



# Maven i Git



# Apache Maven

**Apache Maven** to narzędzie służące do automatyzacji budowy oprogramowania.

Maven pomaga w zarządzaniu całym cyklem życia projektu, od kompilacji, przez testowanie, pakowanie, aż po wdrożenie i generowanie dokumentacji.



# Apache Maven

W IntelliJ IDEA Maven jest wbudowany, więc w prosty sposób możemy utworzyć projekt mavenowy.

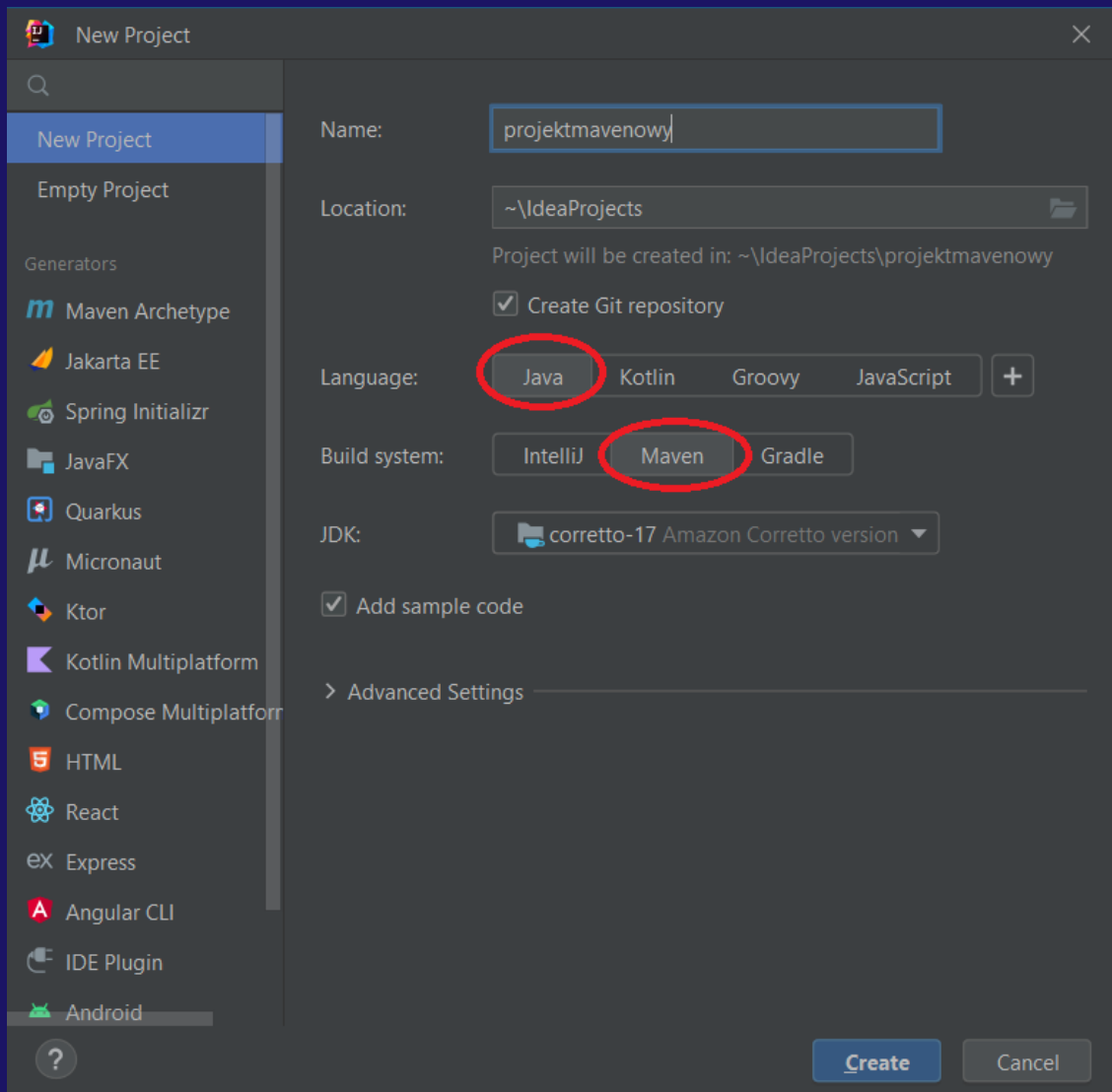
Uruchamiamy IntelliJ IDEA i wybieramy opcję Create New Project na ekranie startowym lub File > New > Project...



# Apache Maven

W IntelliJ IDEA Maven jest wbudowany, więc w prosty sposób możemy utworzyć projekt mavenowy.

Uruchamiamy IntelliJ IDEA i wybieramy opcję Create New Project na ekranie startowym lub File > New > Project...



