Aktuell zu Spring 5.0

craig WALLS

SPRING IM EINSATZ

3. Auflage

HANSER

Walls

Spring im Einsatz

Craig Walls

Spring
im Einsatz

3., überarbeitete Auflage

HANSER

Übersetzung: Frank Langenau, Chemnitz
Titel der Originalausgabe: „Spring in Action“, 5th Edition, © 2019 by Manning Publications Co.

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Lektorat: Brigitte Bauer-Schiewek
Copy editing: Petra Kienle, Fürstenfeldbruck
Layout: Manuela Treindl, Fürth
Umschlagdesign: Marc Müller-Bremer, München, www.rebranding.de
Umschlagrealisation: Max Kostopoulos
Titelmotiv: © Stephan Rönigk
Datenbelichtung, Druck und Bindung: Kösel, Krugzell
Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702
Printed in Germany

Print-ISBN: 978-3-446-45512-2
E-Book-ISBN: 978-3-446-45731-7
E-Pub-ISBN: 978-3-446-46323-3

Inhalt

Vorwort

Nachdem ich fast 15 Jahre mit Spring gearbeitet und mehrere Ausgaben dieses Buches geschrieben habe (ganz zu schweigen von *„Spring Boot in Action"*), sollte man meinen, dass sich kaum noch etwas Aufregendes und Neues über Spring sagen lässt, wenn es um das Vorwort für dieses Buch geht. Aber wie so oft sieht es in der Realität ganz anders aus!

Jedes einzelne Release von Spring, Spring Boot und allen anderen Projekten im Spring-Ökosystem schafft neue erstaunliche Möglichkeiten, die den Spaß an der Entwicklung von Anwendungen wieder aufleben lassen. Mit dem Release 5.0 von Spring und dem Release 2.0 von Spring Boot gibt es so viel mehr Spring zu genießen, dass es ein Kinderspiel war, eine weitere Ausgabe von *Spring im Einsatz* zu schreiben.

Das Großartige an Spring 5 ist die Unterstützung für reaktive Programmierung, unter anderem für Spring WebFlux, ein brandneues reaktives Web-Framework, dessen Programmiermodell sich an Spring MVC orientiert. Damit können Entwickler Webanwendungen schaffen, die sich besser skalieren lassen und weniger Threads effektiver nutzen. In Richtung des Backends einer Spring-Anwendung erlaubt es die neueste Edition von Spring Data, reaktive, nicht blockierende Daten-Repositories aufzubauen. Und alles dies baut auf Project Reactor auf, einer Java-Bibliothek für das Arbeiten mit reaktiven Typen.

Zusätzlich zu den neuen reaktiven Programmierfeatures von Spring 5 bietet Spring Boot 2 nun sogar mehr Unterstützung als je zuvor für die Autokonfiguration sowie einen vollständig neu konzipierten Actuator, mit dem sich eine laufende Anwendung inspizieren und manipulieren lässt.

Da Entwickler zudem ihre monolithischen Anwendungen in diskrete Microservices aufteilen wollen, bietet Spring Cloud Einrichtungen, die es erleichtern, Microservices zu konfigurieren, zu erkennen und sie widerstandsfähiger gegen Ausfälle zu machen.

Ich freue mich, sagen zu können, dass diese fünfte Ausgabe von *Spring im Einsatz* – hier vorliegend als Übersetzung in der dritten Auflage – alle diese und noch mehr Themen abdeckt! Wenn Sie ein erfahrener Veteran von Spring sind, wird *Spring im Einsatz* Ihr Leitfaden für alles Neue sein, das Spring zu bieten hat. Wenn Sie andererseits neu in Spring einsteigen, dann gibt es keinen besseren Zeitpunkt als jetzt, um richtig loszulegen. Die ersten Kapitel bringen Sie im Handumdrehen an den Start!

Die 15 Jahre Arbeit mit Spring sind eine spannende Zeit gewesen. Und da nun diese Edition von *Spring im Einsatz* vor Ihnen liegt, bin ich versessen darauf, diese Begeisterung mit Ihnen zu teilen!

Danksagungen

Zu den erstaunlichsten Dingen bei Spring und Spring Boot ist zu nennen, dass sie automatisch alle grundlegenden Installationen für eine Anwendung bereitstellen, wodurch Sie sich als Entwickler vorrangig auf die Logik konzentrieren können, die Ihre Anwendung im Speziellen ausmacht. Leider gibt es keine solchen magischen Hilfsmittel, um ein Buch zu schreiben. Oder etwa doch?

Bei Manning haben mehrere Leute ihre magischen Kräfte entfaltet, um sicherzustellen, dass dieses Buch das Beste wird, was möglich ist. Vielen Dank insbesondere an Jenny Stout, meine Entwicklungsredakteurin, und das Produktteam, darunter Projektleiterin Janet Vail, die Copyeditoren Andy Carroll und Frances Buran sowie das Korrektorat mit Katie Tennant und Melody Dolab. Dank auch dem Fachlektor Joshua White, der gründlich und hilfreich war.

In dieser Zeit haben wir auch Feedback von mehreren Gutachtern erhalten, die dafür gesorgt haben, den Kurs zu halten und die richtigen Themen abzudecken. Dafür danke ich Andrea Barisone, Arnaldo Ayala, Bill Fly, Colin Joyce, Daniel Vaughan, David Witherspoon, Eddu Melendez, Iain Campbell, Jettro Coenradie, John Gunvaldson, Markus Matzker, Nick Rakochy, Nusry Firdousi, Piotr Kafel, Raphael Villela, Riccardo Noviello, Sergio Fernandez Gonzalez, Sergiy Pylypets, Thiago Presa, Thorsten Weber, Waldemar Modzelewski, Yagiz Erkan und Željko Trogrlić.

