Spring

itconsulting
Consulenza & Formazione

Spring

Andremo ora a vedere tutte le nozioni di base di spring come: l' inversione del controllo (IoC), la dependency injection, gli ambiti bean, il ciclo di vita dei bean, gli INTERNAL bean , cablaggio automatico, programmazione orientata agli aspetti (AOP), accesso al database (JDBC), gestione delle transazioni, framework Web MVC, flusso Web, gestione delle eccezioni e altro ancora.

Spring Framework è uno dei framework Java EE più popolari. È un framework open source e leggero creato da Rod Johnson nel giugno 2003.

Principi fondamentali di Spring Framework:

- **Programmazione orientata agli aspetti (AOP).**
- **Iniezione di dipendenza (DI).**

itconsulting
Consulenza & Formazione

Spring

Andremo ora a vedere tutte le nozioni di base di spring come: l' inversione del controllo (IoC), la dependency injection, gli ambiti bean, il ciclo di vita dei bean, gli INTERNAL bean , cablaggio automatico, programmazione orientata agli aspetti (AOP), accesso al database (JDBC), gestione delle transazioni, framework Web MVC, flusso Web, gestione delle eccezioni e altro ancora.

Spring Framework è uno dei framework Java EE più popolari. È un framework open source e leggero creato da Rod Johnson nel giugno 2003.

Principi fondamentali di Spring Framework:

- **Programmazione orientata agli aspetti (AOP).**
- **Iniezione di dipendenza (DI).**

itconsulting
Consulenza & Formazione

Inversione del controllo (IoC)

Nell'ingegneria del software , l' inversion of control è un modello di progettazione in cui parti di un programma scritte su misura ricevono il flusso di controllo da un framework generico .

Un'architettura software con questo design inverte il controllo rispetto alla programmazione procedurale: nella programmazione tradizionale, il codice personalizzato che esprime lo scopo del programma richiama librerie riutilizzabili per occuparsi di compiti generici, ma con l'inversione del controllo, è il framework che richiama il codice personalizzato o specifico dell'attività. L'inversione del controllo viene utilizzata per aumentare la modularità del programma e renderlo estensibile , e ha applicazioni nella OOP e in altri paradigmi.

Il termine è correlato, ma diverso, al principio di inversione della dipendenza , che si occupa di disaccoppiare le dipendenze tra livelli di alto e basso livello attraverso astrazioni condivise . Il concetto generale è anche correlato alla programmazione guidata dagli eventi in quanto è spesso implementata utilizzando IoC in modo che il codice personalizzato sia comunemente interessato solo alla gestione degli eventi, mentre il ciclo di eventi e l'invio di eventi/messaggi è gestito dal framework o l'ambiente di esecuzione.

itconsulting
Consulenza & Formazione

Dependency injection,

Dependency injection (DI) è un design pattern della Programmazione OOP il cui scopo è quello di semplificare e migliorare la testabilità di software di grandi dimensioni, per utilizzare tale design pattern è sufficiente dichiarare le dipendenze di cui un componente necessita (dette anche interface contract).

Quando il componente verrà istanziato, un iniettore si prenderà carico di risolvere le dipendenze, attuando dunque l'Inversion of control. Se è la prima volta che si tenta di risolvere una dipendenza, l'injector istanzierà il componente dipendente, lo salverà in un contenitore di istanze e lo restituirà. Se non è la prima volta, allora restituirà la copia salvata. Una volta risolte tutte le dipendenze, il controllo può tornare al componente applicativo. Il pattern coinvolge almeno tre elementi:

- ***Una componente dipendente,***
- ***Le dipendenze del componente dichiarate, definite come interface contract,***
- ***Un injector (chiamato anche provider o container) che crea, a richiesta, le istanze delle classi che implementano delle dependency interface.***

itconsulting
Consulenza & Formazione

programmazione orientata agli aspetti

La programmazione orientata agli aspetti è un paradigma di programmazione basato sulla creazione di entità software denominate aspetti che sovrintendono alle interazioni fra gli oggetti finalizzate ad eseguire un compito comune. Il vantaggio rispetto alla tradizionale Programmazione orientata agli oggetti consiste nel non dover implementare separatamente in ciascuna classe il codice necessario ad eseguire questo compito comune.

Un programma aspect-oriented è costituito essenzialmente da due insiemi di costrutti: gli aspetti e gli oggetti. Gli aspetti sono delle entità esterne agli oggetti che osservano il flusso del programma generato dalle interazioni tra oggetti, modificandolo quando opportuno. Se paragonassimo gli oggetti a degli attori, potremmo dire che gli aspetti sono degli spettatori, cioè delle entità che osservano le azioni in corso nel programma senza esservi direttamente coinvolti, ma degli spettatori un po' particolari, visto che in determinate circostanze salgono sul palcoscenico e partecipano alla rappresentazione (l'esecuzione del programma, fuor di metafora).

itconsulting
Consulenza & Formazione

programmazione orientata agli aspetti

È noto che ci sono varie ragioni per cui la Programmazione orientata agli oggetti (in inglese OOP - Object Oriented Programming) è così diffusa oggigiorno. Alcune delle più importanti sono la semplicità dell'ingegnerizzazione del software tramite un design object-oriented, l'incremento nella manutenibilità del codice e le aumentate possibilità di riuso dello stesso. Tuttavia, col passare degli anni ci si è resi conto che il modello di sviluppo object-oriented non si adatta perfettamente alla modellazione di tutti i problemi che un sistema software deve risolvere.

Ad esempio, uno dei principi fondamentali dell'OOP è la modellazione del programma come uno scambio di messaggi tra oggetti, i quali sono entità tra loro indipendenti. Tale principio garantisce sicuramente la modularità del sistema software sviluppato con un linguaggio object oriented, ma al tempo stesso rende difficile l'implementazione di alcune funzionalità che per loro natura sono comuni a più oggetti (quali ad es. logging e sicurezza). Ci sono cioè alcuni compiti del sistema informativo che non possono essere modellati come oggetti, semplicemente perché interessano l'applicazione nel suo insieme.

programmazione orientata agli aspetti

Prendiamo ad esempio un'applicazione che deve effettuare il logging di alcune transazioni che richiedano l'interazione tra più oggetti: ognuno degli oggetti coinvolti deve, nei propri metodi, contenere codice in grado di gestire il suddetto problema; si viene così a creare, nella stesura del codice, una ridondanza non necessaria. Inoltre gli oggetti modificati a tale scopo diventano più complessi e quindi diminuisce la manutenibilità del programma.

L'aspect oriented programming (AOP) è nato con lo scopo di risolvere problemi di questo tipo. Gli aspetti modellano cioè le problematiche trasversali agli oggetti stessi, ossia compiti (quali ad esempio l'accounting) che nell'OOP tradizionale sono difficilmente modellabili.

itconsulting
Consulenza & Formazione

Vantagi Spring

1. Peso leggero: Il framework Spring è un framework leggero grazie alla sua implementazione del modello POJO.

2. Approccio non invasivo: Come sappiamo, i montanti costringono il programmatore a estendere la classe Action, ma il framework Spring non obbliga un programmatore a estendere la classe o implementare l'interfaccia fornita dall'API Spring.

3. Accoppiamento allentato: A causa del concetto di inserimento delle dipendenze, gli oggetti Spring sono accoppiati liberamente.

4. Moda modulare: La struttura Spring è progettata in modo modulare. Un programmatore può utilizzare solo i moduli necessari e ignorare il resto.

itconsulting
Consulenza & Formazione

Vantagi Spring

5. Test facili: L'iniezione di dipendenza e il modello POJO semplificano il test di un'applicazione.

6. Interfaccia di gestione delle transazioni: Il framework Spring fornisce un'interfaccia di gestione delle transazioni per la gestione delle transazioni.

