# Dependency injection with Dagger 2

|  |
| --- |
| <dependencies>  <dependency>  <groupId>com.google.dagger</groupId>  <artifactId>dagger</artifactId>  <version>2.4</version>  </dependency>  <dependency>  <groupId>com.google.dagger</groupId>  <artifactId>dagger-compiler</artifactId>  <version>2.4</version>  <optional>true</optional>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.3</version>  <configuration>  <source>1.8</source>  <target>1.8</target>  </configuration>  </plugin>  </plugins>  </build> |

Вначале напишем пример без испольнования dependency injection:

|  |
| --- |
| public class Main {      public static void main(String[] args) {          Application app = new Application();          app.run();      }  } |
| public class Application {      private Service service;      public Application() {          service = new Service();      }      public void run() {          System.out.println("Application start.");          service.doWork();          System.out.println("Application end.");      }  } |
| package com.bisaga.simple;  public class Service {  */\* @Inject  public Service(){}\*/*      public void doWork() {          System.out.println("Service do work.");      }  } |

Теперь нам нужно создать объект «Module», где мы реализуем метод «provide», чтобы предоставить новый экземпляр класса Service. Класс модуля должен иметь аннотацию @Module, а методы должны иметь аннотации @Provides.

|  |
| --- |
| import dagger.Module;  import dagger.Provides;  *@Module*  public class SimpleModule {  *@Provides*      Service provideService() {          return new Service();      }  } |

Теперь нам нужен объект «Component», который является интерфейсом, из которого dagger генерирует класс инжектора. Компонент должен содержать аннотацию @Component с определенными модулями и вводить сигнатуру функции.

|  |
| --- |
| import dagger.Component;  import javax.inject.Singleton;  *@Singleton*  *@Component*(modules = { SimpleModule.class})  public interface SimpleComponent {      void inject(Application application);  } |

В конструкторе Application мы инициализируем dagger функциями build () и inject (). Инъекция достигается с помощью аннотации @Inject перед локальным объявлением service.

|  |
| --- |
| package com.bisaga.simple;  import javax.inject.Inject;  public class Application {      private SimpleComponent component;  *@Inject* Service service;      public Application() {          component = DaggerSimpleComponent.builder().build();          component.inject(this);      }      public void run() {          System.out.println("Application start.");          service.doWork();          System.out.println("Application end.");      }  } |