Wie immer gäbe es absolut keinen Grund, dieses Buch zu schreiben, wenn da nicht die erstaunliche Arbeit des Spring-Entwicklerteams wäre. Ich kann nur darüber staunen, was es geschaffen hat und wie wir den Entwicklungsstil von Software immer wieder verändern.

Ein großer Dank geht an meine Mitstreiter auf der No Fluff/Just Stuff-Tour. Ich lerne weiterhin so viel von jedem von euch! Besonders danken möchte ich Brian Sletten, Nate Schutta und Ken Kousen für die Gespräche und E-Mails über Spring, die zur Gestaltung dieses Buches beigetragen haben.

Nochmals vielen Dank an die Phönizier. Ihr wisst, was ihr getan habt.

An meine wundervolle Frau Raymie, die Liebe meines Lebens, meinen süßesten Traum und meine Inspiration gerichtet: Danke für dein Engagement und dafür, dass du dich mit einem weiteren Buchprojekt abgefunden hast. Und an meine süßen und wundervollen Mädchen, Maisy und Madi: Ich bin so stolz auf euch und auf die erstaunlichen jungen Damen, die ihr einmal sein werdet. Ich liebe euch alle mehr, als ihr es euch vorstellen könnt oder ich es auszudrücken vermag.

Über dieses Buch

Spring im Einsatz soll Sie in die Lage versetzen, erstaunliche Anwendungen mit dem Spring Framework, Spring Boot und einer breiten Palette von Ergänzungstools des Spring-Ökosystems zu erstellen. Zunächst erfahren Sie, wie Sie webbasierte, datenbankgestützte Java-Anwendungen mit Spring und Spring Boot entwickeln. Anschließend geht es über die Grundlagen hinaus und es wird gezeigt, wie Sie die Integration mit anderen Anwendungen realisieren, mit reaktiven Typen programmieren und dann eine Anwendung in diskrete Microservices aufteilen. Schließlich wird erörtert, wie Sie eine Anwendung für die Bereitstellung fit machen.

Obwohl alle Projekte im Spring-Ökosystem eine ausgezeichnete Dokumentation bieten, gibt Ihnen dieses Buch etwas, was Sie in den Referenzdokumentationen nicht finden: einen praktischen, projektgetriebenen Leitfaden, um die Elemente von Spring im Rahmen einer realen Anwendung zusammenzubringen.

Wer dieses Buch lesen sollte

Die vorliegende Ausgabe von *Spring im Einsatz*, richtet sich an Java-Entwickler, die erste Schritte mit Spring Boot und dem Spring Framework unternehmen möchten, sowie an erfahrene Spring-Entwickler, die über die Grundlagen hinausgehen und die neuesten Features von Spring kennenlernen wollen.

Wie dieses Buch organisiert ist: eine Roadmap

Das Buch umfasst 19 Kapitel, die in fünf Teile gegliedert sind. **Teil I** befasst sich mit den grundlegenden Themen für das Erstellen von Anwendungen:

- *Kapitel 1* führt Spring und Spring Boot ein und zeigt, wie Sie ein Spring-Projekt initialisieren. In diesem Kapitel unternehmen Sie die ersten Schritte und erstellen eine Spring-Anwendung, die Sie im weiteren Verlauf des Buches erweitern und vervollständigen werden.
- *Kapitel 2* erläutert, wie Sie die Web-Ebene einer Anwendung mit Spring MVC aufbauen. In diesem Kapitel erstellen Sie Controller, die Webanfragen behandeln, und Views, die Informationen im Webbrowser darstellen.
- *Kapitel 3* beschäftigt sich mit dem Backend einer Spring-Anwendung, wo die Daten in einer relationalen Datenbank persistent gespeichert werden.
- In *Kapitel 4* nutzen Sie Spring Security, um Benutzer zu authentifizieren und nicht autorisierten Zugriff auf eine Anwendung zu verhindern.
- *Kapitel 5* macht deutlich, wie Sie eine Spring-Anwendung mit Spring-Boot-Konfigurationseigenschaften konfigurieren. Außerdem lernen Sie, wie Sie eine Konfiguration mithilfe von Profilen selektiv anwenden.

Die Themen in **Teil II** helfen Ihnen, wenn Sie Ihre Spring-Anwendung mit anderen Anwendungen integrieren:

- *Kapitel 6* erweitert die in Kapitel 2 begonnene Diskussion zu Spring MVC und zeigt, wie sich REST APIs in Spring schreiben lassen.
- *Kapitel 7* tauscht die Rollen gegenüber Kapitel 6 und zeigt, wie eine Spring-Anwendung eine REST API konsumieren kann.
- *Kapitel 8* befasst sich mit asynchroner Kommunikation, damit eine Spring-Anwendung per Java Message Service, RabbitMQ oder Kafka Nachrichten sowohl senden als auch empfangen kann.
- In *Kapitel 9* geht es um deklarative Anwendungsintegration mit dem Spring-Integration-Projekt.

Teil III ist der neuen Unterstützung für reaktive Programmierung in Spring gewidmet:

- *Kapitel 10* stellt Project Reactor vor, die reaktive Programmierbibliothek, die die reaktiven Features von Spring 5 untermauert.
- *Kapitel 11* beschäftigt sich noch einmal mit der REST-API-Entwicklung und führt Spring WebFlux ein, ein neues Web-Framework, das sich stark an Spring MVC orientiert, dabei aber ein neues reaktives Modell für die Web-Entwicklung bietet.
- In *Kapitel 12* werfen wir einen Blick darauf, wie sich reaktive Datenpersistenz mit Spring Data programmieren lässt, um Daten in und aus den Datenbanken Cassandra und Mongo zu schreiben und zu lesen.