7. Non è necessario un server delle applicazioni: L'applicazione Struts o EJB richiede l'esecuzione del server delle applicazioni, ma l'applicazione Spring non ha bisogno di un server delle applicazioni.

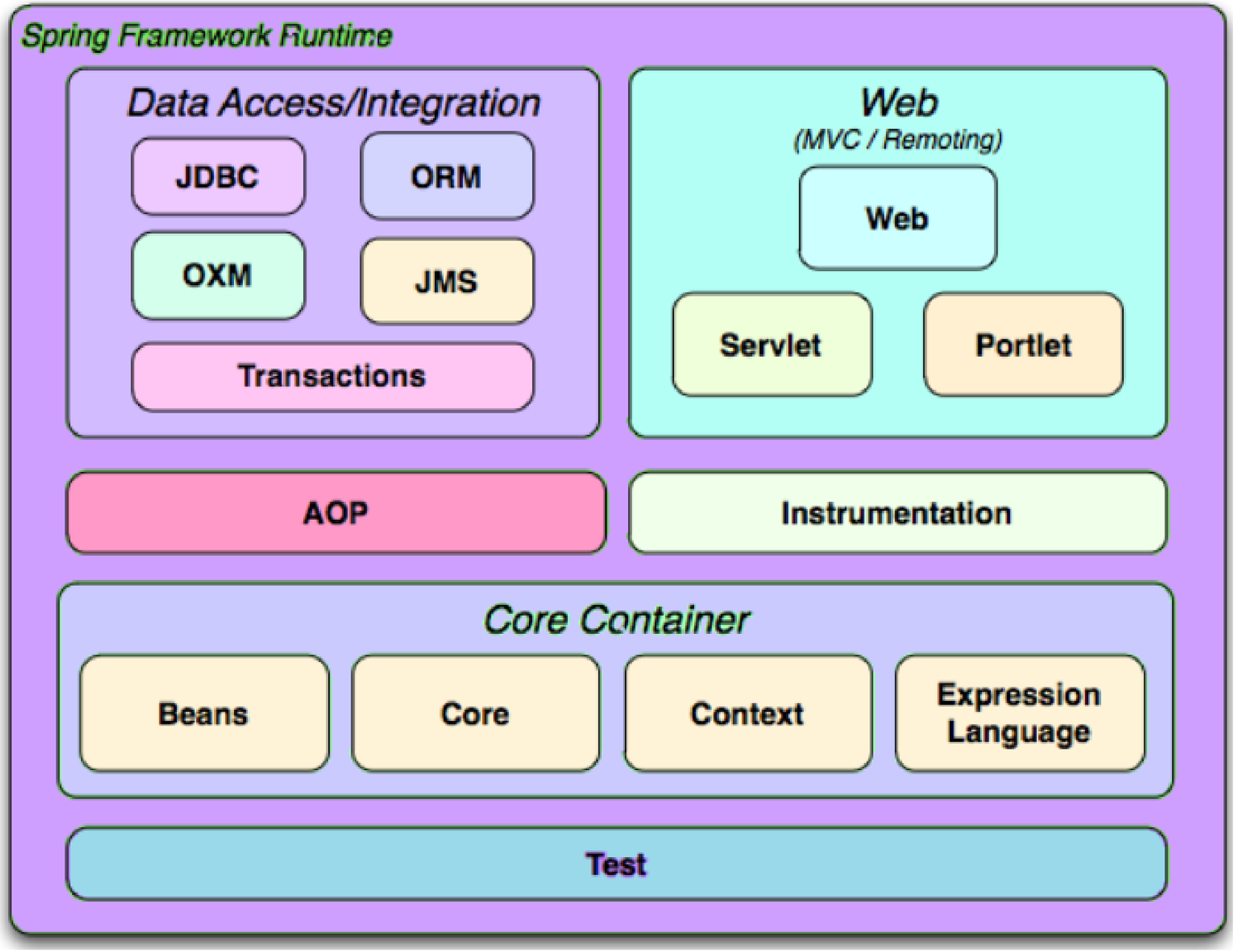
8. Struttura MVC: Il framework Spring è un'ottima alternativa ai framework Web MVC come Struts.

itconsulting
Consulenza & Formazione

Spring Framework Architecture Modules

Il framework Spring è progettato in modo modulare e dissociabile in modo da dare al programmatore la liberta di poter scegliere i moduli applicabili e ignorare il resto.

I moduli del framework Spring sono suddivisi nelle categorie indicate di seguito.



Spring Framework Architecture Modules

1. TEST: Il modulo di test di spring fornisce i supporti per il test dei componenti di spring con i framework JUnit o TestNG.

2. Container Core: Il contenitore dell'anima di spring contiene quanto segue:

- **a. Core: il modulo Core fornisce le caratteristiche fondamentali del framework Spring come IoC e DI.**
- **b. Bean: il modulo Bean fornisce BeanFactory.**
- **c. Context: il modulo Contesto fornisce un modo per accedere a qualsiasi oggetto. L'interfaccia ApplicationContext è la parte principale del modulo Context.**
- **d. Linguaggio delle espressioni: il modulo del linguaggio delle espressioni fornisce un modo per manipolare gli oggetti in fase di esecuzione.**

Spring Framework Architecture Modules

3. L'accesso/integrazione dei dati contiene quanto segue:

- **a. JDBC: i moduli JDBC forniscono un livello di astrazione JDBC.**
- **b. ORM: ORM fornisce livelli di integrazione per API di mappatura relazionale degli oggetti come JPA e Hibernate ecc**
- **c. OXM: il modulo OXM fornisce un livello di astrazione per le API di mapping Object/XML come JAXB, Castor e XMLBeans ecc**
- **d. JMS: il modulo JMS fornisce funzionalità di elaborazione dei messaggi.**
- **e. Transaction: il modulo di transazione fornisce la possibilità di gestione delle transazioni per classi come POJO ecc.**

Spring Framework Architecture Modules

4. WEB o Rete: Il modulo Web è costituito da Web, Web-Servlet, Web-Struts, Web-Socket e Web-Portlet che fornisce la possibilità di creare applicazioni Web.

5. AOP: Il modulo AOP fornisce un'implementazione di programmazione orientata agli aspetti che fornisce la possibilità di definire intercettatori di metodo.

6. TOOL o Strumentazione: Il modulo di strumentazione fornisce il supporto della strumentazione di classe e le implementazioni del caricatore di classi

Tipi di IoC Container

Il Container Spring IoC è responsabile di creare, cablare, configurare e gestire gli oggetti durante il loro ciclo di vita completo.

Utilizza metadati di configurazione per creare, configurare e gestire oggetti. I metadati di configurazione possono essere rappresentati da file xml di configurazione di Spring o da annotazioni.

Tipi di container Spring IoC:

1. **BeanFactory**
2. **ApplicationContext**

Tipi di IoC Container

BeanFactory: org.springframework.beans.factory.BeanFactory è l'interfaccia e XmlBeanFactory è una sua classe di implementazione.

Si tratta di un semplice contenitore che fornisce il supporto di base per l'iniezione di dipendenza.