FileEditViewNavigateCodeRefactorBuildRunToolsGitWindowHelp

projektmavenowy - pom.xml (projektmavenowy)

projektmavenowy m pom.xml

Project

projektmavenowy C:\Users\luke\IdeaProjects\projektmavenowy

.idea

src

main

java

resources

test

java

.gitignore

m pom.xml

External Libraries

Scratches and Consoles

Commit

Bookmarks

Structure

m pom.xml (projektmavenowy)

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5     <modelVersion>4.0.0</modelVersion>
6
7     <groupId>org.example</groupId>
8     <artifactId>projektmavenowy</artifactId>
9     <version>1.0-SNAPSHOT</version>
10
11     <properties>
12         <maven.compiler.source>17</maven.compiler.source>
13         <maven.compiler.target>17</maven.compiler.target>
14         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15     </properties>
16
17 </project>
```

project

Git

TODO

Problems

Terminal

Services

Profiler

Dependencies

17:11

LF

UTF-8

4 spaces

master

# **pom.xml**

Plik **pom.xml** jest niezbędnym elementem projektu.  
Jest typu xml  
(eXtensible Markup Language)

Zawiera konfigurację projektu oraz informacje o jego zależnościach, pluginach, wersjach i innych właściwościach.

```
<?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>org.example</groupId>
  <artifactId>projektmavenowy</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

</project>
```

## pom.xml

```
xmlns="http://maven.apache.org/POM/4.0.0"
```

definiuje domyślną przestrzeń nazw XML dla dokumentu. Przestrzeń nazw jest mechanizmem XML, który dostarcza unikalny kontekst dla elementów i atrybutów.

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

deklaruje przestrzeń nazw XML Schema Instance. XML Schema jest językiem służącym do definiowania struktury i reguł walidacji dla dokumentów XML.

```
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"
```

Wskazuje lokalizację schematu XML Schema, który powinien być użyty do walidacji dokumentu. Podaje parserowi XML dokładną ścieżkę do pliku XSD (XML Schema Definition), który zawiera definicje typów danych, elementów i atrybutów dozwolonych w dokumencie, oraz reguły, które muszą być spełnione.

## pom.xml

```
<groupId>org.example</groupId>  
<artifactId>projektmavenowy</artifactId>  
<version>1.0-SNAPSHOT</version>
```

Musimy też podać wersję modelu POM (Project Object Model):

```
<modelVersion>4.0.0</modelVersion>
```

Następnie mamy standardowe właściwości:

```
<groupId>org.example</groupId>
```

określa identyfikator grupy, do której należy projekt.

Zazwyczaj jest to odwrócona nazwa domeny organizacji/developera.

```
<artifactId>projektmavenowy</artifactId>
```

Jest unikalną nazwą artefaktu generowanego przez projekt. Jest to nazwa, pod którą artefakt jest znany i przechowywany w repozytorium.

```
<version>1.0-SNAPSHOT</version>
```

Określa wersję projektu (SNAPSHOT – jest wersją rozwojową).



## pom.xml

```
<groupId>org.example</groupId>  
<artifactId>projektmavenowy</artifactId>  
<version>1.0-SNAPSHOT</version>
```

Te 3 elementy tworzą unikalny identyfikator artefaktu w systemie Mavena, który umożliwia zarządzanie zależnościami i ich wersjonowanie

## pom.xml