Teil IV zerlegt das monolithische Anwendungsmodell und führt Sie in die Entwicklung mit Spring Cloud und Microservices ein:

- *Kapitel 13* befasst sich mit dem Erkennen von Diensten, wobei Sie Spring mit der Eureka-Registrierung von Netflix verwenden, um Spring-basierte Microservices sowohl zu registrieren als auch zu erkennen.
- In *Kapitel 14* zentralisieren Sie die Anwendungskonfiguration auf einem Konfigurationsserver, der die Konfiguration für mehrere Microservices gemeinsam nutzt.
- *Kapitel 15* führt das Trennschalter-Muster (Circuit Breaker) mit Hystrix ein, das es ermöglicht, Microservices für den Fehlerfall robuster zu machen.

In **Teil V** bereiten Sie eine Anwendung für die Produktion vor und lernen, wie Sie sie bereitstellen:

- *Kapitel 16* stellt den Spring Boot Actuator vor, eine Erweiterung zu Spring Boot, mit der sich die Interna einer laufenden Spring-Anwendung als REST-Endpunkte zugänglich machen lassen.
- In *Kapitel 17* sehen Sie, wie Sie mit Spring Boot Admin eine benutzerfreundliche browserbasierte administrative Anwendung auf Actuator aufsetzen können.
- *Kapitel 18* erläutert, wie sich Spring-Beans als JMX MBeans zugänglich machen und konsumieren lassen.
- Abschließend zeigt *Kapitel 19*, wie Sie Ihre Spring-Anwendung in den verschiedensten Produktionsumgebungen bereitstellen.

Im Allgemeinen sollten Entwickler, die in Spring einsteigen, mit Kapitel 1 beginnen und alle Kapitel nacheinander durcharbeiten. Erfahrene Spring-Entwickler ziehen es vielleicht vor, gleich mit dem Thema zu beginnen, das sie vorrangig interessiert. Es sei aber darauf

hingewiesen, dass jedes Kapitel auf dem vorhergehenden aufbaut, sodass möglicherweise der Kontext unklar ist, wenn Sie gleich in der Mitte des Buches einsteigen.

Über den Code

Dieses Buch enthält viele Beispiele im Quellcode sowohl in nummerierten Listings als auch in Form von Codefragmenten, die in den laufenden Text eingefügt sind. In beiden Fällen ist der Quellcode in `Einer-Schreibmaschinenschrift-wie-dieser` formatiert, um ihn vom normalen Text zu trennen. Manchmal ist der Code auch **fett** gedruckt, um ihn von Code abzuheben, der sich gegenüber vorherigen Schritten im Kapitel geändert hat, beispielsweise wenn ein neues Feature zu einer schon vorhandenen Codezeile hinzukommt.

Der Quellcode zu den Beispielen in diesem Buch steht auf der Website des Verlages der Originalausgabe unter www.manning.com/books/spring-in-action-fifth-edition sowie im GitHub-Konto des Autors unter github.com/habuma/spring-in-action-5-samples zum Download bereit.

Buchforum

Beim Kauf dieser Ausgabe von *Spring im Einsatz* erhalten Sie kostenfreien Zugriff auf ein privates Web-Forum unter Leitung von Manning Publications, in dem Sie Kommentare zum Buch abgeben, technische Fragen stellen und Hilfe vom Autor und von anderen Benutzern erhalten können. Um auf das Forum zuzugreifen, besuchen Sie https://forums.manning.com/forums/spring-in-action-fifth-edition. Mehr über die Foren von Manning und die Verhaltensregeln erfahren Sie auch unter https://forums.manning.com/forums/about.

Manning engagiert sich für die Leser, um ihnen einen Treffpunkt zu bieten, an dem ein sinnvoller Dialog zwischen Lesern untereinander und zwischen dem Leser und dem Autor stattfinden kann. Es handelt sich dabei nicht um eine Verpflichtung dem Autor gegenüber, in einem bestimmten Umfang mitzuwirken, wobei der Beitrag zum Forum freiwillig (und unbezahlt) bleibt. Stellen Sie doch dem Autor herausfordernde Fragen, damit sein Interesse nicht verloren geht! Das Forum und die Archive früherer Diskussionen sind über Website des Verlags zugänglich, solange das Buch lieferbar ist.

Andere Online-Quellen

Brauchen Sie zusätzliche Hilfe?

- Die Spring-Website bietet unter https://spring.io/guides mehrere nützliche Leitfäden (von denen der Autor selbst einige geschrieben hat).
- Die Tags für Spring auf StackOverflow (https://stackoverflow.com/questions/tagged/spring) und Spring Boot auf StackOverflow sind empfehlenswerte Orte rund um das Thema Spring, um Fragen zu stellen und anderen zu helfen. Gleichzeitig ist es ein guter Weg, um mehr über Spring zu lernen, wenn man anderen bei der Beantwortung ihrer Spring-Fragen hilft!

Über den Autor

Craig Walls ist leitender Ingenieur bei Pivotal. Er ist ein eifriger Förderer des Spring Frameworks, ist häufig in lokalen Benutzergruppen und Konferenzen anzutreffen und schreibt über Spring. Wenn er nicht gerade mit Code um sich wirft, plant Craig seine nächste Reise nach Disney World oder Disneyland und verbringt so viel Zeit wie möglich mit seiner Frau, seinen beiden Töchtern, zwei Vögeln und drei Hunden.

Teil I: Die Grundzüge von Spring

In Teil I dieses Buches schreiben Sie zum Einstieg eine Spring-Anwendung und lernen dabei die Grundlagen von Spring kennen.

- *Kapitel 1* gibt Ihnen einen kurzen Überblick über die wesentlichen Elemente von Spring und Spring Boot und zeigt, wie Sie ein Spring-Projekt initialisieren, während Sie Taco Cloud, Ihre erste Spring-Anwendung, erstellen.
- In *Kapitel 2* dringen Sie tiefer in Spring MCV ein und lernen, wie Sie Modelldaten im Browser darstellen und wie Sie Formulareingaben verarbeiten und auf Gültigkeit überprüfen. Außerdem bekommen Sie einige Tipps zur Auswahl einer View-Vorlagenbibliothek.
- In *Kapitel 3* statten Sie die Taco-Cloud-Anwendung mit Datenpersistenz aus. Dabei lernen Sie die JDBC-Vorlage von Spring kennen und erfahren, wie Sie Daten einfügen und wie Sie JPA-Repositories mit Spring Data deklarieren.
- *Kapitel 4* befasst sich mit der Sicherheit für Ihre Spring-Anwendung. Dabei geht es um die automatische Konfiguration von Spring Security, das Definieren benutzerdefinierter Benutzerspeicher, die Anpassung der Anmeldeseite und die Absicherung gegen CSRF-(Cross-Site-Request-Forgery-)Angriffe.
- Zum Abschluss von Teil I geht *Kapitel 5* auf Konfigurationseigenschaften ein. Hier lernen Sie, wie Sie automatisch konfigurierte Beans optimieren, Konfigurationseigenschaften auf Anwendungskomponenten anwenden und mit Spring-Profilen arbeiten.

1 Erste Schritte mit Spring

Die Themen dieses Kapitels:

- Die Grundzüge von Spring und Spring Boot
- Ein Spring-Projekt initialisieren
- Die Spring-Landschaft im Überblick

Obwohl der griechische Philosoph Heraklit nicht gerade als Softwareentwickler bekannt war, schien er das Thema gut im Griff zu haben. Er wird mit den Worten zitiert: »Die einzige Konstante ... ist die Veränderung.« In dieser Aussage steckt eine grundlegende Wahrheit der Softwareentwicklung.

Heute entwickeln wir Anwendungen in einer anderen Art und Weise als vor einem Jahr, vor fünf Jahren, vor zehn Jahren und zweifellos vor 15 Jahren, als Rod Johnson in seinem Buch *Expert One-on-One J2EE Design and Development* (Wrox, 2002, http://mng.bz/oVjy) eine anfängliche Form des Spring Frameworks eingeführt hat.

Damals hat man vor allem browserbasierte Webanwendungen entwickelt, die von relationalen Datenbanken unterstützt wurden. Auch wenn diese Art der Entwicklung immer noch eine Rolle spielt und Spring für derartige Anwendungen gut gerüstet ist, sind wir heute auch interessiert an der Entwicklung von Anwendungen, die aus Microservices bestehen und darauf ausgelegt sind, Daten dauerhaft in der Cloud in verschiedensten Datenbanken zu speichern. Zudem zielt ein neu erwachtes Interesse an reaktiver Programmierung darauf ab, mit nicht blockierenden Operationen größere Skalierbarkeit und verbesserte Performance zu erreichen.

Mit dem Voranschreiten der Softwareentwicklung hat sich auch das Spring Framework geändert, um modernen Entwicklungsansprüchen gerecht zu werden. Dazu gehören auch Microservices und reaktive Programmierung. Mit der Einführung von Spring Boot tritt Spring auch an, um sein eigenes Entwicklungsmodell zu vereinfachen.

Egal, ob Sie eine einfache datenbankgestützte Webanwendung entwickeln oder eine moderne Anwendung um Microservices herum aufbauen – das Spring-Framework hilft Ihnen, Ihre Ziele zu erreichen. Dieses Kapitel ist der erste Schritt auf einer Tour durch die moderne Anwendungsentwicklung mit Spring.

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.

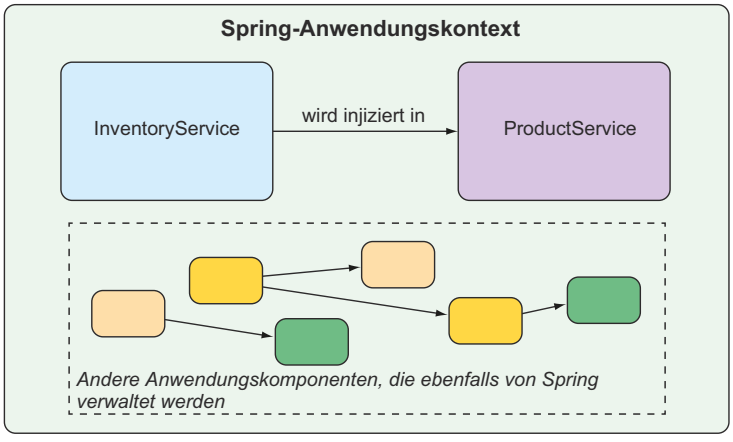

Bild 1.1 Anwendungskomponenten werden durch den Spring-Anwendungskontext verwaltet und ineinander injiziert.

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.

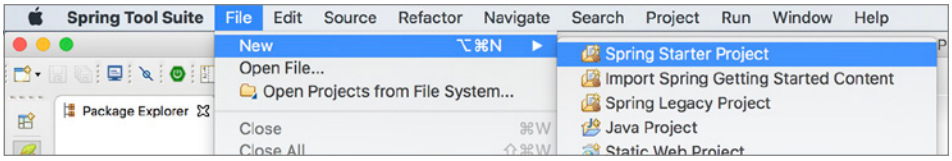

Bild 1.2 Ein neues Projekt mit dem Initializr in der Spring Tool Suite starten

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.