La sintassi per usare la BeanFactory:

```
Resource resource = new ClassPathResource("spring configuration file");
BeanFactory beanFactory = new XmlBeanFactory(resource)
```

Tipi di IoC Container

ApplicationContext: **org.springframework.context.ApplicationContext è l'interfaccia e ClassPathXmlApplicationContext è una classe di implementazione di esso. Il contenitore ApplicationContext include tutte le funzionalità del contenitore BeanFactory con alcune funzionalità extra come l'internazionalizzazione, gli ascoltatori di eventi ecc.**

La sintassi per usare la ApplicationContext:

```
ApplicationContext applicationContext  =new
ClassPathXmlApplicationContext("spring configuration file");
```

Poiché ApplicationContext fornisce funzionalità extra, incluse tutte quelle fornite da BeanFactory, è meglio utilizzare il contenitore ApplicationContext

itconsulting
Consulenza & Formazione

Tipi di IoC Container

Uno Spring Bean rappresenta un oggetto creato, configurato e gestito da Spring Container. Un bean Spring viene creato dai metadati di configurazione passati al contenitore Spring che comunicano al contenitore la creazione del bean, il ciclo di vita del bean e le dipendenze del bean. Proprietà di uno Spring Bean:

1. **classe È obbligatorio e specifica la classe bean utilizzata per creare il bean.**
2. **nome Specifica l'identificatore univoco del bean.**
3. **scope Specifica l'ambito degli oggetti creati da una particolare definizione di bean.**
4. **costruttor-arg Viene utilizzato per iniettare le dipendenze.**
5. **proprietà Viene utilizzato per iniettare le dipendenze.**
6. **modalità di cablaggio automatico Viene utilizzato per iniettare le dipendenze.**
7. **lazy-initialization mode Serve a dire al contenitore IoC di creare un'istanza di bean quando viene richiesta per la prima volta, piuttosto che all'avvio.**
8. **metodo di inizializzazione È un metodo di callback da chiamare subito dopo che tutte le proprietà necessarie sul bean sono state impostate dal contenitore.**
9. **metodo di distruzione È un callback da richiamare quando il contenitore contenente il bean viene distrutto.**

Syntax using XML based configuration file:

```
<bean id="..." class="..." lazy-init="true">//bean configuration</bean>
```

itconsulting
Consulenza & Formazione

Spring Bean Scopes

Come abbiamo discusso, il contenitore Spring è responsabile della creazione e della gestione dell'oggetto bean. Spring offre la possibilità di restituire la stessa istanza o una nuova istanza ogni volta che ne è necessaria una. Dipende dalla portata del Bean.

Scope del bean framework Spring:

- ***1. singleton: Limita la definizione del bean a una singola istanza per contenitore Spring. È l'ambito predefinito. Il contenitore Spring lo tiene nella cache e restituisce la stessa istanza ogni volta che viene effettuata una richiesta per quel particolare bean.***
- ***2. prototype: Stabilisce l'ambito di una singola definizione di bean per avere una nuova istanza di bean ogni volta che viene effettuata una richiesta per quel particolare bean.***
- ***3. request: Applica una definizione di bean a una richiesta HTTP.***
- ***4. session: Applica una definizione di bean a una sessione HTTP.***
- ***5. global-session: Applica una definizione di bean a una sessione HTTP globale.***

Nota: la richiesta, la sessione e la sessione globale sono disponibili solo nel contesto di un ApplicationContext compatibile con il Web.

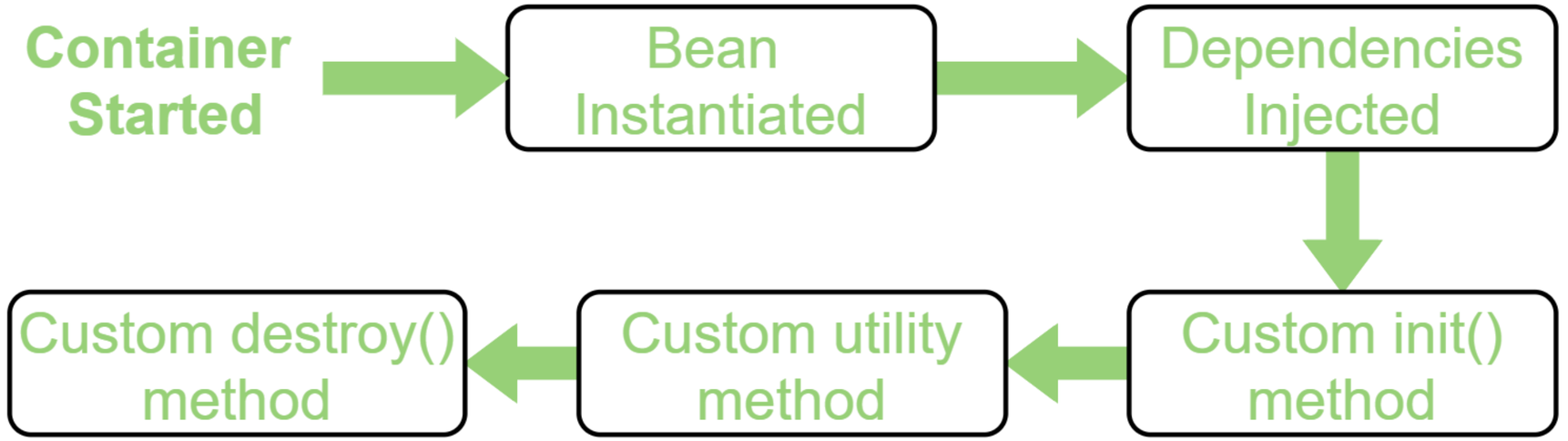
Spring bean life cycle

Come discusso in precedenza, il contenitore Spring IoC è responsabile della creazione, configurazione e gestione degli oggetti durante il loro ciclo di vita completo utilizzando i metadati di configurazione. Vedere i punti per comprendere il ciclo di vita dello spring bean.

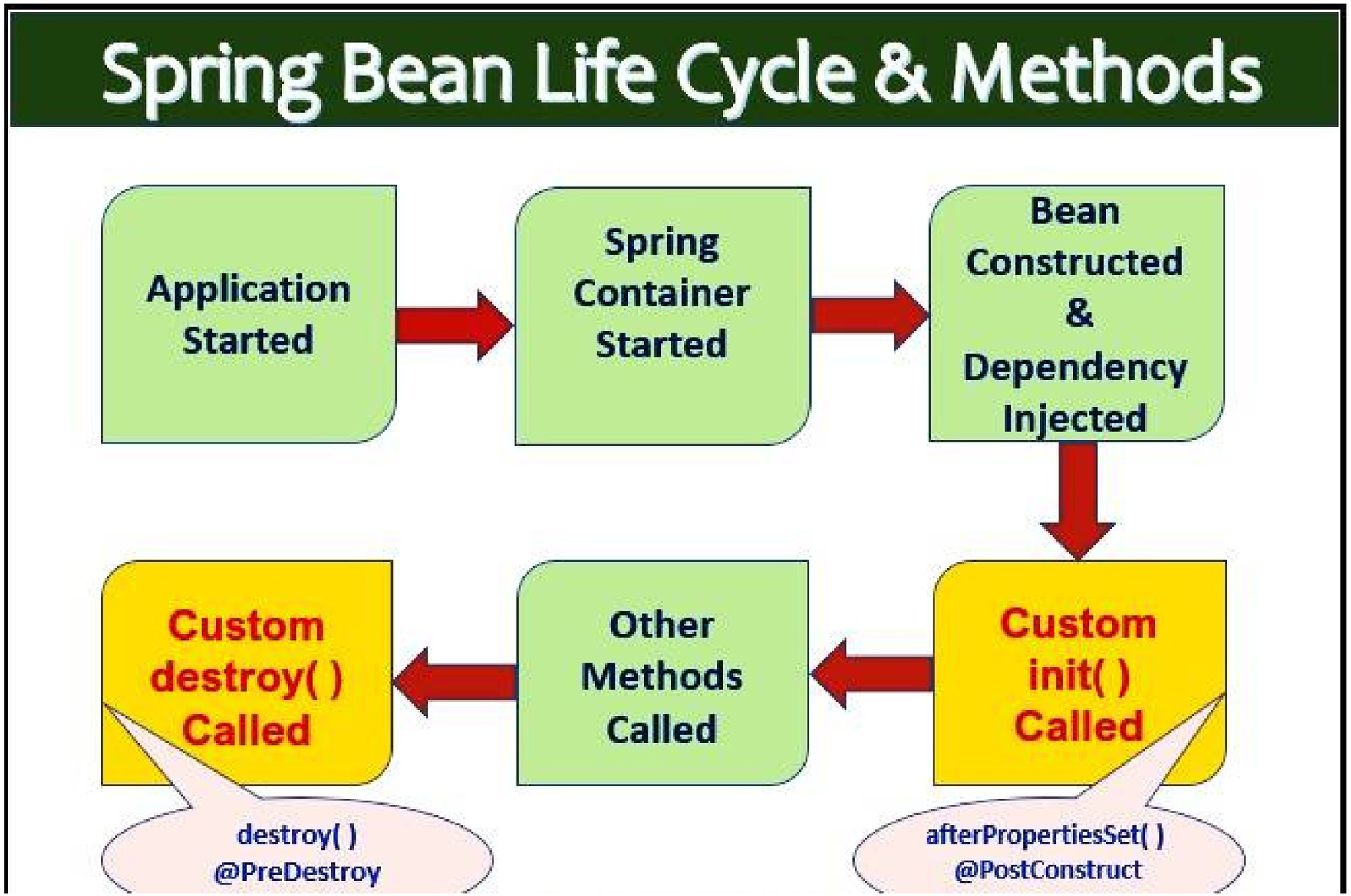
1. Il contenitore Spring trova la definizione del bean dal file di configurazione.
2. Il contenitore Spring crea un'istanza del bean utilizzando l'API Java Reflection.
3. Il contenitore Spring applica tutte le proprietà specificate utilizzando DI.
4. Se la classe bean implementa l'interfaccia BeanNameAware, allora il container spring chiama il metodo setBeanName() passando l'id del bean.
5. Se la classe bean implementa l'interfaccia BeanClassLoaderAware, il contenitore spring chiama il metodo setBeanClassLoader() passando un'istanza dell'oggetto ClassLoader che ha caricato il bean.
6. Se la classe bean implementa l'interfaccia BeanFactoryAware, il contenitore Spring chiama il metodo setBeanFactory() passando un'istanza dell'oggetto BeanFactory.
7. Se sono presenti oggetti BeanPostProcessors associati a BeanFactory, il contenitore Spring chiama il metodo postProcessBeforeInitialization() anche prima di impostare le proprietà del bean.