```
<properties>  
  <maven.compiler.source>17</maven.compiler.source>  
  <maven.compiler.target>17</maven.compiler.target>  
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
</properties>
```

Ustawienia te kolejno opisują, jaką wersję javy używamy, z jaką wersją javy mają być kompatybilne skompilowane klasy, oraz jakie kodowanie ma być użyte dla plików źródłowych.

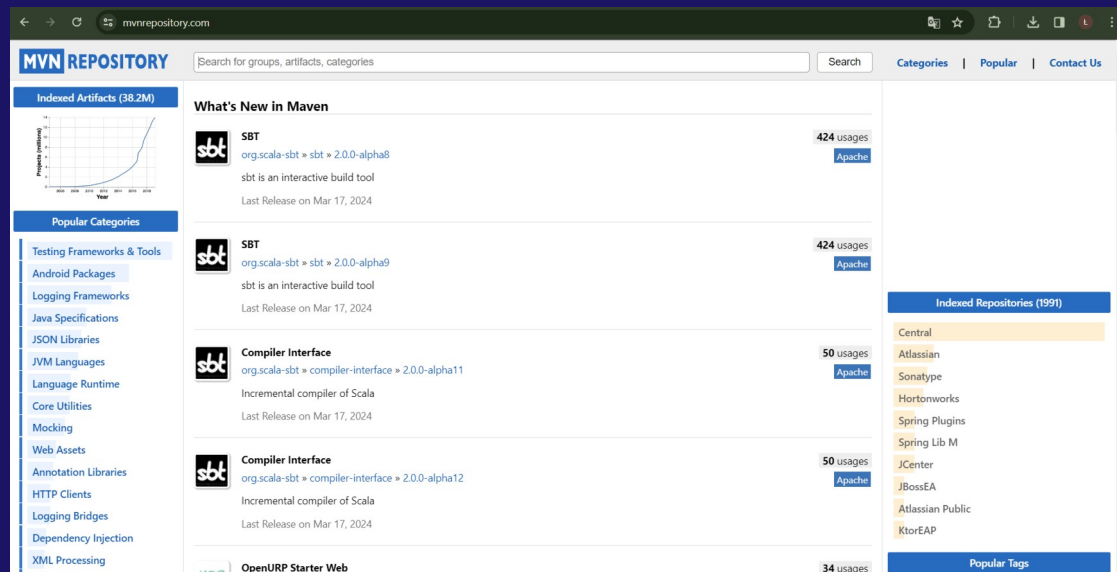
# Zarządzanie zależnościami

Jest to kluczowa funkcjonalność  
Mavena ułatwia dodawanie,  
aktualizowanie i usuwanie  
zależności – automatyzując to  
zadanie .



# Zarządzanie zależnościami

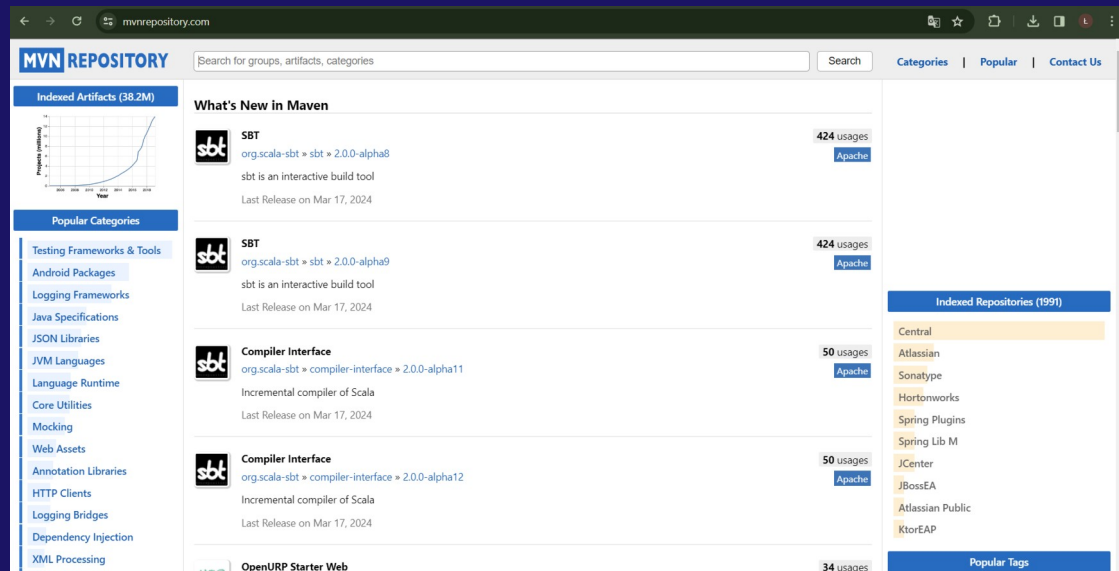
Maven korzysta z centralnych repozytoriów (takich jak Maven Central), gdzie przechowywane są artefakty (biblioteki, frameworki itp.), które mogą być używane jako zależności w projektach.



<https://mvnrepository.com/>

# Zarządzanie zależnościami

Maven automatycznie rozwiązuje zależności co oznacza, że gdy dodasz do projektu pewną zależność, Maven sam znajduje i pobiera wszystkie dodatkowe biblioteki, których ta zależność może wymagać do działania. Dzięki temu nie musisz ręcznie zarządzać każdą zależnością i jej zależnościami.



<https://mvnrepository.com/>

# Zarządzanie zależnościami

Przykład: chcemy umożliwić aplikacji z ćwiczeń logowanie, dane użytkownika przechowujemy razem z hasłem podobnie jak informacje o pojazdach – w pliku.

Hasła nie mogą być przechowywane w pliku jako „plain-text”. Osoby nieuprawnione mogą mieć dostęp do haseł. W tym również my.

Hasła należy zahashować. A następnie porównywać zahashowane hasło podawane przy logowaniu z hashem z pliku.

# Zarządzanie zależnościami

Zamiast samemu pisać metodę hashującą wykorzystamy rozwiązanie z repozytorium mavena:

<https://mvnrepository.com/artifact/commons-codec/commons-codec/1.16.1>

Przykład użycia tego rozwiązania:

<https://www.baeldung.com/sha-256-hashing-java>

# Zarządzanie zależnościami

Dodajemy zależność do pliku pom.xml, maven pobierze i doda ją za nas:

```
<dependencies>

  <!-- https://mvnrepository.com/artifact/commons-codec/commons-codec -->
  <dependency>
    <groupId>commons-codec</groupId>
    <artifactId>commons-codec</artifactId>
    <version>1.16.0</version>
  </dependency>

</dependencies>
```



# Zarządzanie zależnościami

Teraz możemy użyć metodę służącą do hashowania importując odpowiednią klasę:

```
package org.example;

import org.apache.commons.codec.digest.DigestUtils;