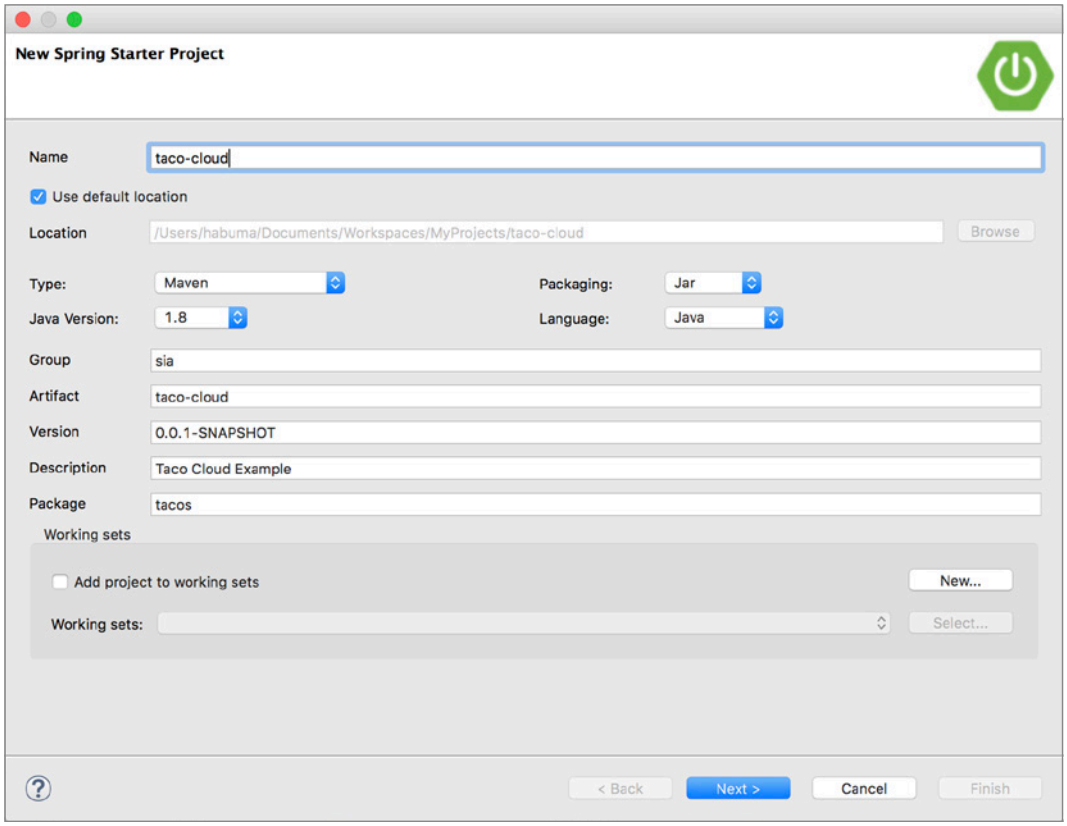

Bild 1.3 Allgemeine Projektinformationen für die Taco-Cloud-Anwendung festlegen

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

hosten (vielleicht als lokale Kopie auf Ihrem Computer oder einen zugeschnittenen Klon, der innerhalb der Firewall Ihrer Firma läuft), können Sie im Feld Base Url die Adresse Ihrer Instanz von Initializr eintragen, bevor Sie auf Finish klicken.

Nachdem Sie auf Finish geklickt haben, wird das Projekt vom Initializr heruntergeladen und in Ihren Arbeitsbereich gebracht. Warten Sie etwas, bis das Laden und Erstellen abgeschlossen ist. Dann können Sie beginnen, die Funktionalität der Anwendung zu entwickeln. Werfen Sie vorher aber einen Blick auf das, was der Initializr Ihnen geliefert hat.