Spring bean life cycle

8. Se la classe bean implementa l'interfaccia InitializingBean, il contenitore spring chiama il metodo afterPropertiesSet() dopo aver impostato le proprietà del bean.
9. Se init-method è specificato nel file di configurazione per il bean, il contenitore Spring chiama il metodo corrispondente nella classe bean.
10. Se sono presenti BeanPostProcessors associati al bean, il contenitore Spring chiama il metodo postProcessAfterInitialization().
11. Se la classe bean implementa l'interfaccia DisposableBean, il contenitore spring chiama il metodo destroy() quando l'applicazione non ha più bisogno del riferimento al bean.
12. Se destroy-method è specificato nel file di configurazione per il bean, il contenitore spring chiama il metodo corrispondente nella classe del bean.



Spring bean life cycle

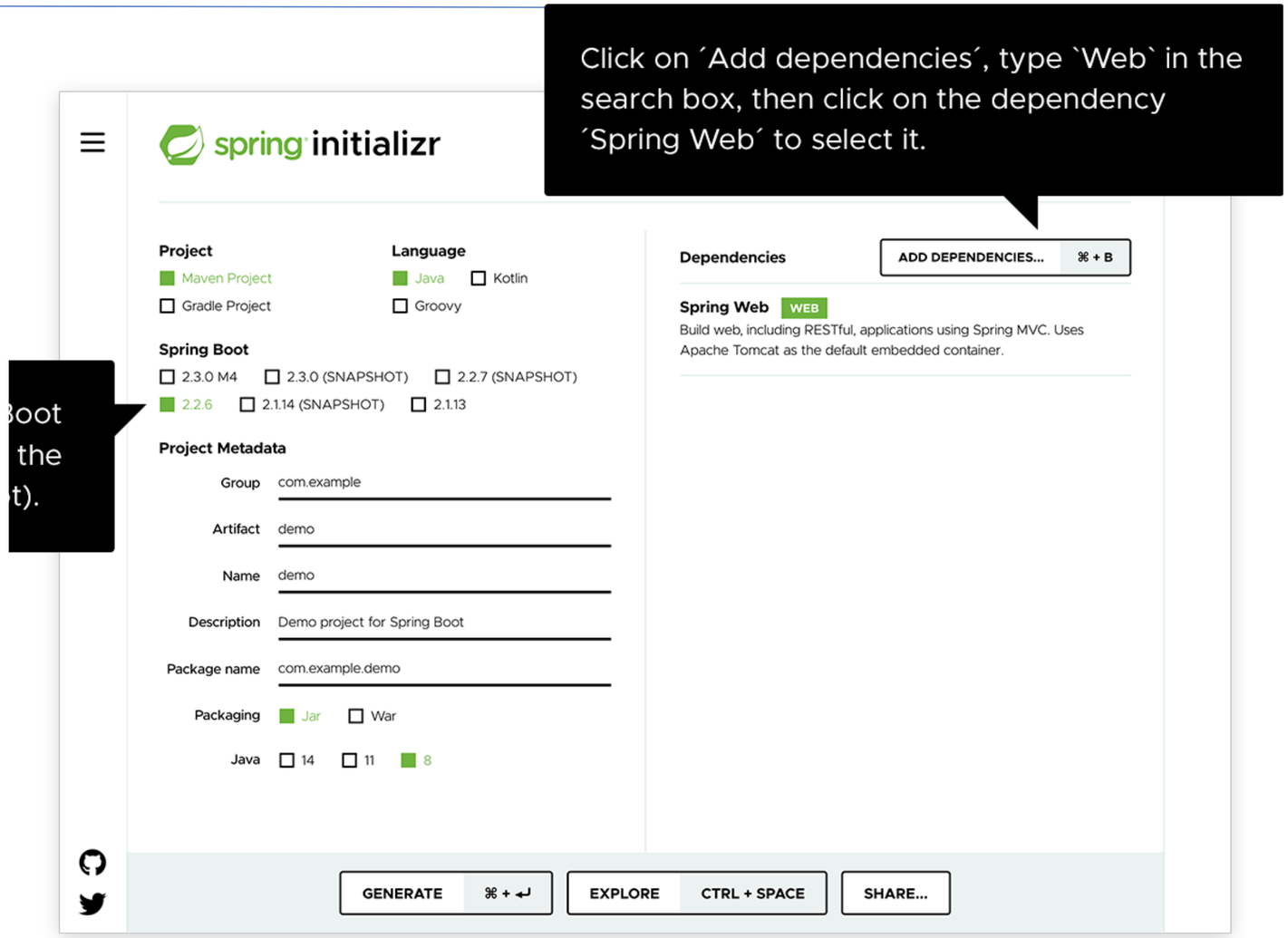


itconsulting
Consulenza & Formazione

Spring Frist step

Passaggio 1: avviare un nuovo progetto Spring Boot Utilizzo START.SPRING.io per creare un progetto “web”.

Nella finestra di dialogo "Dipendenze" cerca e aggiungi la dipendenza "web" come mostrato nello screenshot. Premi il pulsante "Genera", scarica lo zip e decomprimilo in una cartella sul tuo computer.



Passaggio 2: aggiungi il tuo codice

Apri il progetto nel tuo IDE e individua il DemoApplication.javafile nella src/main/java/com/example/democartella. Ora cambia il contenuto del file aggiungendo il metodo extra e le annotazioni mostrate nel codice qui sotto. Puoi copiare e incollare il codice o semplicemente digitarlo.