public class Hasher {
    public static String hashPassword(String password){
        return DigestUtils.sha256Hex(password);
    }
}
```

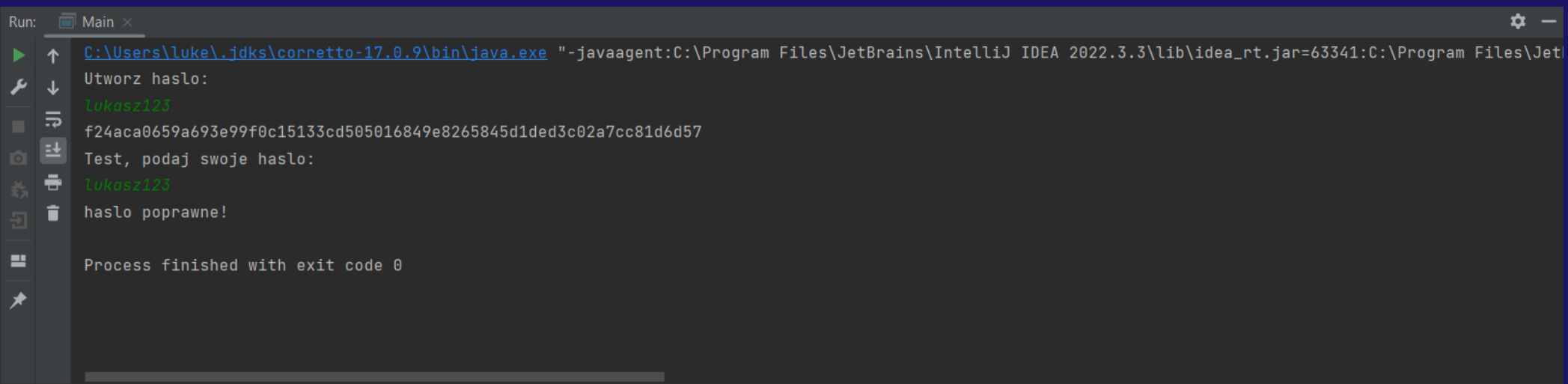
# Zarządzanie zależnościami

Przetestujmy działanie metody pobierając hasło od usera, hashując je, zapisując do pliku i porównując z hasłem podanym drugi raz (try,catch zostały pominięte):

```
public class Main {  
    public static void main(String[] args) throws FileNotFoundException {  
        Scanner sc = new Scanner(System.in);  
        System.out.println("Utworz haslo:");  
        String hashed = Hasher.hashPassword(sc.nextLine());  
        System.out.println(hashed);  
        PrintWriter pw = new PrintWriter("./pass.txt");  
        pw.println(hashed);  
        pw.close();  
        Scanner scfile = new Scanner(new File("./pass.txt"));  
        System.out.println("Test, podaj swoje haslo:");  
        if (Hasher.hashPassword(sc.nextLine()).equals(scfile.nextLine())){  
            System.out.println("haslo poprawne!");  
        }  
        else{  
            System.out.println("bledne haslo!");  
        }  
    }  
}
```

# Zarządzanie zależnościami

Uruchomienie aplikacji:



```
Run: Main x
C:\Users\luke\.jdk\corretto-17.0.9\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.3.3\lib\idea_rt.jar=63341:C:\Program Files\Jet
Utworz haslo:
lukasz123
f24aca0659a693e99f0c15133cd505016849e8265845d1ded3c02a7cc81d6d57
Test, podaj swoje haslo:
lukasz123
haslo poprawne!

Process finished with exit code 0
```

# Cykl życia budowy projektu

Cykl życia definiuje proces budowy i dystrybucji projektu. Składa się z uporządkowanych w kolejności faz, które należy wykonać.

# Cykl życia budowy projektu

Istnieją 3 podstawowe cykle życia:

**clean** - czyści po sobie poprzednie efekty budowania aplikacji (często oznacza to wyczyszczenie folderu target)

**site** - buduje dokumentację projektu (np. javadoc)

domyślny cykl, który z kolei składa się z następujących faz (w prawidłowej kolejności):

**validate**: Sprawdza, czy projekt jest poprawny i czy wszystkie niezbędne informacje są dostępne.

**compile**: Kompiluje źródła projektu.

**test**: Testuje skompilowany kod źródłowy przy użyciu odpowiedniego frameworka testowego.

**package**: Pakuje skompilowany kod np. JAR w katalogu target.

**verify**: Sprawdza, czy pakiet jest poprawny i spełnia kryteria jakości.

**install**: Instaluje pakiet do lokalnego repozytorium, co umożliwia jego użycie jako zależności w innych projektach na tej samej maszynie.

**deploy**: Kopiuje ostateczny pakiet do zdalnego repozytorium, aby był udostępniony dla innych deweloperów i projektów.

# Zbudowanie artefaktu i Instalacja

Okazuje się, że nasz artefakt przy próbie uruchomienia nie będzie działał. Z 2 powodów:

- brak pliku manifestu
- brak dołączonych zależności w paczce.



# Zbudowanie artefaktu i Instalacja

Pierwszy jak i drugi problem można rozwiązać za pomocą odpowiedniej wtyczki mavena użytej przy budowaniu projektu.



# Zbudowanie artefaktu i Instalacja

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-shade-plugin</artifactId>
      <version>3.5.2</version>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
            <goal>shade</goal>
          </goals>
          <configuration>
            <createDependencyReducedPom>false</createDependencyReducedPom>
            <transformers>
              <transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
                <mainClass>org.example.Main</mainClass>
              </transformer>
            </transformers>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```



# Zbudowanie artefaktu i Instalacja

Po instalacji artefaktu, możemy na lokalnej maszynie podpinać tak utworzoną paczkę do innych projektów w taki sam sposób jak z repozytorium zdalnego:

