапы сборки и развертывания: программист использует IDE для внесения изменений в проект, а запущенный веб-контейнер автоматически подхватывает их, позволяя разработчику сразу же их протестировать.

**Вывод:** В данной лабораторной работе мы познакомились с корпоративными средствами создания и сборки приложений. Создали веб-приложение посредством Maven и импортировали его под IntelliJ IDEA