Aktuell zu Spring 5.0

craig WALLS

SPRING IM EINSATZ

3. Auflage

HANSER

1.1 Was ist Spring?

Mir ist klar, dass Sie wahrscheinlich darauf brennen, eine Spring-Anwendung zu schreiben, und ich versichere Ihnen, dass Sie noch in diesem Kapitel eine einfache Anwendung entwickeln werden. Doch zuerst möchte ich mit einigen grundlegenden Spring-Konzepten den Weg bereiten, damit Sie besser verstehen, was Spring am Laufen hält.

Jede nichttriviale Anwendung besteht aus vielen Komponenten, die jeweils für ihren eigenen Teil der gesamten Anwendungsfunktionalität verantwortlich sind. Dabei stimmen sie sich mit den anderen Anwendungselementen ab, um ihre Aufgaben zu erledigen. Wenn die Anwendung läuft, müssen diese Komponenten irgendwie erzeugt und einander bekannt gemacht werden.

In seinem Kern (Core) bietet Spring einen *Container*, oftmals auch als *Spring-Anwendungskontext* bezeichnet, der Anwendungskomponenten erzeugt und verwaltet. Diese Komponenten oder *Beans* werden im Spring-Anwendungskontext miteinander zu einer vollständigen Anwendung verknüpft, ähnlich wie aus Bausteinen, Mörtel, Holz, Nägeln, Rohrleitungen und Kabeln ein Haus entsteht.

Das Verknüpfen der Beans untereinander beruht auf einem Muster, das als *Dependency Injection* (DI) bekannt ist. Anstatt Komponenten den Lebenszyklus anderer Beans, von denen sie abhängig sind, erzeugen und verwalten zu lassen, stützt sich eine DI-Anwendung auf eine separate Entität (den Container), um alle Komponenten zu erzeugen und zu verwalten und diese in die Beans, die auf sie angewiesen sind, zu injizieren. In der Regel geschieht dies über Konstruktorargumente oder die Zugriffsmethoden von Eigenschaften.

Nehmen wir zum Beispiel an, dass es unter den vielen Komponenten einer Anwendung zwei gibt, um die es Ihnen geht: einen Inventurdienst (mit dem sich Lagerbestände abrufen lassen) und einen Produktdienst (der die grundlegenden Produktdaten bereitstellt). Der Produktdienst hängt vom Inventurdienst ab, um vollständige Informationen über Produkte liefern zu können. Bild 1.1 veranschaulicht die Beziehungen zwischen diesen Beans und dem Spring-Anwendungskontext.

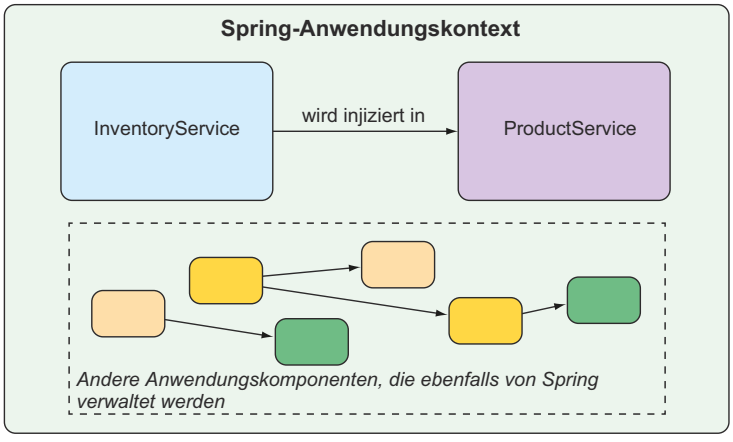

Bild 1.1 Anwendungskomponenten werden durch den Spring-Anwendungskontext verwaltet und ineinander injiziert.

Aufbauend auf dem Core-Container von Spring und einem kompletten Portfolio verwandter Bibliotheken bekommen Sie ein Web-Framework, eine Vielzahl von Optionen für dauerhafte Datenspeicherung, ein Sicherheits-Framework, Integration mit anderen Systemen, Laufzeitüberwachung, Microservice-Unterstützung, ein reaktives Programmiermodell und viele andere Features, die für die moderne Anwendungsentwicklung notwendig sind.