```
<dependency>  
  <groupId>org.example</groupId>  
  <artifactId>projektmavenowy</artifactId>  
  <version>1.0-SNAPSHOT</version>  
</dependency>
```



# Struktura katalogów

W projektach mavenowych obowiązuje ustalona struktura katalogów.

Kod źródłowy umieszczamy w :

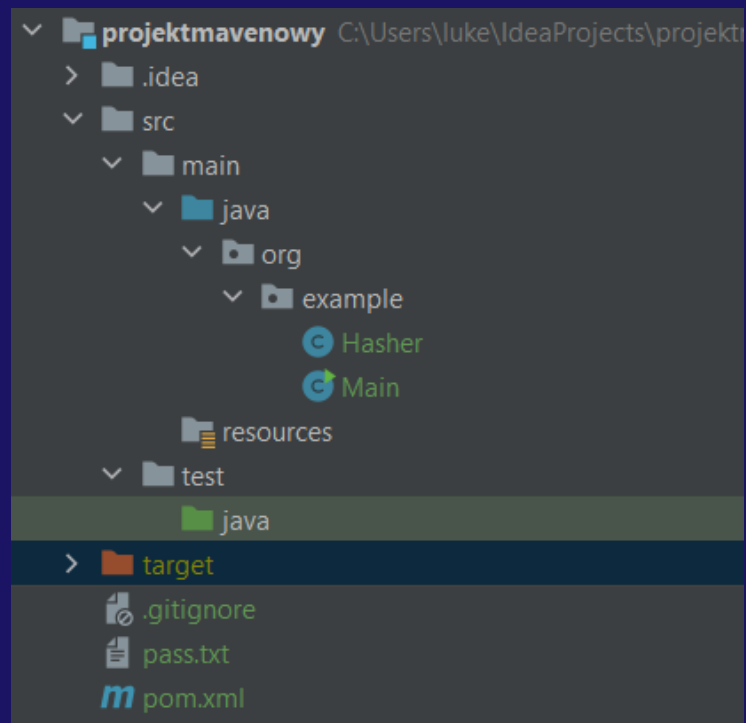
src>main>java>paczki

Zasoby statyczne jak pliki konfiguracyjne, skrypty w:

src>main>resources

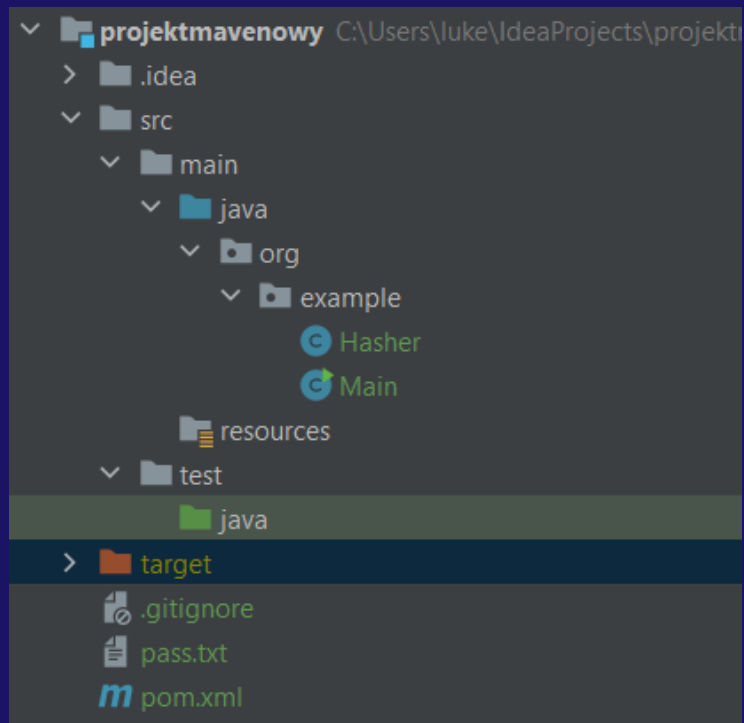
Testy w:

src>test>java



# Git i Github

W trakcie dodawania projektu  
zaznaczyłem, że chcę  
zainicjować również  
repozytorium git.  
Stąd kolorowe nazwy plików.  
Repozytorium git możemy  
również zainicjować ręcznie.



# Git i Github

Git to system kontroli wersji, umożliwia śledzenie zmian w kodzie źródłowym podczas tworzenia oprogramowania. Dzięki Gitowi możemy tworzyć wiele wersji projektu oraz łatwo wracać do poprzednich stanów i zarządzać zmianami w kodzie.



# Git i Github

Jeżeli nie mamy gita na komputerze należy go zainstalować.

Instalację można pobrać ze strony:

**<https://git-scm.com/downloads>**



# Git i Github

---

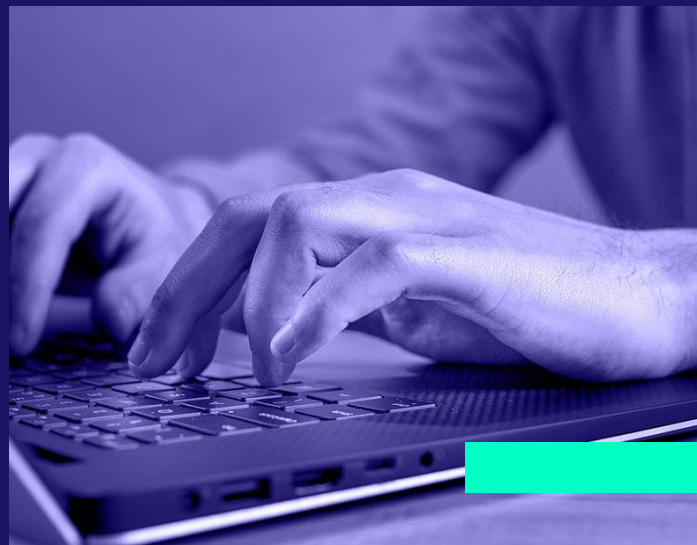
**GitHub** jest jednym z najpopularniejszych serwisów internetowych hostujących repozytoria Git w chmurze.

**GitHub** umożliwia programistom i zespołom programistycznym zarządzanie projektami programistycznymi, współpracę przy kodzie, a także śledzenie i zgłaszanie błędów.



# Repozytorium

**Repozytorium**, to zbiór plików projektu i dodatkowych informacji zarządzanych przez system kontroli wersji Git. Repozytorium przechowuje kompletną historię zmian dokonanych w plikach projektu oraz informacje o strukturze projektu w danym momencie. Każda zmiana zapisana w repozytorium Git zawiera autora zmian, datę oraz wiadomość opisującą zmianę



# Repozytorium

## **Lokalne repozytorium:**

repozytorium zlokalizowane na Twoim lokalnym komputerze, gdzie dokonujesz zmian w kodzie i zarządzasz historią projektu.

