01. Fundamentalen van Spring

## Inleiding

Het Spring Framework is een Java-gebaseerd framework dat is ontworpen om ontwikkelaars te helpen bij het bouwen van enterprise-level applicaties. Het is open-source en biedt een breed scala aan functies en functionaliteiten die kunnen worden gebruikt om diverse aspecten van een applicatie te beheren. Het framework is gebouwd op basis van een aantal kernmodules, waaronder Spring Core, Spring MVC en Spring Data.

## Waarom Spring?

Een van de belangrijkste functies van het Spring Framework is dependency injection. Dit is een techniek waarbij objecten worden voorzien van de afhankelijkheden die ze nodig hebben om correct te functioneren. Het biedt ook ondersteuning voor aspect-oriented programming, waarmee ontwikkelaars de mogelijkheid hebben om cross-cutting concerns, zoals logging en beveiliging, in te bouwen in hun applicatie.

Het Spring Framework biedt ook ondersteuning voor data access. Dit omvat onder andere JDBC, ORM en NoSQL-databases. Hierdoor kunnen ontwikkelaars eenvoudig en efficiënt gegevens in hun applicatie opslaan en ophalen.

## Enterprise-niveau

Het framework biedt ook ondersteuning voor webapplicationontwikkeling met behulp van de Spring MVC-module. Hiermee kunnen ontwikkelaars eenvoudig en efficiënt webapplicaties bouwen en beheren.

Al met al is het Spring Framework een krachtige tool voor het bouwen van enterprise-level applicaties. Het biedt een breed scala aan functies en functionaliteiten, waaronder dependency injection, aspect-oriented programming, data access en webapplicationontwikkeling. Hierdoor is het een populaire keuze onder ontwikkelaars en wordt het gebruikt in veel toonaangevende bedrijven en organisaties over de hele wereld.

## Installatie van Spring Framework

Om te beginnen met het werken met Spring Framework, moet u het framework downloaden en installeren. Spring Framework kan worden gedownload van de officiële website. U kunt Spring Framework downloaden van de officiële website (<https://spring.io/>). Klik op de knop 'Download' op de startpagina en selecteer de gewenste versie. Klik op de link om het ZIP-bestand te downloaden. Nadat u het ZIP-bestand hebt gedownload, pakt u het uit en plaatst u de inhoud in uw werkmap.

## Basisconcepten van Spring Framework

Spring Framework is gebaseerd op twee belangrijke concepten:

1. Inversion of Control (IoC)
2. Dependency Injection (DI)

### Inversion of Control (IoC)

Inversion of Control (IoC) is een ontwerpconcept waarbij de controle over de uitvoering van code wordt omgedraaid. In plaats van dat uw code de controle heeft over het aanroepen van methoden, wordt de controle overgedragen aan het framework. Het framework beheert de levenscyclus van objecten en roept methoden aan wanneer dat nodig is.

### Dependency Injection (DI)

Dependency Injection (DI) is een ontwerpconcept waarbij objecten afhankelijk zijn van andere objecten. Met DI worden objecten geïnjecteerd in de code van andere objecten, zodat ze kunnen samenwerken om de gewenste functionaliteit te bieden.

Stel dat u een klasse heeft die afhankelijk is van een andere klasse om zijn werk te doen. In plaats van de afhankelijke klasse te maken met behulp van het 'nieuwe' trefwoord, kunt u Spring Framework gebruiken om de afhankelijke klasse automatisch in te voegen.

Voorbeeld:

public class CustomerService {  
 private CustomerDao customerDao;  
  
 // setter method for customerDao  
 public void setCustomerDao(CustomerDao customerDao) {  
 this.customerDao = customerDao;  
 }  
   
 public void save(Customer customer) {  
 customerDao.save(customer);  
 }  
}  
Copy

In dit voorbeeld is CustomerService afhankelijk van CustomerDao. Spring Framework zal de CustomerDao-instantie maken en deze automatisch invoegen in de setter-methode.

Dependency Injection (DI):

In plaats van de afhankelijke klasse expliciet te maken met behulp van het 'nieuwe' trefwoord, kunt u Spring Framework gebruiken om de afhankelijke klasse automatisch in te voegen.