Spring Frist step

```
1. package com.example.demo;
2. import org.springframework.boot.SpringApplication;
3. import org.springframework.boot.autoconfigure.SpringBootApplication;
4. import org.springframework.web.bind.annotation.GetMapping;
5. import org.springframework.web.bind.annotation.RequestParam;
6. import org.springframework.web.bind.annotation.RestController;
7.
8. @SpringBootApplication
9. @RestController
10. public class DemoApplication {
11.     public static void main(String[] args) {
12.       SpringApplication.run(DemoApplication.class, args); }
13.     @GetMapping("/hello")
14.     public String hello(@RequestParam(value = "name", defaultValue = "World") String
      name) {
15.       return String.format("Hello %s!", name); } }
```

Il metodo hello()che abbiamo aggiunto è progettato per accettare un parametro String chiamato name e quindi combinare questo parametro con la parola "Hello"nel codice. Ciò significa che se imposti il tuo nome "Amy"nella richiesta, la risposta sarà "Ciao Amy".

L' @RestControllerannotazione dice a Spring che questo codice descrive un endpoint che dovrebbe essere reso disponibile sul web. @GetMapping(“/hello”)Dice a Spring di utilizzare il nostro metodo hello()per rispondere alle richieste che vengono inviate http://localhost:8080/helloall'indirizzo. Infine, @RequestParamsta dicendo a Spring di aspettarsi un valore di nome nella richiesta, ma se non è presente, utilizzerà la parola "Word"per impostazione predefinita.

Passaggio 3: provalo
Costruiamo ed eseguiamo il programma. Apri una riga di comando (o terminale) e vai alla cartella in cui hai i file di progetto. Possiamo costruire ed eseguire l'applicazione emettendo il seguente comando:

MacOS/Linux:
COPY./gradlew spring-boot:run

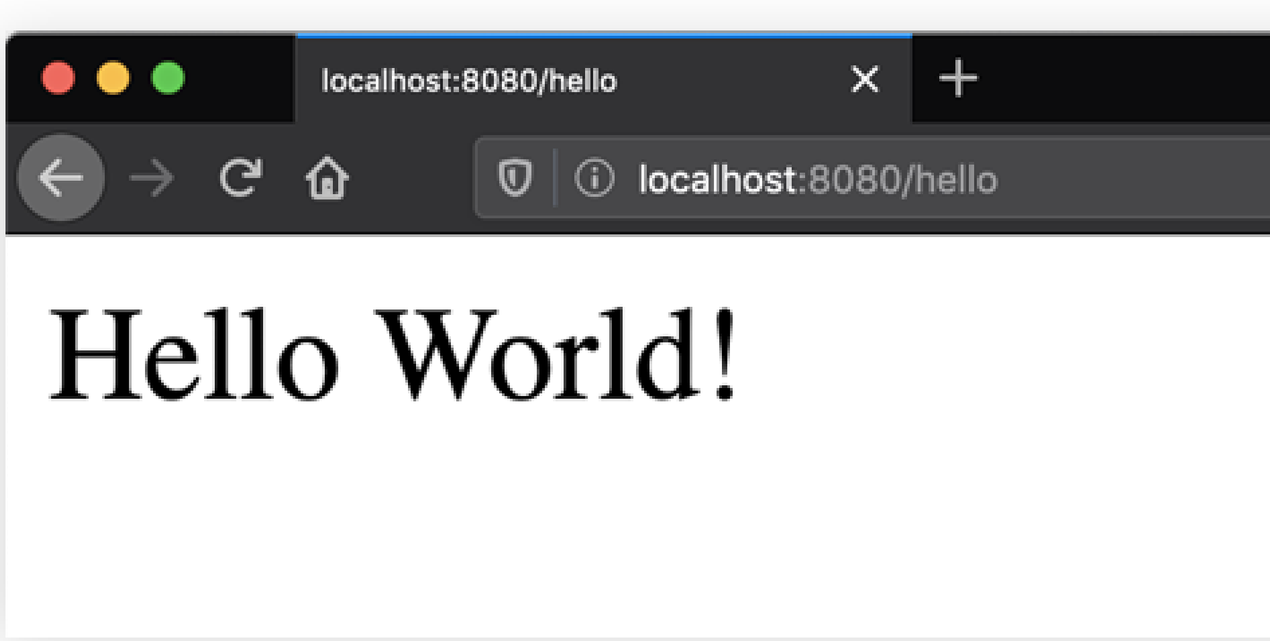
WINDOW:
COPYgradlew.bat spring-boot:run

Le ultime due righe qui ci dicono che la primavera è iniziata. Il server Apache Tomcat integrato di Spring Boot funge da server Web e ascolta le richieste sulla localhost port 8080.

Apri il tuo browser e nella barra degli indirizzi in alto inserisci

http://localhost:8080/hello.

Dovresti ottenere una bella risposta amichevole come questa:



itconsulting
Consulenza & Formazione

Spring Callback Methods

Un metodo di callback in java è un metodo che viene chiamato quando si verifica un evento. Normalmente possiamo implementarlo passando un'implementazione di una certa interfaccia al sistema che è responsabile dell'attivazione dell'evento. Metodi di callback post-inizializzazione:

InitializingBean:

- **L'interfaccia InitializingBean fornisce il metodo afterPropertiesSet() che può essere utilizzato per qualsiasi attività post-inizializzazione. Sintassi:**

```
public class TestBean implements InitializingBean {
   public void afterPropertiesSet() {
      // post-initialization task.
   } }
```

itconsulting
Consulenza & Formazione

Spring Callback Methods

Usando l'attributo init-method:
Nei metadati di configurazione XML possiamo specificare il nome del metodo che ha una firma void senza argomento nell'attributo init-method per qualsiasi attività post-inizializzazione. Sintassi:

```
<bean id="testBean" class="Test" init-method="init"/>
In class definition:
public class Test {
   public void init() {
      // post-initialization task.
   }
}
```

itconsulting
Consulenza & Formazione

Spring Callback Methods

Utilizzando l'interfaccia DisposableBean:

L'interfaccia DisposableBean fornisce il metodo destroy() che può essere utilizzato per qualsiasi attività pre-distruzione. Sintassi:

```
public  class TestBean implements DisposableBean {
  public  void destroy ( )  {
    // pre-destroy task.   } }
```

Usando l'attributo destroy-method:

Nei metadati di configurazione XML possiamo specificare il nome del metodo che ha una firma void senza argomento nell'attributo destroy-method per qualsiasi attività pre-destroy. Sintassi:

```
< bean id = "testBean"  class = "Test" destroy - method = " destroy" />
Nella definizione della classe :
public  class Test {
  public  void destroy ( )  {
    // pre-destroy task. } }
```

itconsulting
Consulenza & Formazione

Spring Hello World Example In Eclipse

1. Scarica i file jar spring e aggiungili al percorso di classe del progetto. Oppure usa, guarda l'esempio Maven Eclipse Spring .

2. Creare una classe java (bean).

3. Creare un file di configurazione di spring che contenga tutte le configurazioni dei bean.

4. Crea una classe di prova spring.

5. Caricare il file di configurazione dello spring nell'oggetto Resource Interface.
Sintassi: Resource resource = new ClassPathResource(“spring configuration file”);

6. Creare un oggetto BeanFactory (contenitore Spring IOC) leggendo il file di configurazione Spring tramite l'oggetto risorsa.
Sintassi: BeanFactory beanFactory = new XmlBeanFactory(risorsa);

7. Ottieni l'oggetto bean da beanFactory (contenitore spring ioc) passando beanId.
TestBean testBean = beanFactory.getBean(“testBeanId");

Spring Hello World Example In Eclipse

Possiamo anche utilizzare ApplicationContext per ottenere l'oggetto bean
Sintassi:

```
ApplicationContext context = new ClassPathXmlApplicationContext("spring configuration file");
TestBean testBean = (TestBean) context.getBean("testBeanId");
```

2. Il contenitore Spring IOC è responsabile della creazione di oggetti bean e dell'iniezione di dipendenza.

3. Spring utilizza il modello di progettazione singleton per i bean, quindi ogni bean Spring è una classe singleton per impostazione predefinita.

itconsulting
Consulenza & Formazione

Spring Hello World Example In Eclipse

Per prima cosa crea una classe di bean java chiamata "HelloWorld.java" che ha una proprietà userName e un metodo sayHello. Quindi crea un file di configurazione di primavera che definisce il bean e le sue proprietà.

Qui definiamo il bean HelloWorld.java con id helloWorld. Questo ID verrà utilizzato in seguito per ottenere l'oggetto bean dall'oggetto ApplicationContext.

Quindi creare la classe Test.java come programma principale. Framework crea l'oggetto ApplicationContext utilizzando il file di configurazione di Spring. Quindi ottenere l'oggetto bean HelloWorld da ApplicationContext utilizzando il metodo getBean(). Dobbiamo passare beanId nel metodo getBean(). Al momento della creazione dell'oggetto, il contenitore IoC imposta il valore delle proprietà del bean definito nel file di configurazione di Spring, ad esempio il valore della proprietà userName sarà impostato su jai.

Spring Hello World Example In Eclipse

CiaoMondo.java

```
package com.w3spoint.business;

public  class HelloWorld {
  private  String userName ;
  public  void setUserName ( String userName )  {
     this . userName = userName ; }

  public  void sayHello ( ) {
     System . out . println ( "Ciao: "  + userName ) ;  } }
```

itconsulting
Consulenza & Formazione

Spring Hello World Example In Eclipse

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<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="helloWorld" class="com.w3spoint.business.HelloWorld">
      <property name="userName" value="jai"/>
   </bean>

</beans>
```

Spring Hello World Example In Eclipse

Test.java

```
package com.w3spoint.business;

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

public class Test {
 public static void main(String[] args) {
  ApplicationContext context =
  new ClassPathXmlApplicationContext("applicationContext.xml");
  HelloWorld helloWorld = (HelloWorld) context.getBean("helloWorld");

  helloWorld.sayHello();
 }
}
```

itconsulting
Consulenza & Formazione

Ereditarietà Della Definizione Dello spring bean

Come abbiamo discusso in precedenza, una definizione di bean nei metadati di configurazione può contenere argomenti del costruttore, valori di proprietà, ecc. Il framework Spring fornisce la possibilità che una definizione di bean figlio possa ereditare i dati di configurazione da una definizione genitore.

Una definizione figlio può sovrascrivere alcuni valori della definizione padre o aggiungerne altri, come richiesto. Usa l'attributo genitore nella definizione figlio e passaci il bean genitore. Sintassi:

```
< bean id = "parentBeanId"  class = "TestParentBean" >
    < nome proprietà = "name1" value = "value1" />
    < nome proprietà = "name2" value = "value2" />
</ bean >

< bean id = "childBeanId"  class = "ChildBeanId" parent = "parentBeanId" >
    < nome proprietà = "name1" valore = "value" />
    < nome proprietà = "name3" valore = "value3" />
</ bean >
```

itconsulting
Consulenza & Formazione

Ereditarietà Della Definizione Dello spring bean

Esempio Spiegazione: Andiamo a creare due bean "HelloWorld" e "HelloJava". HelloWorld.java ha due proprietà msg1, msg2 e HelloJava.java ha tre proprietà msg1, msg2, msg3. In applicationContext.xml bean HelloWorld definito con proprietà msg1 e msg2. Il bean HelloJava usa il bean HelloWorld come genitore ed eredita le proprietà msg1 e msg2. Usa lo stesso valore per la proprietà msg1 come definito in HelloWorld e sovrascrive la proprietà msg2. Aggiunge anche un'altra proprietà msg3.

CiaoMondo.java

```
pacchetto  com.w3spoint.business ;
public  class HelloWorld {
   private String msg1 ;
   string private  msg2 ;
 public  String getMsg1 ( )  { return msg1 ; }
 public  void setMsg1 ( String msg1 )  { this . msg1  = msg1 ; }
 public  String getMsg2 ( )  { return msg2 ; }
 public  void setMsg2 ( String msg2 )  { this . msg2  = msg2 ; }  }
```

itconsulting
Consulenza & Formazione

Ereditarietà Della Definizione Dello spring bean

HelloJava.java

```
pacchetto  com.w3spoint.business ;

public  class HelloJava {
 private String msg1 ;
 string private  msg2 ; string private msg3 ;

 public  String getMsg1 ( )  { return msg1 ; }
 public  void setMsg1 ( String msg1 )  { this . msg1  = msg1 ;  }
 public  String getMsg2 ( )  { return msg2 ; }
 public  void setMsg2 ( String msg2 )  { this . msg2  = msg2 ; }
 string public getMsg3 ( )  {  return msg3 ; }
 public  void setMsg3 ( String msg3 )  {  this . msg3  = msg3 ; }
}
```

itconsulting
Consulenza & Formazione

Ereditarietà Della Definizione Dello spring bean

applicationContext.java

```
<?xml version="1.0" encoding="UTF-8"?>
<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="helloWorld" class="com.w3spoint.business.HelloWorld">
     <property name="msg1" value="Java World"/>
     <property name="msg2" value="World."/>
  </bean>
   <bean id="helloJava"
        class="com.w3spoint.business.HelloJava" parent="helloWorld">
     <property name="msg2" value="Java"/>
     <property name="msg3" value="Java."/>
   </bean>
</beans>
```

itconsulting
Consulenza & Formazione

Ereditarietà Della Definizione Dello spring bean

Test.java

```
package com.w3spoint.business;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class Test {
 public static void main(String[] args) {
  ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
  HelloWorld helloWorld = (HelloWorld) context.getBean("helloWorld");
  System.out.println("HelloWorld bean properties: ");
  System.out.println("Hello " + helloWorld.getMsg1());
  System.out.println("Hello " + helloWorld.getMsg2());