## **Zdalne repozytorium:**

Repozytorium znajdujące się na serwerze w Internecie (np. na GitHub, GitLab, Bitbucket)





# GitHub

Aby założyć konto należy  
odwiedzić stronę:

**<https://github.com/>**



# Github

Aby założyć konto należy odwiedzić stronę:

**<https://github.com/>**

Welcome to GitHub!

Let's begin the adventure

Enter your email\*

✓ studentumcs@email.com

Create a password\*

✓ .....

Enter a username\*

→ studentumcs

Continue

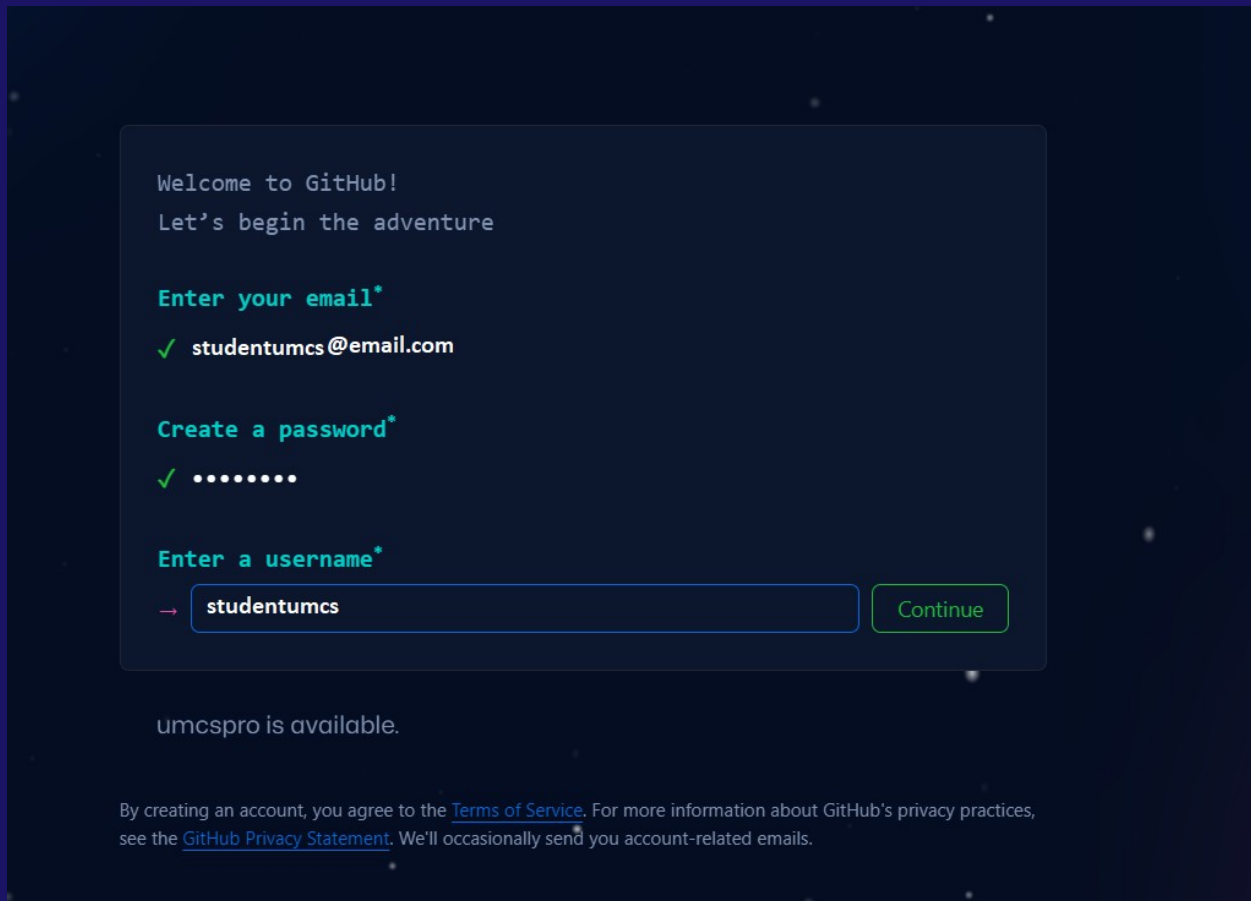
umcspro is available.

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.

# Github

Po utworzeniu konta, do uwierzytelniania podczas wysyłania zmian w kodzie na serwer najbezpieczniej jest używać **tokenów**.

**Token** w każdym momencie można usunąć, lub zmienić uprawnienia do niego przypisane.



The screenshot shows the GitHub account creation interface. It has a dark background with a central light gray box containing the form. The text is in a monospaced font. The form includes fields for email, password, and username, each with a green checkmark indicating successful input. A 'Continue' button is at the bottom right of the form. Below the form, there is a note about 'umcspro' and a footer with links to the Terms of Service and Privacy Statement.

Welcome to GitHub!  
Let's begin the adventure

Enter your email\*

✓ studentumcs@email.com

Create a password\*

✓ .....

Enter a username\*

→ studentumcs

Continue

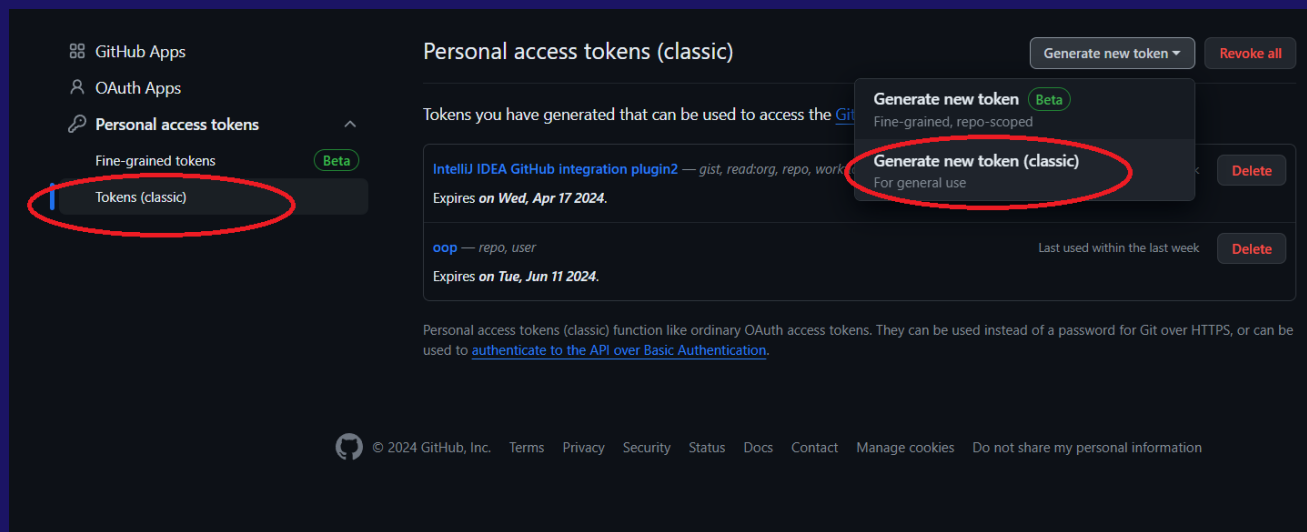
umcspro is available.