Aus historischer Sicht würde man den Anwendungskontext von Spring dazu bringen, Beans miteinander zu verbinden, indem man mit einer oder mehreren XML-Dateien die Komponenten und ihre Beziehungen zu anderen Komponenten beschreibt. So deklariert das folgende XML zwei Beans, eine `InventoryService`-Bean und eine `ProductService`-Bean, und verknüpft die `InventoryService`-Bean über ein Konstruktorargument mit der `ProductService`-Bean:

```
<bean id="inventoryService"
      class="com.example.InventoryService" />

<bean id="productService"
      class="com.example.ProductService" />
  <constructor-arg ref="inventoryService" />
</bean>
```

In neueren Versionen von Spring ist aber eine Java-basierte Konfiguration gebräuchlicher. Die folgende Java-basierte Konfigurationsklasse ist der XML-Konfiguration äquivalent:

```
@Configuration
public class ServiceConfiguration {
  @Bean
  public InventoryService inventoryService() {
    return new InventoryService();
  }

  @Bean
  public ProductService productService() {
    return new ProductService(inventoryService());
  }
}
```

Die Annotation `@Configuration` teilt Spring mit, dass es sich um eine Konfigurationsklasse handelt, die dem Spring-Anwendungskontext Beans bereitstellt. Die Methoden der Konfigurationsklasse sind mit `@Bean` annotiert. Dies zeigt an, dass die von ihnen zurückgegebenen Objekte als Beans in den Anwendungskontext eingefügt werden sollen (wobei ihre jeweiligen Bean-IDs standardmäßig die gleichen sind wie die Namen der Methoden, die sie definieren).

Die Java-basierte Konfiguration bietet gegenüber der XML-basierten Konfiguration mehrere Vorteile, darunter größere Typsicherheit und verbesserte Möglichkeiten zur Refaktorisierung. Selbst dann ist eine explizite Konfiguration mit Java oder XML nur dann notwendig, wenn Spring nicht in der Lage ist, die Komponenten automatisch zu konfigurieren.

Die automatische Konfiguration geht auf die Spring-Techniken *Autowiring* und *Komponentensuche* (Component Scanning) zurück. Mit einer Komponentensuche kann Spring automatisch Komponenten im Klassenpfad einer Anwendung erkennen und sie als Beans

im Spring-Anwendungskontext erzeugen. Beim Autowiring injiziert Spring automatisch die Komponenten zusammen mit den anderen Beans, von denen sie abhängen.

In jüngster Zeit geht die automatische Konfiguration mit Einführung von Spring Boot weit über Komponentensuche und Autowiring hinaus. Spring Boot ist eine Erweiterung des Spring Frameworks, die mehrere Produktivitätsverbesserungen bietet. Die bekannteste dieser Verbesserungen ist die *Autokonfiguration*. Spring Boot kann dabei anhand von Einträgen im Klassenpfad, in den Umgebungsvariablen und anderen Faktoren vernünftige Schätzungen abgeben, welche Komponenten konfiguriert und miteinander verbunden werden müssen.

Die Autokonfiguration würde ich Ihnen gern anhand von Beispielcode demonstrieren, doch leider kann ich es nicht. Denn die Autokonfiguration ist wie der Wind. Man kann zwar ihre Wirkungen sehen, doch es gibt keinen Code, den ich Ihnen zeigen und sagen könnte: »Sehen Sie! Hier ist ein Beispiel für die Autokonfiguration!« Es passiert etwas, Komponenten werden aktiviert und Funktionalität wird bereitgestellt, ohne dass Sie Code schreiben müssen. Gerade dieser fehlende Code ist wichtig für die Autokonfiguration und das macht ihre Vorzüge aus.

Die Spring-Boot-Autokonfiguration hat die Anzahl der expliziten Konfigurationen (ob mit XML oder Java), die für das Erstellen einer Anwendung erforderlich sind, drastisch reduziert. Und wenn Sie mit dem Beispiel in diesem Kapitel fertig sind, haben Sie tatsächlich eine funktionsfähige Spring-Anwendung vor sich, die nur eine einzige Zeile Spring-Konfigurationscode enthält!

Mit Spring Boot geht die Spring-Entwicklung so viel leichter von der Hand, dass es schwer vorstellbar ist, Spring-Anwendungen ohne Spring Boot zu entwickeln. Aus diesem Grund behandelt dieses Buch Spring und Spring Boot so, als wären sie ein und dasselbe. Wir werden so weit wie möglich auf Spring Boot setzen und eine explizite Konfiguration nur verwenden, wenn es wirklich notwendig ist. Und da die Spring-XML-Konfiguration die Old-School-Methode für das Arbeiten mit Spring ist, konzentrieren wir uns hauptsächlich auf die Java-basierte Konfiguration von Spring.

Doch genug der Vorrede. Im Titel dieses Buches ist die Phrase *im Einsatz* enthalten. Fangen Sie also an und schreiben Sie Ihre erste Anwendung mit Spring.

1.2 Eine Spring-Anwendung initialisieren

Anhand dieses Buches erstellen Sie die Online-Anwendung Taco Cloud. Kunden können damit die wundervollsten Speisen bestellen, die je von Menschen kreiert wurden – Tacos. Selbstverständlich verwenden Sie Spring, Spring Boot und eine Vielzahl verwandter Bibliotheken und Frameworks, um dieses Ziel zu erreichen.

Für die Initialisierung einer Spring-Anwendung stehen Ihnen mehrere Optionen zur Verfügung. Zwar könnte ich Ihnen die Schritte erläutern, wie Sie eine Projektverzeichnisstruktur manuell einrichten und eine Build-Spezifikation definieren, doch wäre das Zeitverschwendung – die Zeit lässt sich besser für den Anwendungscode nutzen. Deshalb werden Sie sich auf den Spring Initializr stützen, um die Anwendung hochzufahren.

Der Spring Initializr ist sowohl eine browserbasierte Webanwendung als auch eine REST-API. Damit erzeugen Sie eine Spring-Projektgerüststruktur, die Sie mit jeder gewünschten Funktionalität ausfüllen können. Es gibt verschiedene Wege, um auf Spring Initializr zuzugreifen:

- von der Webanwendung unter http://start.spring.io,
- von der Befehlszeile mit dem Befehl `curl`,
- von der Befehlszeile mit der Befehlszeilenoberfläche von Spring Boot,
- beim Erstellen eines neuen Projekts mit der Spring Tool Suite,
- beim Erstellen eines neuen Projekts mit IntelliJ IDEA,
- beim Erstellen eines neuen Projekts mit NetBeans.

Anstatt mehrere Seiten dieses Kapitels zu opfern, um jede dieser Optionen zu besprechen, habe ich diese Details im Anhang zusammengestellt. In diesem Kapitel und im gesamten Buch zeige ich Ihnen, wie Sie ein neues Projekt mit meiner Lieblingsoption erzeugen: mit der Unterstützung von Spring Initializr in der Spring Tool Suite.

Wie der Name schon andeutet, ist die Spring Tool Suite eine hervorragende Spring-Entwicklungsumgebung. Sie bietet aber auch ein praktisches Spring-Boot-Dashboard-Feature, das in keiner der anderen IDE-Optionen verfügbar ist (zumindest bei Redaktionsschluss dieses Buches).

Wenn Sie nicht mit der Spring Tool Suite arbeiten, ist das auch in Ordnung, wir können trotzdem Freunde bleiben. Springen Sie zum Anhang und ersetzen Sie jeweils die Initializr-Option durch die für Sie am besten geeigneten Anweisungen in den folgenden Abschnitten. Beachten Sie aber, dass ich mich in diesem Buch gelegentlich auf spezielle Features der Spring Tool Suite beziehe, beispielsweise das Spring Boot Dashboard. Wenn Sie die Spring Tool Suite nicht verwenden, müssen Sie diese Anweisungen entsprechend Ihrer IDE anpassen.

1.2.1 Ein Spring-Projekt mit der Spring Tool Suite initialisieren

Um mit einem neuen Spring-Projekt zu beginnen, wählen Sie im Menü File den Befehl New und dann Spring Starter Project. Bild 1.2 zeigt die Menüstruktur.

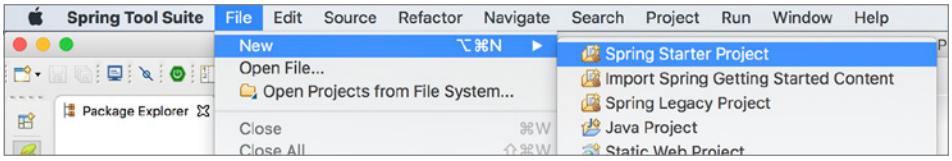

Bild 1.2 Ein neues Projekt mit dem Initializr in der Spring Tool Suite starten

Nachdem Sie Spring Starter Project ausgewählt haben, erscheint ein Assistentendialogfeld für ein neues Projekt (Bild 1.3). Die erste Seite des Assistenten fragt einige allgemeine Projektinformationen ab, unter anderem den Projektnamen, eine Beschreibung und andere wichtige Informationen. Wenn Sie mit dem Inhalt einer *pom.xml*-Datei von Maven vertraut sind, werden Sie die meisten Felder als Elemente erkennen, die in einer Maven-Build-Spezifikation landen. Füllen Sie das Dialogfeld für die Taco-Cloud-Anwendung wie in Bild 1.3 gezeigt aus und klicken Sie dann auf Next.

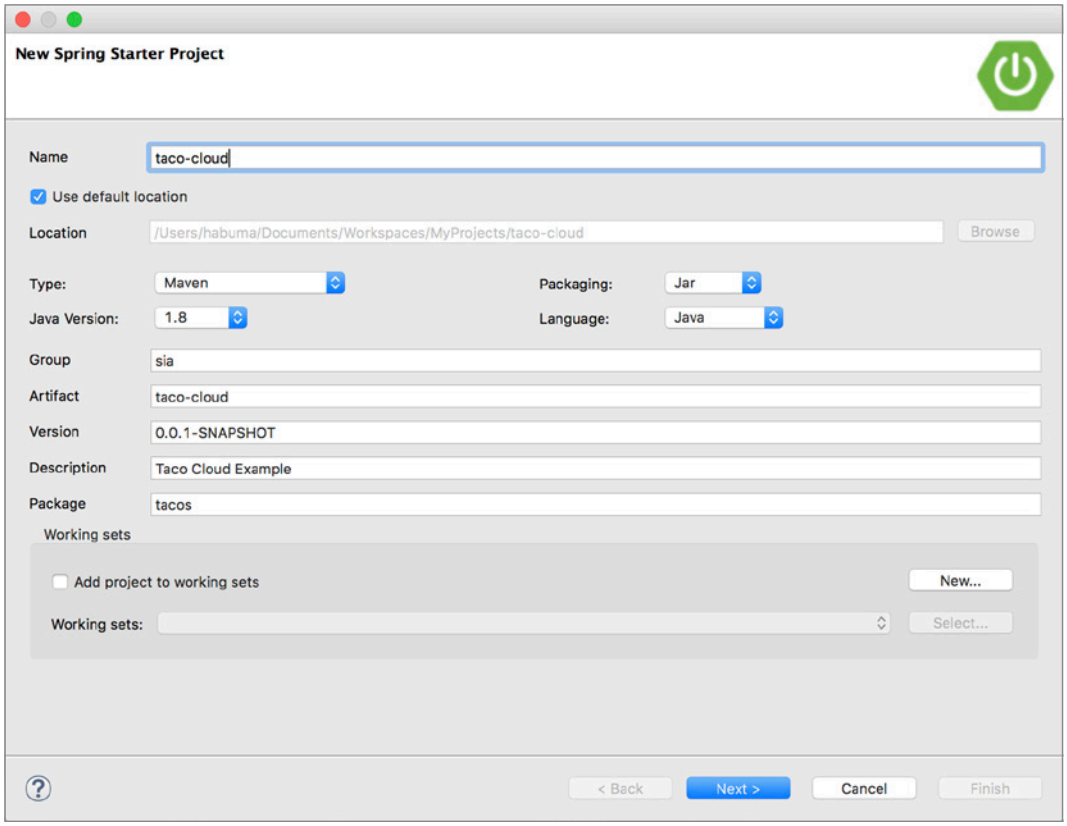

Bild 1.3 Allgemeine Projektinformationen für die Taco-Cloud-Anwendung festlegen

Auf der nächsten Seite des Assistenten wählen Sie die Abhängigkeiten aus, die Sie Ihrem Projekt hinzufügen möchten (siehe Bild 1.4). Im oberen Teil des Dialogfelds können Sie festlegen, auf welcher Version von Spring Ihr Projekt aufbauen soll. Standardeinstellung ist die aktuelle Version, die jeweils verfügbar ist. Im Allgemeinen empfiehlt es sich, die Vorgabe beizubehalten, außer wenn Sie eine andere Zielversion verwenden müssen.

Hinsichtlich der Abhängigkeiten selbst können Sie entweder die verschiedenen Abschnitte erweitern und die gewünschten Abhängigkeiten manuell heraussuchen oder Sie gehen über das Suchfeld im oberen Teil der Liste AVAILABLE. Für die Taco-Cloud-Anwendung beginnen Sie mit den in Bild 1.4 gezeigten Abhängigkeiten.

An dieser Stelle können Sie auf FINISH klicken, um das Projekt zu generieren und es Ihrem Arbeitsbereich hinzuzufügen. Doch wenn Sie unternehmungslustig sind, klicken Sie noch einmal auf NEXT, um die letzte Seite des Assistenten, die in Bild 1.5 dargestellt ist, für ein neues Projekt zu öffnen.

Standardmäßig ruft der Assistent für ein neues Projekt den Spring Initializr unter http://start.spring.io auf, um das Projekt zu generieren. Da Sie diese Standardeinstellung normalerweise nicht überschreiben müssen, können Sie auf der zweiten Seite des Assistenten auf FINISH klicken. Falls Sie aber aus irgendeinem Grund Ihren eigenen Klon von Initializr