  HelloJava helloJava = (HelloJava) context.getBean("helloJava");
  System.out.println("HelloJava bean properties: ");
  System.out.println("Hello " + helloJava.getMsg1());
  System.out.println("Hello " + helloJava.getMsg2());
  System.out.println("Hello " + helloJava.getMsg3());   } }
```

itconsulting
Consulenza & Formazione

JDBC (Java DataBase Connectivity

In informatica JDBC, è un connettore e un driver per database che consente l'accesso e la gestione della persistenza dei dati sulle basi di dati da qualsiasi programma scritto con il linguaggio di programmazione Java, indipendentemente dal tipo di DBMS utilizzato. È costituito da un'API object oriented orientata ai database relazionali, raggruppata nel package java.sql, che serve ai client per connettersi a un database fornendo i metodi per interrogare e modificare i dati.

La piattaforma Java 2 Standard Edition contiene le API JDBC, insieme all'implementazione di un bridge JDBC-ODBC, che permette di connettersi a database relazionali che supportino ODBC, che è in codice nativo e non in Java. Tipicamente ciascun DB ha il suo specifico driver JDBC per interfacciarsi con l'applicazione. Spesso i framework di persistenza in ambito Java (es. Hibernate) nella loro implementazione a più alto livello si interfacciano a più basso livello proprio con uno strato software JDBC.

JDBC (Java DataBase Connectivity

L'architettura di JDBC, così come quella di ODBC, prevede l'utilizzo di un "driver manager", che espone alle applicazioni un insieme di interfacce standard e si occupa di caricare a "run-time" i driver opportuni per "pilotare" gli specifici DBMS. Le applicazioni Java utilizzano le "JDBC API" per parlare con il JDBC driver manager, mentre il driver manager usa le JDBC driver API per parlare con i singoli driver che pilotano i DBMS specifici. Esiste un driver particolare, il "JDBC-ODBC Bridge", che consente di interfacciarsi con qualsiasi driver ODBC in ambiente Windows.

Esistono driver free e commerciali per la maggior parte dei server di database relazionali. I driver possono essere di quattro tipi:

- **il JDBC-ODBC Bridge, API nativa, protocollo di rete, protocollo nativo**

JDBC (Java DataBase Connectivity

Prima di vedere le loro differenze cerchiamo prima di capire cosa sono JDBC e Hibernate. JDBC(Java Database Connectivity) è un API usata per accedere ai database in un programma Java. In parole povere è uno strumento per la connessione java-database. Ci sono vari metodi e query forniti da JDBC per l'accesso ai database. I driver per creare connessioni sono forniti da JDBC.

Hibernate è uno strumento ORM usato per mappare dalle tabelle di un database alle classi java. Rende facile implementare la logica OOP di Java nelle tabelle. Hibernate crea la connessione da solo e per operare sulle tabelle usa il suo linguaggio query. Con JDBC usiamo il linguaggio SQL, Hibernate usa un linguaggio proprietario conosciuto come Hql.

itconsulting
Consulenza & Formazione

JDBC (Java DataBase Connectivity

In informatica JDBC, è un connettore e un driver per database che consente l'accesso e la gestione della persistenza dei dati sulle basi di dati da qualsiasi programma scritto con il linguaggio di programmazione Java, indipendentemente dal tipo di DBMS utilizzato. È costituito da un'API object oriented orientata ai database relazionali, raggruppata nel package java.sql, che serve ai client per connettersi a un database fornendo i metodi per interrogare e modificare i dati.

La piattaforma Java 2 Standard Edition contiene le API JDBC, insieme all'implementazione di un bridge JDBC-ODBC, che permette di connettersi a database relazionali che supportino ODBC, che è in codice nativo e non in Java. Tipicamente ciascun DB ha il suo specifico driver JDBC per interfacciarsi con l'applicazione. Spesso i framework di persistenza in ambito Java (es. Hibernate) nella loro implementazione a più alto livello si interfacciano a più basso livello proprio con uno strato software JDBC.

JDBC (Java DataBase Connectivity

n JDBC i driver vengono usati per aprire e chiudere la connessione, in Hibernate la session è utilizzata per lo stesso motivo, ma le operazioni vengono eseguite usando oggetti persistenti (classi sulle quali sono mappate le tabelle). La Session è fornita dal sessionFactory che è un'interfaccia. Per una connessione al database viene mantenuta una sessionFactory.

Mentre in JDBC dovevamo scrivere ogni linea di codice ogni volta per mappare ogni colonna di ogni tabella, in Hibernate la cosa è gestita in maniera molto semplice da Hibernate stesso. Nella configurazione forniamo un file di configurazione e il resto è gestito da Hibernate. L'hql fornito da Hibernate fa uso di vari concetti di oggetti come eredità ecc.