Voorbeeld:

public class CustomerService {  
 private CustomerDao customerDao;  
  
 // constructor injection for customerDao  
 public CustomerService(CustomerDao customerDao) {  
 this.customerDao = customerDao;  
 }  
   
 public void save(Customer customer) {  
 customerDao.save(customer);  
 }  
}  
Copy

In traditionele Java-programmering heeft de applicatiecode controle over het aanroepen van methoden. Met Spring Framework wordt de controle overgedragen aan het framework. Dit betekent dat de objecten worden beheerd door Spring en het framework roept methoden aan wanneer dat nodig is. Hier is een voorbeeld:

public class MyApp {  
 public static void main(String[] args) {  
 MyService myService = new MyService(); // traditionele manier van instantiëring van een object  
 myService.doSomething();  
 }  
}  
  
public class MyService {  
 public void doSomething() {  
 System.out.println("Do something");  
 }  
}  
Copy

Met Spring Framework zouden we dit als volgt herschrijven:

public class MyApp {  
 public static void main(String[] args) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 MyService myService = (MyService) context.getBean("myService");  
 myService.doSomething();  
 }  
}  
  
public class MyService {  
 public void doSomething() {  
 System.out.println("Do something");  
 }  
}  
Copy

In dit voorbeeld hebben we een klasse die afhankelijk is van een andere klasse. Met behulp van DI kunnen we de afhankelijke klasse injecteren in de code van de andere klasse.

public class OrderService {  
 private CustomerService customerService;  
  
 public OrderService(CustomerService customerService) {  
 this.customerService = customerService;  
 }  
  
 public void placeOrder() {  
 Customer customer = customerService.getCustomer();  
 // plaats bestelling  
 }  
}  
  
public class CustomerService {  
 public Customer getCustomer() {  
 // haal klantgegevens op  
 }  
}  
Copy

## Spring-toepassingscontext

De Spring-toepassingscontext is een container voor objecten die worden gemaakt met Spring Framework. Het is verantwoordelijk voor het beheren van de levenscyclus van objecten en het bieden van configuratie-informatie voor de applicatie.

De Spring-toepassingscontext is verantwoordelijk voor het beheren van de levenscyclus van objecten en het bieden van configuratie-informatie voor de applicatie.

public class MyApp {  
 public static void main(String[] args) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 MyService myService = (MyService) context.getBean("myService");  
 myService.doSomething();  
 }  
}  
Copy

## Configuratie van Spring-toepassingscontext

De Spring-toepassingscontext wordt geconfigureerd met behulp van XML-bestanden of Java-configuratieklassen. XML-bestanden zijn het meest gebruikelijk en bieden een gestructureerde manier om de configuratie-informatie te definiëren. Java-configuratieklassen bieden echter meer flexibiliteit en maken gebruik van Java-code om de configuratie-informatie te definiëren.

Spring-toepassingscontext kan worden geconfigureerd met behulp van XML-bestanden of Java-configuratieklassen.

Voorbeeld van configuratie met XML-bestand:

<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">  
  
 <bean id="myService" class="com.example.MyService"></bean>   
 <bean id="customerService" class="com.example.CustomerService">   
 <!-- configureer de afhankelijkheden voor customerService -->   
 </bean>   
</beans>  
Copy

Voorbeeld van configuratie met Java-configuratieklasse:

@Configuration  
public class AppConfig {  
 @Bean  
 public MyService myService() {  
 return new MyService(customerService());  
 }  
  
 @Bean  
 public CustomerService customerService() {  
 return new CustomerServiceImpl();  
 }  
}  
Copy

## Spring Framework-modules

Spring Framework is opgedeeld in verschillende modules, waaronder:

1. Core Container: Dit is de basislaag van Spring Framework en biedt de kernfunctionaliteit van het framework, zoals IoC en DI.
2. Data Access / Integratie: Dit module biedt ondersteuning voor het werken met databases en biedt integratie met andere technologieën zoals Hibernate.
3. Web: Dit module biedt ondersteuning voor webapplicatieontwikkeling, inclusief ondersteuning voor MVC-frameworks.

## Conclusie

Spring Framework is een zeer krachtig Java-framework dat kan worden gebruikt om enterprise-level applicaties te bouwen. Het biedt een breed scala aan functies en functionaliteiten, waaronder dependency injection, aspect-oriented programming, data access en webapplicationontwikkeling. Door het gebruik van Spring Framework kunnen ontwikkelaars gemakkelijk betrouwbare, schaalbare en flexibele applicaties bouwen.