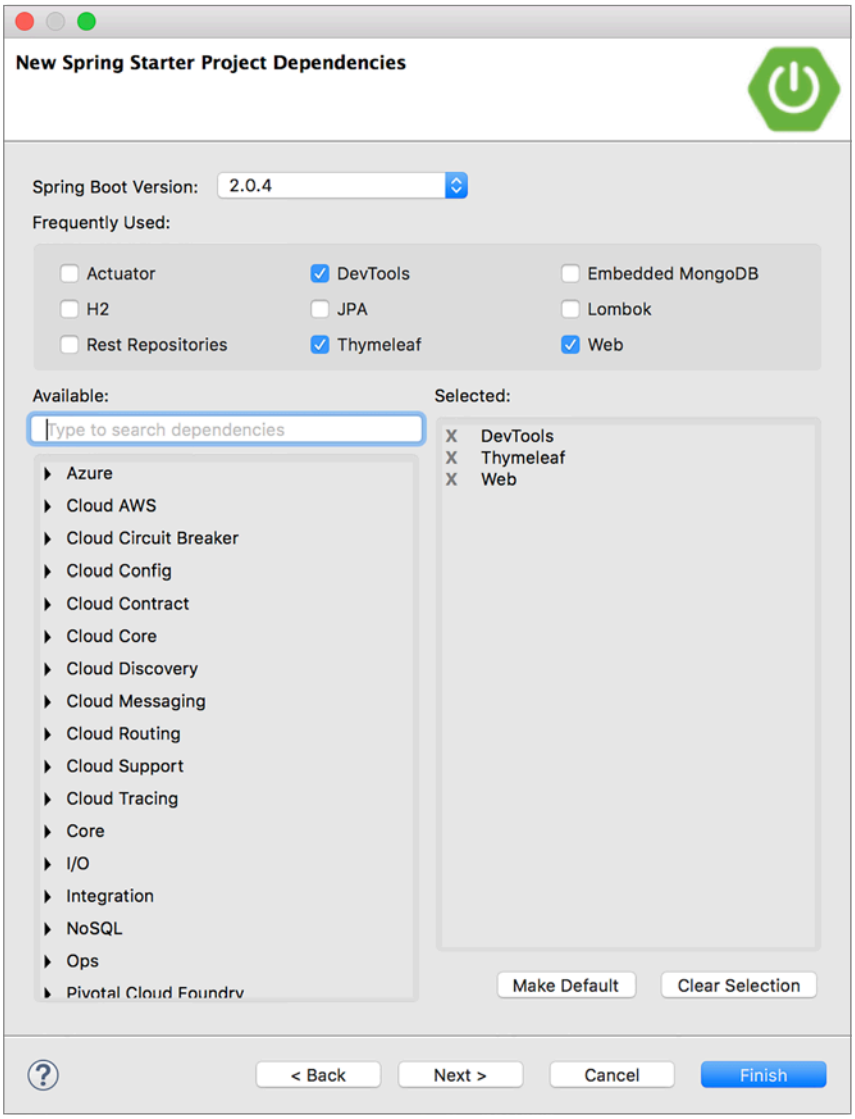

Bild 1.4 Starter-Abhängigkeiten auswählen

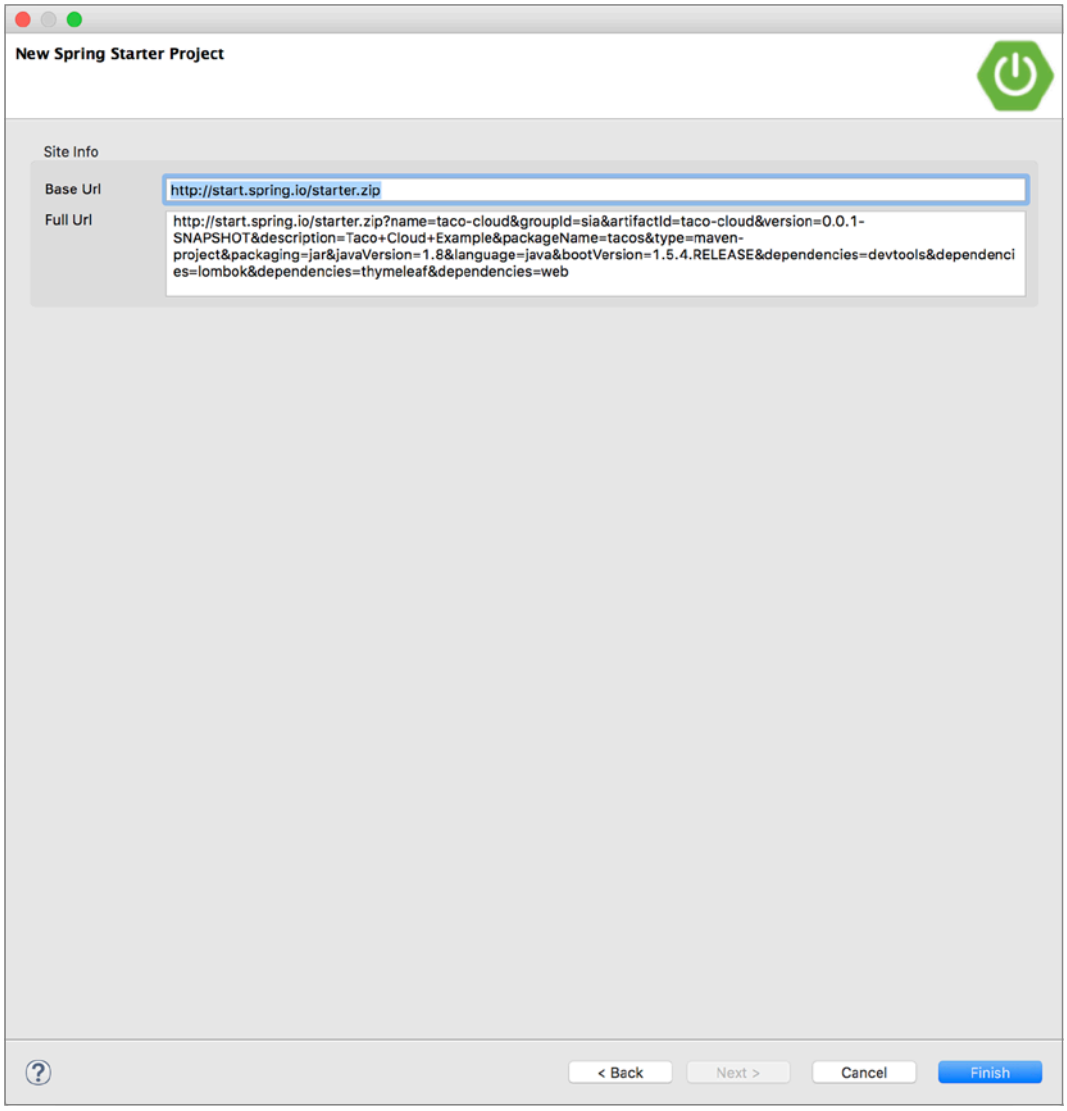

Bild 1.5 Optional eine alternative Initializr-Adresse angeben

1.2.2 Die Spring-Projektstruktur untersuchen

Nachdem das Projekt in der IDE geladen ist, erweitern Sie es, um den Inhalt zu sehen. Bild 1.6 zeigt das erweiterte Taco-Cloud-Projekt in der Spring Tool Suite.

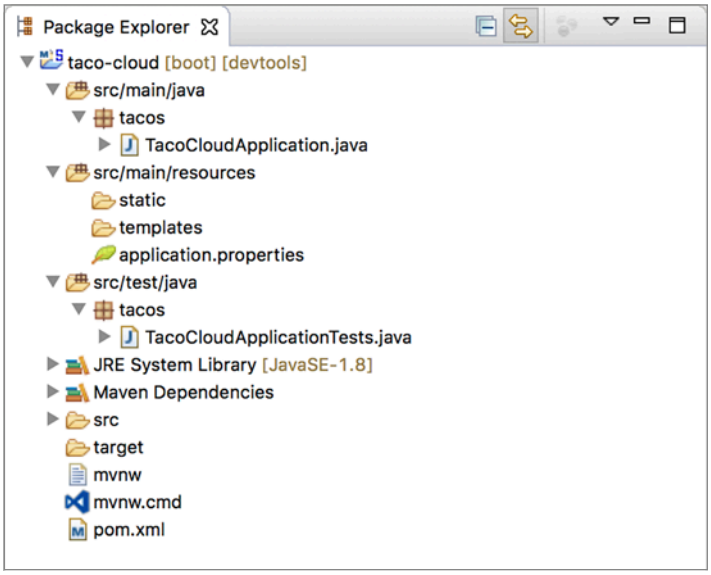

Bild 1.6 Die anfängliche Spring-Projektstruktur, wie sie sich in der Spring Tool Suite darstellt

Vielleicht erkennen Sie in dieser Struktur eine typische Maven- oder Gradle-Projektstruktur wieder, wobei der Quellcode unter *src/main/java*, der Testcode unter *src/test/java* und Java-fremde Ressourcen unter *src/main/resources* gespeichert sind. Innerhalb dieser Projektstruktur finden Sie die folgenden Elemente:

- `mvnw` und `mvnw.cmd` – das sind Maven-Wrapper-Skripts. Mit diesen Skripts können Sie Ihr Projekt erstellen, selbst wenn Sie Maven nicht auf Ihrem Computer installiert haben.
- *pom.xml* – diese Datei enthält die Maven-Build-Spezifikation. Mehr dazu gleich.
- `TacoCloudApplication.java` – dies ist die `main`-Klasse von Spring Boot, die das Projekt hochfährt. Auch hierzu gleich mehr.
- *application.properties* – diese Datei ist anfangs leer, bietet aber einen Platz, wo Sie Konfigurationseigenschaften unterbringen können. Mit dieser Datei befassen wir uns etwas später in diesem Kapitel, eine ausführliche Erläuterung der Konfigurationseigenschaften folgt erst in Kapitel 5.
- *static* – dieser Ordner ist für statische Inhalte vorgesehen (Bilder, Stylesheets, JavaScript usw.), die Sie dem Browser zur Verfügung stellen wollen. Der Ordner ist anfangs leer.
- *templates* – in diesem Ordner platzieren Sie Vorlagendateien, die für das Rendern der Inhalte im Browser vorgesehen sind. Anfangs ist der Ordner leer, doch Sie werden bald eine Thymeleaf-Vorlage hinzufügen.
- `TacoCloudApplicationTests.java` – diese einfache Testklasse stellt sicher, dass sich der Spring-Anwendungskontext erfolgreich laden lässt. Weitere Tests fügen Sie diesem Mix im Verlauf der Anwendungsentwicklung hinzu.

Wenn die Taco-Cloud-Anwendung wächst, füllen Sie diese minimale Projektstruktur mit Java-Code, Bildern, Stylesheets, Tests und anderen Elementen, um Ihr Projekt zu komplettieren. Bis dahin wollen wir uns näher mit einigen Elementen befassen, die Spring Initializr bereitgestellt hat.

Die Build-Spezifikation untersuchen

Im Initializr-Formular haben Sie auch festgelegt, dass Ihr Projekt mit Maven erstellt werden soll. Deshalb hat der Spring Initializr eine *pom.xml*-Datei erzeugt, die bereits die von Ihnen festgelegten Einstellungen enthält. Listing 1.1 zeigt die gesamte *pom.xml*-Datei, die Initializr angelegt hat.

Listing 1.1 Die anfängliche Maven-Build-Spezifikation

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
          http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>sia</groupId>
  <artifactId>taco-cloud</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>    ◂ JAR-Packaging

  <name>taco-cloud</name>
  <description>Taco Cloud Example</description>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.4.RELEASE</version>    ◂ Spring Boot-Version
    <relativePath/> <!-- lookup parent from repository -->
  </parent>

  <properties>
    <project.build.sourceEncoding>
        UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>
        UTF-8</project.reporting.outputEncoding>
    <java.version>1.8</java.version>
  </properties>

  <dependencies>
    <dependency>    ◂ Starter-Abhängigkeiten
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
```

```
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>org.seleniumhq.selenium</groupId>
      <artifactId>selenium-java</artifactId>
        <scope>test</scope>
    </dependency>

     <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>htmlunit-driver</artifactId>
        <scope>test</scope>
      </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>    ◂ Spring Boot-Plugin
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>