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.

# Github

Aby wygenerować token przechodzimy na stronie githuba klikając na ikonę naszego konta:

Settings>Developer settings>Personal access tokens>tokens (classic)



# Github

Wybieramy uprawnienia,  
Np tylko dostęp do  
repozytorium.

Generujemy i kopiujemy  
token. Klucz pojawia się tylko  
raz. Należy go skopiować aby  
móc go użyć.

GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens

Tokens (classic)

## New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

**Note**

repotoken

What's this token for?

**Expiration \***

30 days The token will expire on Wed, Apr 17 2024

**Select scopes**

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> <b>repo</b>	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> <b>workflow</b>	Update GitHub Action workflows
<input type="checkbox"/> <b>write:packages</b>	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> <b>delete:packages</b>	Delete packages from GitHub Package Registry



Tworzymy nowe repozytorium  
Na githubie.

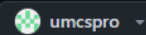
Aby utworzyć lokalne  
repozytorium i wysłać zmiany  
na serwer wykonujemy  
polecenia zgodnie z  
podpowiedzią githuba.

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

Owner \*



umcspro

Repository name \*

projektmavenowy

✔ mavenowyprojekt is available.

Great repository names are short and memorable. Need inspiration? How about [expert-octo-parakeet](#) ?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

You are creating a public repository in your personal account.

Create repository

# Github

Plik .gitignore – zawiera ścieżki plików i katalogów które nie będą wrzucane na serwer. Należy umieścić taki plik w katalogu projektu.

