Praktikum 2

Injeksi Ketergantungan

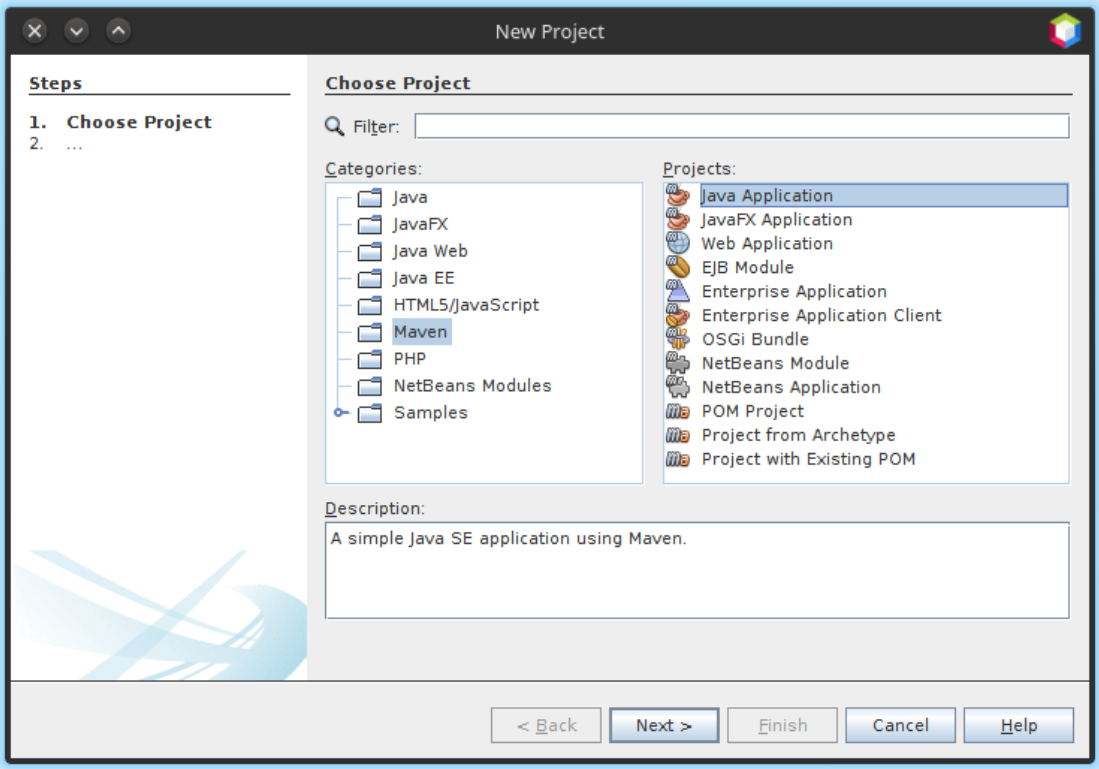
1. Pengantar

- Injeksi ketergantungan adalah suatu cara yang disediakan oleh framework untuk memasukkan kontruktor, metode, ke dalam metode atau konstruktor lainnya.
- Framework spring telah menyediakan injeksi ini dengan berbagai macam cara, mahasiswa diharapkan dapat memahami cara kerja injeksi ketergantungan yang sudah disediakan oleh Spring.
- Proses injeksi dan koding dilakukan menggunakan Netbeans dengan template dari Maven