Hibernate

In informatica Hibernate (talvolta abbreviato in H8) è una piattaforma middleware open source per lo sviluppo di applicazioni Java, attraverso l'appoggio al relativo framework, che fornisce un servizio di Object-relational mapping (ORM) ovvero gestisce la persistenza dei dati sul database attraverso la rappresentazione e il mantenimento su database relazionale di un sistema di oggetti Java.

Come tale dunque, nell'ambito dello sviluppo di applicazioni web, tale strato software si frappone tra il livello logico di business o di elaborazione e quello di persistenza dei dati sul database (Data Access Layer).
È stato originariamente sviluppato da un team internazionale di programmatori volontari coordinati da Gavin King; in seguito il progetto è stato proseguito sotto l'egida di JBoss, che ne ha curato la standardizzazione rispetto alle specifiche Java EE.

Hibernate

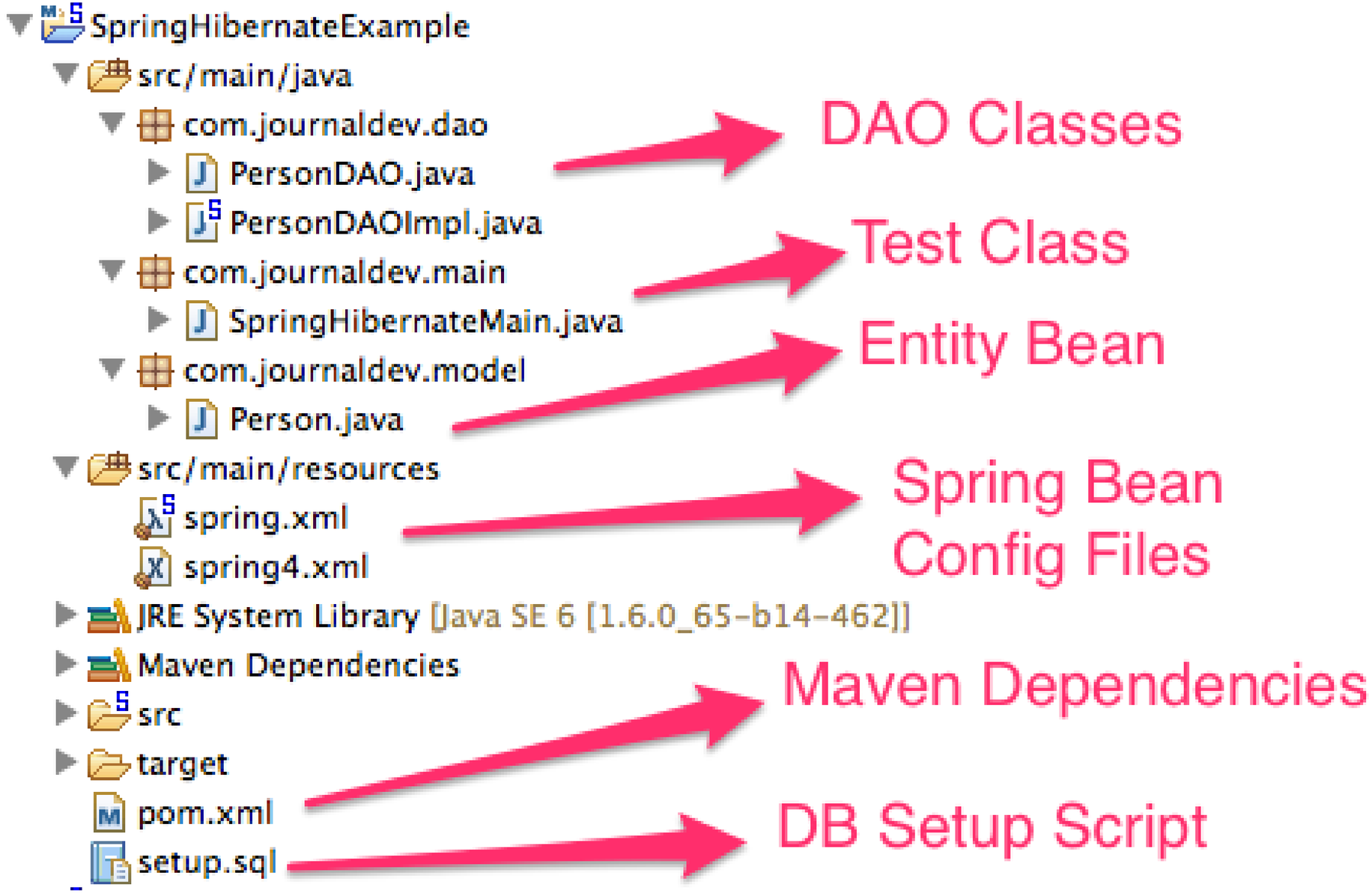
Hibernate è distribuito in licenza LGPL sotto forma di librerie software da linkare nel progetto di sviluppo software. Lo scopo principale di Hibernate è quello di fornire un mapping delle classi Java in tabelle di un database relazionale; sulla base di questo mapping Hibernate gestisce il salvataggio degli oggetti di tali classi su database (tipicamente attributi di oggetti per ciascun campo dati della tabella). Si occupa inoltre al rovescio del reperimento degli oggetti dal database, producendo ed eseguendo automaticamente le query SQL necessarie al recupero delle informazioni e la successiva reistanziazione dell'oggetto precedentemente "ibernato" (mappato su database).

L'obiettivo di Hibernate è quello di esonerare lo sviluppatore dall'intero lavoro relativo alla persistenza dei dati. Hibernate si adatta al processo di sviluppo del programmatore, sia se si parte da zero sia se da un database già esistente. Hibernate genera le chiamate SQL e solleva lo sviluppatore dal lavoro di recupero manuale dei dati e dalla loro conversione, mantenendo l'applicazione portabile in tutti i database SQL. Hibernate fornisce una persistenza trasparente per Plain Old Java Object (POJO); l'unica grossa richiesta per la persistenza di una classe è la presenza di un costruttore senza argomenti. In alcuni casi si richiede un'attenzione speciale per i metodi equals() e hashcode().

Hibernate

Hibernate è tipicamente usato sia in applicazioni Swing che Java EE facenti uso di servlet o EJB di tipo session beans.
La versione 3 di Hibernate arricchisce la piattaforma con nuove caratteristiche come una nuova architettura Interceptor/Callback, filtri definiti dall'utente, e annotazione stile JDK 5.0 (Java's metadata feature).
Hibernate 3 è vicino anche alle specifiche di EJB 3.0 (nonostante sia stato terminato prima di EJB 3.0 le specifiche erano già state pubblicate dalla Java Community Process) ed è usato come spina dorsale per l'implementazione EJB 3.0 di JBoss.
Nel dicembre 2011 è uscita la versione 4.0, e a gennaio 2012 la versione 4.01. Nel mese di agosto 2013 è stata resa disponibile la versione 4.2.4.

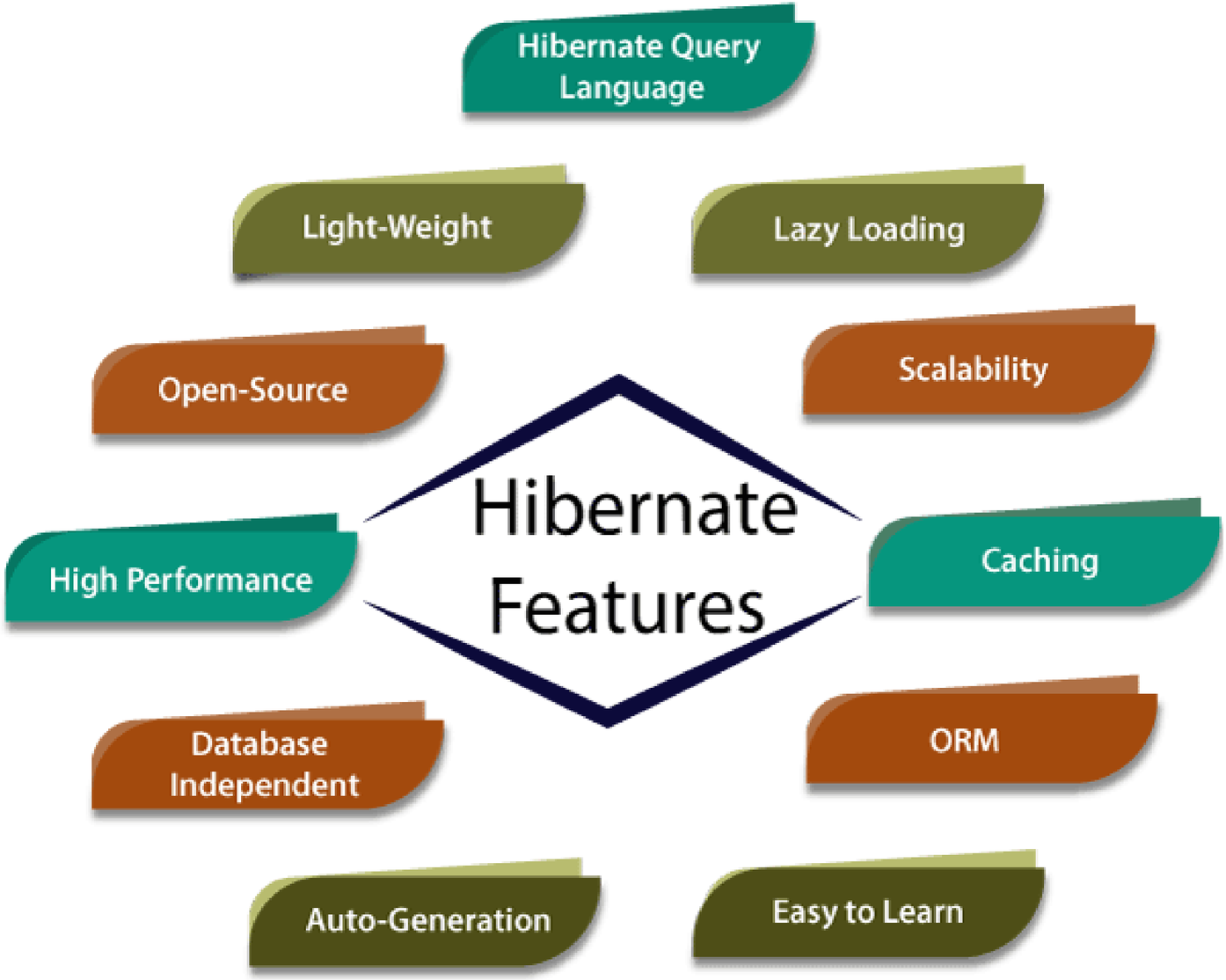
Hibernate


SpringHibernateExample
src/main/java
com.journaldev.dao
PersonDAO.java
PersonDAOImpl.java
com.journaldev.main
SpringHibernateMain.java
com.journaldev.model
Person.java
src/main/resources
spring.xml
spring4.xml
JRE System Library [Java SE 6 [1.6.0_65-b14-462]]
Maven Dependencies
src
target
pom.xml
setup.sql
DAO Classes
Test Class
Entity Bean
Spring Bean
Config Files
Maven Dependencies
DB Setup Script

– la classe corrisponde agli elementi tipizati tutoria Tutorial di entità e tabelle, è un'interfaccia che estende JpaRepository per metodi CRUD e metodi di ricerca personalizzati. Sarà cablato automaticamente in . – è un controller che dispone di metodi di mappatura delle richieste per le richieste RESTful come: getAll , addTutorial,saveTutorial,editTutorial,deleteTutorial,updateTutorialPublishedStatus .
TutorialRepositoryTutorialController, TutorialController

– static/css contiene uno stile css personalizzato.
– templatememorizza i file modello HTML per il progetto.
– Configurazione per Spring Datasource, JPA e Hibernate in application.properties .
– pom.xml contiene dipendenze per Spring Boot, Thymeleaf, Bootstrap e Database.

Hibernate


Hibernate Query Language
Light-Weight
Lazy Loading
Open-Source
Scalability
Hibernate Features
High Performance
Caching
Database Independent
ORM
Auto-Generation
Easy to Learn

Esempio Crud

Potete ritrovare Vari esempi su

https://www.bezkoder.com/spring-boot-thymeleaf-example/

https://www.digitalocean.com/community/tutorials/spring-hibernate-integration-example-tutorial

https://www.baeldung.com/spring-data-crud-repository-save