```
target/  
!.mvn/wrapper/maven-wrapper.jar  
!**/src/main/**/target/  
!**/src/test/**/target/
```

```
### IntelliJ IDEA ###  
.idea/modules.xml  
.idea/jarRepositories.xml  
.idea/compiler.xml  
.idea/libraries/  
*.iws  
*.iml  
*.ipr
```

```
### Eclipse ###  
.apt_generated  
.classpath  
.factorypath  
.project  
.settings  
.springBeans  
.sts4-cache
```

```
### NetBeans ###  
/nbproject/private/  
/nbbuild/  
/dist/  
/nbdist/  
/.nb-gradle/  
build/  
!**/src/main/**/build/  
!**/src/test/**/build/
```

```
### VS Code ###  
.vscode/
```

```
### Mac OS ###  
.DS_Store
```

# GitHub

**git init** - inicjuje lokalne repozytorium.  
**git add .** - Polecenie ustawia pliki jako śledzone. Używając kropki śledzimy wszystkie pliki z katalogu poza tymi opisanymi w .gitignore.  
**git branch -M main** – ustawia główną gałąź projektu.  
**git commit -m "first commit"** – zatwierdza zmiany w lokalnym repozytorium





# Git

## hub

**git init** - inicjuje lokalne repozytorium.  
**git add .** - Polecenie ustawia pliki jako śledzone. Używając kropki śledzimy wszystkie pliki z katalogu poza tymi opisanymi w .gitignore.  
**git branch -M main** – ustawia główną gałąź projektu.  
**git commit -m "first commit"** – zatwierdza zmiany w lokalnym repozytorium

Uwaga!! Git podpowie nam, że należy ustawić dane użytkownika (uzupełniamy tak jak na koncie gitgub):

```
git config --global user.email "you@example.com"  
git config --global user.name "Your Name"
```



# GitHub

**git init** - inicjuje lokalne repozytorium.

**git add .** - Polecenie ustawia pliki jako śledzone.

Używając kropki śledzimy wszystkie pliki z katalogu poza tymi opisanymi w .gitignore.

**git branch -M main** – ustawia główną gałąź projektu.

**git commit -m "first commit"** – zatwierdza zmiany w lokalnym repozytorium




**git remote add origin https://github.com/umcspro/projektmavenowy.git**



-polecenie sprawia, że repozytorium lokalne łączy się z repozytorium zdalnym




**git push origin main**


-wysyłamy zmiany na serwer. **Uwaga! Do uwierzytelnienia używamy wygenerowany token!**


# Github



 projekt.mavenowy Public






 Pin  Unwatch 1


 main  1 Branch  0 Tags


 Add file

 Code

 umcspro Merge branch 'main' of <https://github.com/umcspro/projekt.mavenowy> d117df2 · 15 minutes ago  7 Commits

 .idea	first commit	12 hours ago
 src/main/java/org/example	first commit	25 minutes ago
 .gitignore	first commit	25 minutes ago
 pass.txt	first commit	25 minutes ago
 pom.xml	first commit	25 minutes ago

 README



## Add a README

Help people interested in this repository understand your project by adding a README.

Add a README

# Git

## hub

Aby pobrać zmiany z serwera używamy:

**git pull origin main**

Aby sklonować repozytorium:

**git clone**

<https://github.com/umcspro/projektmavenowy.git>



# GitHub

Np po usunięciu projektu z katalogu IdeaProjects  
Klonuje projekt z repozytorium:

```
luka@DESKTOP-SBMVVMV MINGW64 ~/IdeaProjects
$ git clone https://github.com/umcspro/projektmavenowy.git
Cloning into 'projektmavenowy'...
remote: Enumerating objects: 38, done.
remote: Counting objects: 100% (38/38), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 38 (delta 8), reused 34 (delta 4), pack-reused 0
Receiving objects: 100% (38/38), 4.26 KiB | 2.13 MiB/s, done.
Resolving deltas: 100% (8/8), done.
```



# GitHub


Podpinam się jeszcze raz do zdalnego repozytorium (w tym wypadku nie jest to konieczne) dodaje nową klasę w intelij, dodaję ją do śledzenia, tworzę commita i wysyłam zmiany:



```
PS C:\Users\luke\IdeaProjects\projektmavenowy> git add .
PS C:\Users\luke\IdeaProjects\projektmavenowy> git remote add origin https://github.com/umcspro/projektmavenowy.git
error: remote origin already exists.
usage: git remote remove <name>


PS C:\Users\luke\IdeaProjects\projektmavenowy> git remote remove origin
PS C:\Users\luke\IdeaProjects\projektmavenowy> git remote add origin https://github.com/umcspro/projektmavenowy.git
PS C:\Users\luke\IdeaProjects\projektmavenowy> git commit -m "add new class NowaKlasa2"
[main f2be0f0] add new class NowaKlasa2
1 file changed, 4 insertions(+)
create mode 100644 src/main/java/org/example/NowaKlasa2.java
PS C:\Users\luke\IdeaProjects\projektmavenowy> git push origin main
```





# Github


 projekt.mavenowy Public






 Pin  Unwatch 1


 main 1 Branch 0 Tags


 Add file

 Code

 umcspro Merge branch 'main' of <https://github.com/umcspro/projekt.mavenowy> d117df2 · 15 minutes ago 7 Commits


 .idea	first commit	12 hours ago
 src/main/java/org/example	first commit	25 minutes ago
 .gitignore	first commit	25 minutes ago
 pass.txt	first commit	25 minutes ago
 pom.xml	first commit	25 minutes ago

 README



## Add a README

Help people interested in this repository understand your project by adding a README.



# Github

Usuwanie swoich danych z pracowni na linuxach:

```
git config --global --unset credential.helper  
rm ~/.git-credentials (jeżeli trzymane w pliku)
```

```
rm ~/.gitconfig
```



# **Dziękuję za uwagę!**

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik.