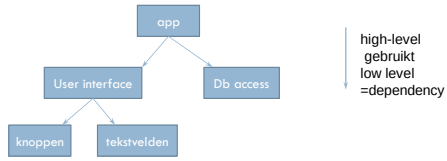
INVERSION OF CONTROL, DEPENDENCY INJECTION, SPRING FRAMEWORK

Veerle Ongenae, Wijnand Schepens

Traditioneel

Traditionele (procedurele) applicaties:

High-level-componenten samenstellen uit low-level-componenten (zelfgemaakte en/of libraries)



Framework

- Het framework regelt het globale applicatieverloop
- Clientcode kan ingeplugd worden
- Vb. event-based programmeren, zoals in Java Swing:
  - Windows regelt vensters, low-level I/O enz.
  - Clientcode registreert eventhandlers (callbacks) om op bepaalde events te reageren
- "Instead of your program running the system, the system runs your program"

Het Hollywoodprincipe

- Don't call us, we'll call you
- Callback, eventhandler, listener, observer, hook
- Template method
- Servlets
- Applets
- …

Inversion of Control

- Framework
  - Roept applicatiecode op
  - Coördineert de workflow
- Spring
  - Framework dat Inversion of Control toepast
  - Ontwerper hoeft alleen "gewone" Java-klassen te schrijven (POJO: Plain Old Java Objects)

Dependency Injection

- Specifieker dan IoC
- Traditioneel
  - Pull
  - Applicatie haalt afhankelijkheden op uit zijn omgeving
- Dependency Injection (DI)
  - Push
  - Duwt afhankelijkheden in de applicatie (push)
  - Specifieke vorm van IoC

Dependency Injection: bad code

7

In "The Dependency Inversion Principle" (or **DIP**), the author states the three defining factors of "bad code":

- RIGIDITY: It is hard to change because every change affects too many other parts of the system
- FRAGILITY: When you make a change, unexpected parts of the system break
- IMMOBILITY: It is hard to reuse in another application because it cannot be disentangled from the current application[3]

Principes Dependency Injection

8

- *1. High level modules should not depend upon low level modules, both should depend upon abstractions.*
- *2. Abstractions should not depend upon details, details should depend upon abstractions.*

Dependency Injection

9

- Dependency (afhankelijkheid)
  - Object A gebruikt object B
  - Object A is afhankelijk van object B
- Geen Inversion of Control
  - A instantieert B
- Inversion of Control
  - Een andere "entiteit" brengt maakt B en koppelt dit aan A

Spring Framework

10

- Framework dat Inversion of Control toepast
- Ontwerper hoeft alleen "gewone" Java-klassen te schrijven (POJO: Plain Old Java Objects)
- + veel meer (MVC, AOP, …)

Voorbeeld 1

11

```
interface WeerDAO
{
      WeerData zoek(Date datum);
}

public class WeerDAOImpl implements WeerDAO
{
      // ...
}
```


Voorbeeld 1 zonder DI

12

```
public class WeerDienst
{
    WeerDAO weerDao = new WeerDAOImpl();

    public Double getHoogsteTemp(Date datum)
    {
        WeerData wData = weerDao.zoek(datum);
        Double hoogste = null;
        if (wData != null)
            hoogste = new Double(wData.getHoogste());
        return hoogste;
    }
}
```

afhankelijkheid

Testen voorbeeld 1

13

```
public class Main
{
    public static void main(String[] args)
    {
        WeerDienst weerDienst = new WeerDienst();

        Double hoogste = weerDienst.getHoogsteTemp(
            new GregorianCalendar(2009,22,4).getTime());

        System.out.println(hoogste);
    }
}
```


Voorbeeld 1

14

- WeerDienst afhankelijk van
  - WeerDAO
    - Interface
    - Niet zo erg
  - WeerDAOImpl
    - Implementatie van de interface WeerDAO
    - Te vermijden

Voorbeeld 2 met DI

15

```
public class WeerDienst
{                    afhankelijkheid
    private WeerDAO weerDAO;
                                  geïnjecteerd van buitenaf
    public void setWeerDAO(WeerDAO weerDAO)
    {
        this.weerDAO = weerDAO;
    }
    public Double getHoogste(Date datum) {
        // ...
    }
}
```


Voorbeeld 2

16

Configuratiebestand: WeerSpringXMLConfig2.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" …">

    <bean id="weerDao"
          class="springvoorbeeld.WeerDAOImpl“ />

</beans>
```


Testen voorbeeld 2

17

```
public static void main(String[] args)
{
    ApplicationContext ctx = new ClassPathXmlApplicationContext(
       "springvoorbeeld/WeerSpringXMLConfig2.xml");

    WeerDAO weerDAO = (WeerDAO) ctx.getBean("weerDao");

    WeerDienst weerDienst = new WeerDienst();
    weerDienst.setDAO(weerDAO);

    Double hoogste = weerDienst.getHoogsteTemp(
         new GregorianCalendar(2009,22,4).getTime());
    System.out.println(hoogste);
}
```


Voorbeeld 2

18

- Framework instantieert WeerDAO
  - Geconfigureerd in XML-bestand
- WeerDAO
  - Interface: geen implementatiedetails
    - Implementatie kan veranderen
- WeerDAOImpl
  - Enkel afhankelijk van interface WeerDAO

Voorbeeld 3 met meer DI

19

```
public interface WeerDienst
{
    public Double getHoogste(Date datum);
}
public class WeerDienstImpl implements WeerDienst
{
    private WeerDAO weerDao;

    public void setWeerDao(WeerDAO weerDao)
    {
        this.weerDao = weerDao;
    }
    public Double getHoogste(Date datum) {
        // ...
    }
}
```

afhankelijkheid

geïnjecteerd van buitenaf

Voorbeeld 3

20

Configuratiebestand: WeerSpringXMLConfig3.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" …">

    <bean id="weerDienst"
          class="springvoorbeeld.WeerDienstImpl">
        <property name="weerDao"
                  ref="weerDao" />
    </bean>

    <bean id="weerDao"
          class="springvoorbeeld.WeerDAOImpl">
    </bean>
</beans>
```


Testen voorbeeld 3

21

```
public static void main(String[] args)
{
    ApplicationContext ctx = new ClassPathXmlApplicationContext(
        "springvoorbeeld/WeerSpringXMLConfig3.xml");

    WeerDienst weerDienst = (WeerDienst) ctx.getBean("weerDienst");

    Double hoogste = weerDienst.getHoogste(
         new GregorianCalendar(2009,22,4).getTime());
    System.out.println(hoogste);
}
```


Voorbeeld 3

22

- Framework brengt WeerDienst en WeerDAO samen
  - Geconfigureerd in XML-bestand
- WeerDienst
  - Interface: geen implementatiedetails
    - Implementatie kan veranderen
- WeerDienstImpl
  - Enkel afhankelijk van interface WeerDAO
  - Attribuut weerDao wordt door Spring-framework ingevuld

Voordeel Dependency Injection

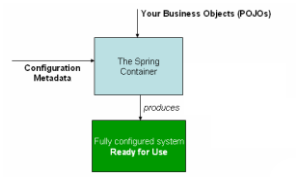
23

- Code onafhankelijk van framework
- Herbruikbare code
- Programmeren naar interface i.p.v. naar klassen

Spring IoC-Containter

24

- Beans
  - Objecten
  - Kern applicatie
  - Beheerd door IoC-Container

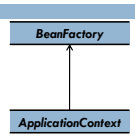


bron: http://static.springsource.org

Spring IoC-Containter

25

- org.springframework.beans
  - Interface BeanFactory (uit org.springframework.beans.factory)
    - Beheer objecten
    - Configuratie
- org.springframework.context
  - Interface ApplicationContext
    - Extra functionaliteit zoals bv. id, timestamp, …



ApplicationContext

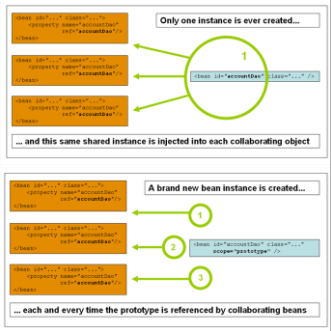
26

- Stelt IoC-Container voor
- Aanmaken, configureren en beheren beans
- Specifiëren beans: metadata
  - XML
  - Annotaties
  - Java-code
- Implementatie bij eenvoudige applicaties (stand alone)
  - ClassPathXmlApplicationContext
  - FileSystemXmlApplicationContext

Metadata

27

- Beschrijving beans
  - id
  - class
  - property
  - scope
    - singleton (default)
      - 1 per container
    - prototype
      - telkens een nieuw object
  - …



Dependency Injection

28

- Afhankelijkheden initialiseren
  - Constructor-based:

```
<bean id="weerDienst“ class="springvoorbeeld.WeerDienstImpl">
        <constructor-arg ref="weerDao"/>
 </bean>
```

  - Setter-based:

```
<bean id="weerDienst" class="springvoorbeeld.WeerDienstImpl">
        <property name="weerDao“ ref="weerDao"/>
</bean>
```


Gebruik Spring in Netbeans

29

- Voeg bij de Libraries het Spring Framework toe
- Voeg een “Spring XML Configuration File” toe
  - Vind je in “Other” onder het menu “New File”

Infobronnen

30

- “Professional Java Development with the Spring Framework”, Rod Johnson, Juergen Hoeller, Alef Arendsen, Thomas Risberg, Colin Samaleanu, Wrox, Wiley Publishing, Inc., 2005
- “Introduction to the Spring Framework”, Rod Johnson, 1 May 2005, TheServerSide.com (http://www.theserverside.com/news/1364527/Introduction-to-the-Spring-Framework)

Infobronnen

- "Putting the Spring in Your Step (and Application)", Josh Long, November 9th, 2010,Green Beans, Spring (http://blog.springsource.com/2010/11/09/green-beans-putting-the-spring-in-your-step-and-application/)
- Spring Framework, Reference Documentation, 3.0 (http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/)