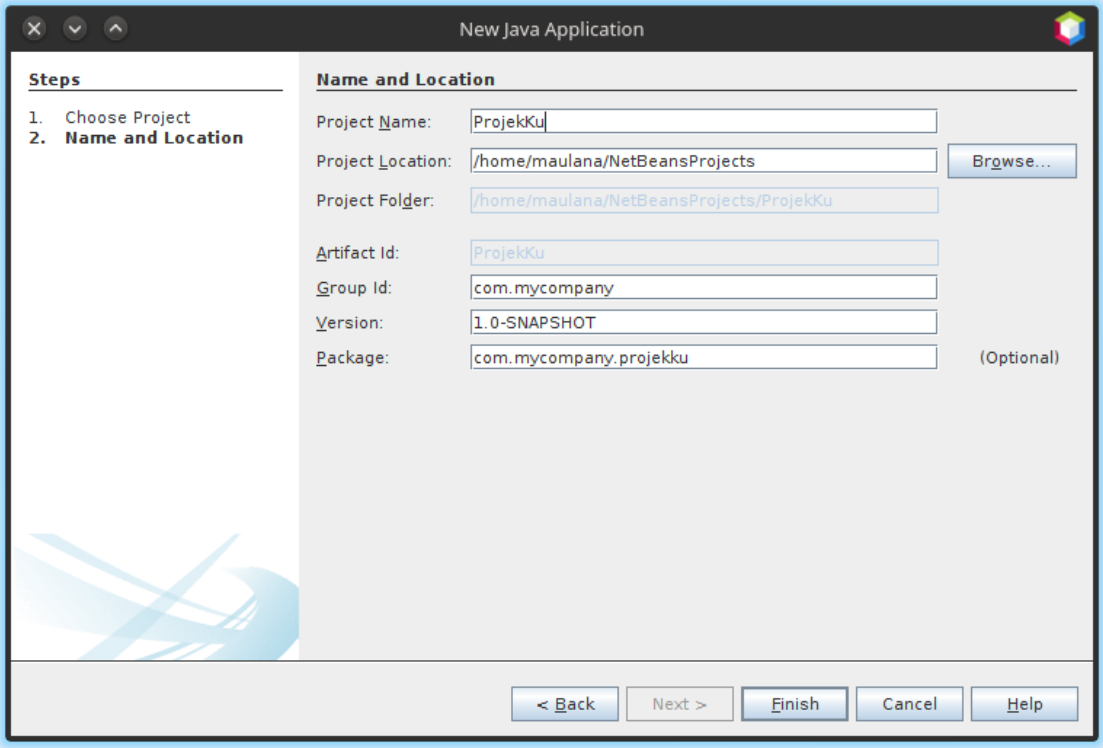
2. Pembahasan

Praktikum ini memiliki tujuan untuk memberikan pemahaman kepada mahasiswa bagaimana cara melakukan injeksi ketergantungan (Dependency Injection) secara mudah dan cepat. Injeksi dilakukan tanpa menggunakan XML untuk mempermudah pemahaman mahasiswa dalam membuat projek dengan menggunakan framework Spring.

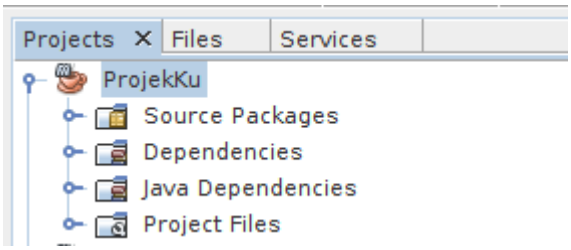
1) Praktikum dimulai dengan membuat projek Maven baru



2) Lalu klik Next untuk melanjutkan, berikan nama sesuai keinginan



3) Lalu klik Finish untuk menyelesaikan pembuatan projek awal. Maka Netbeans akan membuat projek beserta folder-folder seperti berikut:

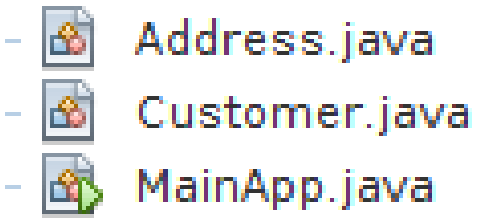


4) Source packages digunakan untuk kode-kode Java dan paket-paketnya, dan Project Files untuk ketergantungan dengan Spring. Kita buka Project Files, lalu buka pom.xml. Ketikkan tambahan kode berikut ini (huruf Bold)

```
<properties>
   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
   <maven.compiler.source>1.8</maven.compiler.source>
   <maven.compiler.target>1.8</maven.compiler.target>
   <spring.version>3.0.5.RELEASE</spring.version>
</properties>

<dependencies>
      <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.8.2</version>
            <scope>test</scope>
      </dependency>

      <!-- Spring 3 dependencies -->
      <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>${spring.version}</version>
      </dependency>

      <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${spring.version}</version>
      </dependency>
</dependencies>
```

5) Setelah itu klik Build agar Netbeans mengunduh framework-framework yang dibutuhkan secara otomatis.

6) Jika sudah selesai Build, kita dapat memulai koding Java. Buatlah file Java class baru di dalam package yang ada di folder Source Packages. Buatlah Address.java; Customer.java; MainApp.java seperti berikut:



7) Buka Address.java lalu ketikkan kode berikut ini:

```
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

@Component("addressBean")
public class Address {
   @Value("Block ABC, LakeView")
   private String street;

   @Value("98700")
   private int postcode;

   @Value("US")
   public String country;

   public String getFullAddress(String prefix) {

         return prefix + " : " + street + " " + postcode + " " + country;
   }

   public void setCountry(String country) {
         this.country = country;
   }

   @Override
   public String toString() {
         return "Address [street=" + street + ", postcode=" + postcode
                     + ", country=" + country + "]";
   }
}
```

8) Berikutnya buka Customer.java lalu ketik kode berikut:

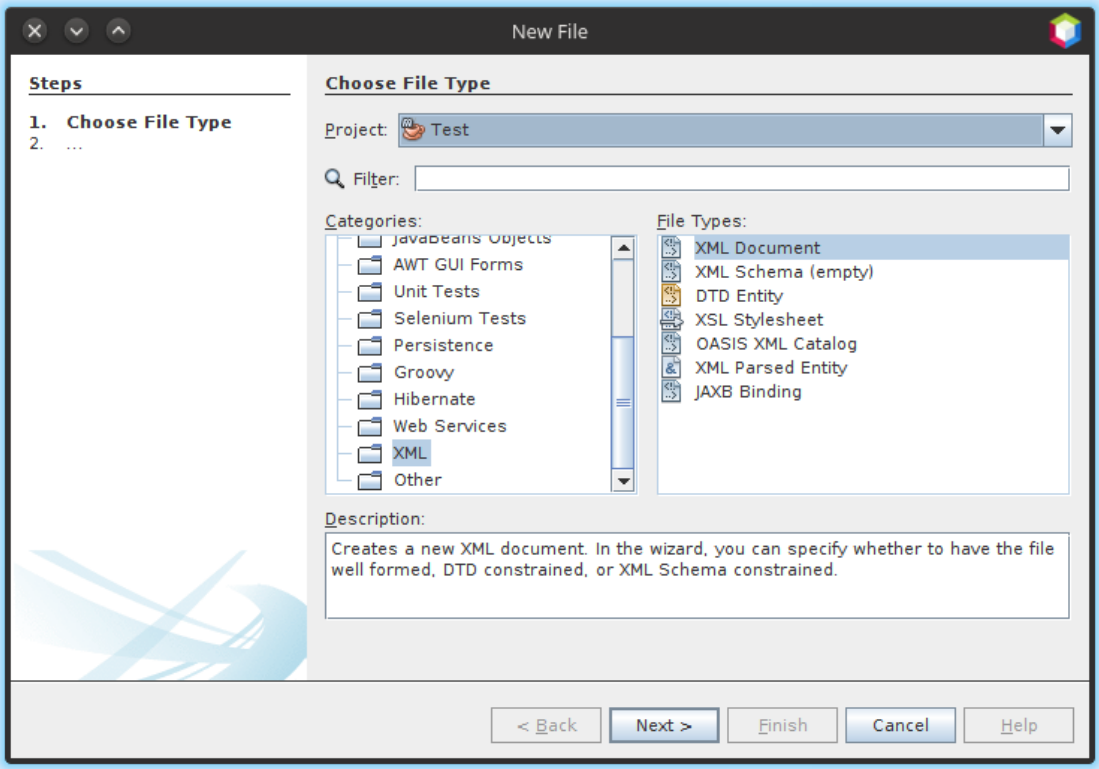
```
package com.mycompany.test;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
```

```
@Component("customerBean")
public class Customer {
   @Value("#{addressBean}")
   private Address address;

   @Value("#{addressBean.country}")
   private String country;

   @Value("#{addressBean.getFullAddress('mkyong')}")
   private String fullAddress;

   @Override
   public String toString() {
         return "Customer [address=" + address + "\n, country=" + country
                     + "\n, fullAddress=" + fullAddress + "]";
   }
}
```

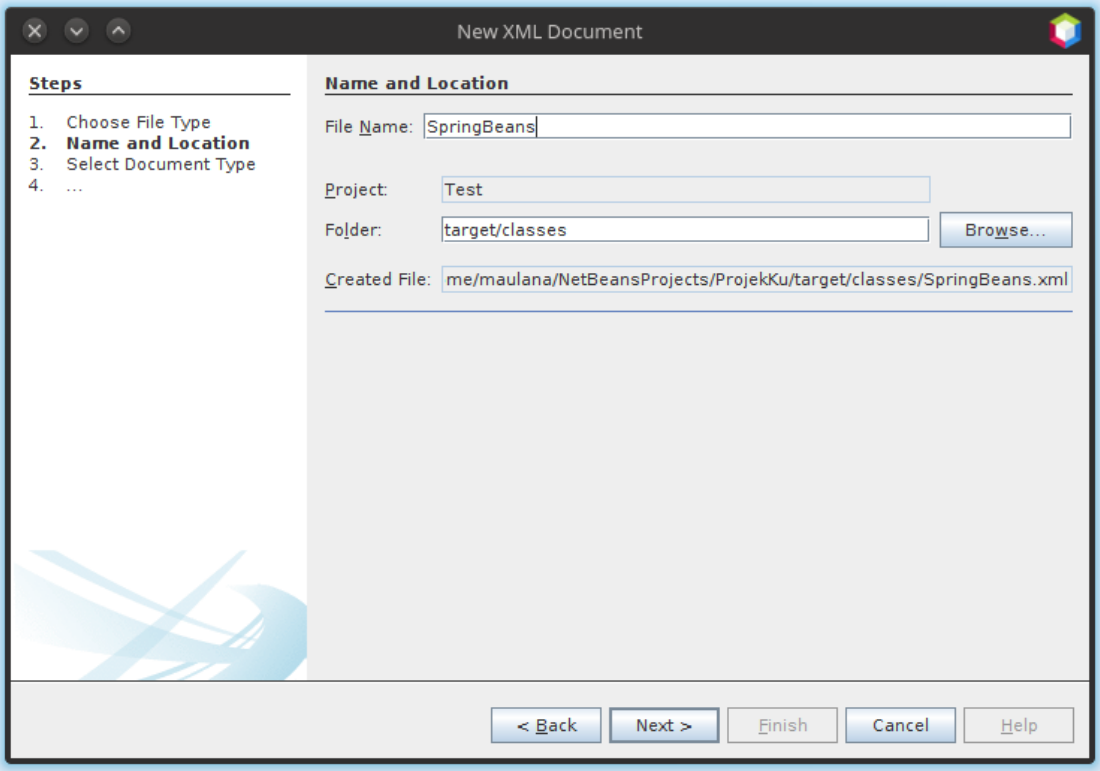
9) Lalu buka MainApp.java lalu ketik kode berikut

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

/**
 *
 * @author maulana
 */
public class MainApp {
   public static void main(String[] args) {
         ApplicationContext context = new ClassPathXmlApplicationContext("SpringBeans.xml");
         Customer obj = (Customer) context.getBean("customerBean");
         System.out.println(obj);
  }
}
```

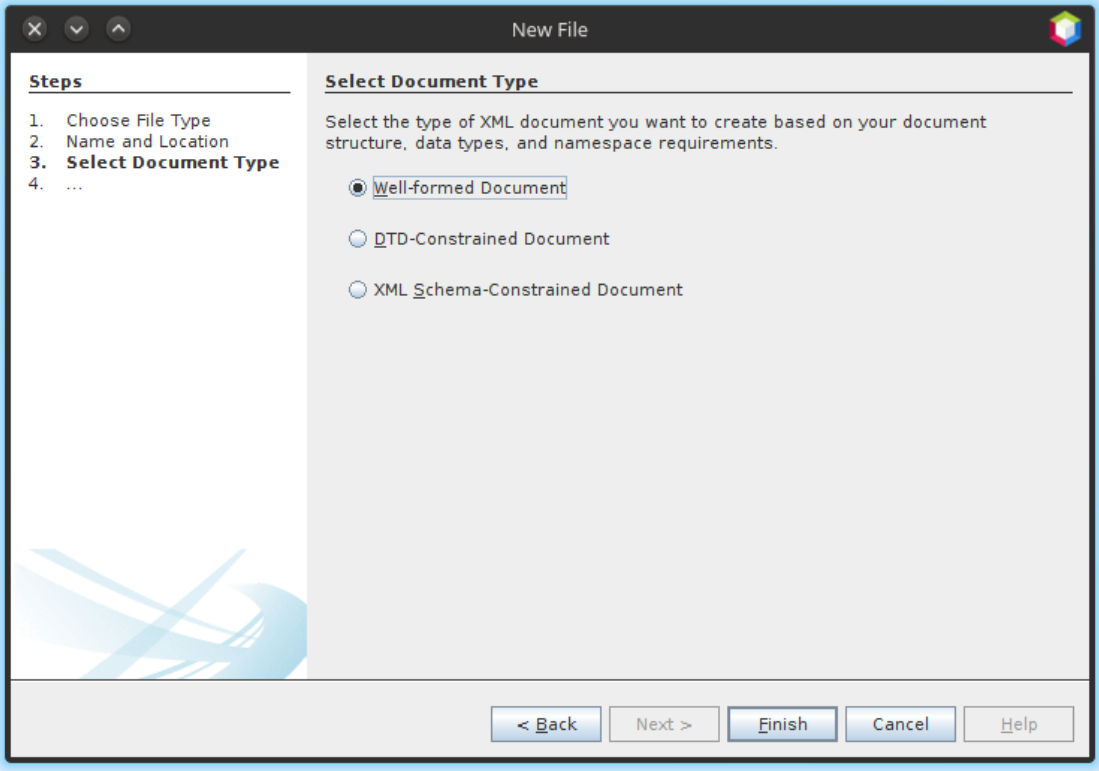
10) Pindah tab Project ke Files, navigasikan ke target > classes. Klik kanan, lalu pilih New > XML Document (gunakan Other > XML > XML Document)



11) Pastikan XML document, lalu klik Next. Berikan nama SpringBeans.xml lalu klik Next.



12) Pilih Well-formed Document lalu klik Finish



13) Buka SpringBeans.xml lalu ketik kode berikut ini: Pastikan tulisan bold disesuaikan dengan nama paket di folder Source Packages

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:context="http://www.springframework.org/schema/context"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
                        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
                        http://www.springframework.org/schema/context
                        http://www.springframework.org/schema/context/spring-context-3.0.xsd">

       <context:component-scan base-package="com.mycompany.test" />
</beans>
```

14) Jika sudah, kita dapat menjalankan aplikasi yang dibuat, jangan lupa untuk mem-Build ulang projek sebelum di-Run. Jika sukses, akan memperlihatkan teks sebagai berikut

```
Customer [address=Address [street=Block ABC, LakeView, postcode=98700, country=US]
, country=US
, fullAddress=mkyong : Block ABC, LakeView 98700 US]
```

15) Kita dapat mengubah info @Value sesuai dengan keinginan kita.