</project>
```

Das erste erwähnenswerte Element in der *pom.xml*-Datei ist `<packaging>`. Sie haben festgelegt, die Anwendung als ausführbare JAR-Datei im Unterschied zu einer WAR-Datei zu erstellen. Das ist wahrscheinlich eine der merkwürdigsten Entscheidungen, die Sie getroffen haben, insbesondere für eine Webanwendung. Schließlich werden herkömmliche Java-Webanwendungen als WAR-Dateien verpackt, sodass JAR-Dateien als bevorzugte Verpackung für Bibliotheken und die gelegentlichen Desktop-UI-Anwendungen bleiben.

Die Wahl der JAR-Verpackung ist eine Entscheidung, die auf die Cloud abzielt. Während WAR-Dateien für das Bereitstellen auf einem herkömmlichen Java-Anwendungsserver prädestiniert sind, eignen sie sich für die meisten Cloud-Plattformen nur bedingt. Zwar sind einige Cloud-Plattformen (wie zum Beispiel Cloud Foundry) in der Lage, WAR-Dateien bereitzustellen und auszuführen, eine ausführbare JAR-Datei ist aber auf sämtlichen Java-Cloud-Plattformen lauffähig. Deshalb stellt der Spring Initializr standardmäßig das JAR-Packaging ein, sofern Sie nichts anderes anweisen.

Wenn Sie Ihre Anwendung auf einem herkömmlichen Java-Anwendungsserver bereitstellen wollen, müssen Sie die Option WAR-Packaging wählen und eine Web-Initialisierungsklasse einbinden. Wie Sie WAR-Dateien erstellen, erläutert Kapitel 2 ausführlicher.

Sehen Sie sich als Nächstes das `<parent>`-Element an, speziell sein untergeordnetes `<version>`-Element. Es gibt an, dass Ihr Projekt `spring-boot-starter-parent` als sein übergeordnetes POM hat. Unter anderem bietet dieses übergeordnete POM eine Abhängigkeitsverwaltung für mehrere Bibliotheken, die in Spring-Projekten gebräuchlich sind. Für diejenigen Bibliotheken, die das übergeordnete POM abdeckt, brauchen Sie keine Version zu spezifizieren, da sie vom übergeordneten Element geerbt wird. Die Version `2.0.4.RELEASE` zeigt an, dass Sie mit Spring Boot 2.0.4 arbeiten. Somit erbt das Projekt die Abhängigkeitsverwaltung, wie sie durch diese Version von Spring Boot definiert wird.

Da wir schon beim Thema Abhängigkeiten sind: Beachten Sie, dass unter dem `<dependencies>`-Element drei Abhängigkeiten deklariert sind. Die ersten beiden sollten Ihnen bekannt vorkommen. Sie entsprechen direkt den Abhängigkeiten `Web` und `Thymeleaf`, die Sie im Assistenten für ein neues Projekt der Spring Tool Suite ausgewählt haben, bevor Sie auf die Schaltfläche FINISH geklickt haben. Durch die dritte Abhängigkeit wird eine ganze Reihe von hilfreichen Testmöglichkeiten bereitgestellt. Es ist nicht erforderlich, ein Kontrollkästchen zu aktivieren, um diese Abhängigkeit einzubinden, weil der Spring Initializr annimmt, dass Sie Tests schreiben (was hoffentlich auch zutrifft).

Sicherlich haben Sie auch bemerkt, dass alle drei Abhängigkeiten das Wort *starter* in ihrer Artefakt-ID enthalten. Die Spring-Boot-Starter-Abhängigkeiten sind insofern etwas Besonderes, als sie normalerweise selbst keinen Bibliothekscode enthalten, sondern stattdessen transitiv andere Bibliotheken heranziehen. Diese Starter-Abhängigkeiten bieten drei wesentliche Vorteile:

- Ihre Build-Datei wird deutlich kleiner und einfacher zu verwalten, weil Sie nicht von jeder Bibliothek, die Sie gegebenenfalls brauchen, eine Abhängigkeit deklarieren müssen.
- Sie können sich Abhängigkeiten in Bezug auf die von ihnen bereitgestellten Funktionen vorstellen statt in Form von Bibliotheksnamen. Wenn Sie eine Anwendung entwickeln, fügen Sie die Web-Starter-Abhängigkeit hinzu und keine lange Liste von individuellen Bibliotheken, mit denen Sie eine Webanwendung schreiben.
- Sie brauchen sich keine Gedanken mehr um Bibliotheksversionen zu machen. Sie dürfen darauf vertrauen, dass die transitiv eingebrachten Bibliotheken mit der angegebenen Version von Spring Boot kompatibel sind. Sie müssen nur noch entscheiden, welche Version von Spring Boot Sie verwenden.

Schließlich endet die Build-Spezifikation mit dem Spring-Boot-Plug-in. Dieses Plug-in führt einige wichtige Funktionen aus:

- Es stellt ein Maven-Ziel bereit, sodass Sie die Anwendung mithilfe von Maven ausführen können. Dieses Ziel werden Sie in Abschnitt 1.3.4 ausprobieren.
- Es stellt sicher, dass alle Abhängigkeitsbibliotheken in die ausführbare JAR-Datei eingebunden werden und im Laufzeit-Klassenpfad verfügbar sind.
- Es erzeugt eine Manifest-Datei in der JAR-Datei, die die Bootstrap-Klasse (in unserem Fall `TacoCloudApplication`) als `main`-Klasse für die ausführbare JAR-Datei benennt.

Da wir gerade von der Bootstrap-Klasse sprechen: Öffnen Sie die Datei, wir sehen sie uns jetzt genauer an.

Die Anwendung hochfahren

Da Sie die Anwendung von einer ausführbaren JAR-Datei starten, ist eine `main`-Klasse erforderlich, die beim Aufruf der JAR-Datei ausgeführt wird. Außerdem brauchen Sie eine minimale Spring-Konfiguration, um die Anwendung hochzufahren. Diese Elemente finden Sie in der Klasse `TacoCloudApplication`, die Listing 1.2 zeigt.

Listing 1.2 Die Taco-Cloud-Bootstrap-Klasse

```
package tacos;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication    ◂ Spring Boot-Anwendung
public class TacoCloudApplication {

  public static void main(String[] args) {
    SpringApplication.run(TacoCloudApplication.class, args);    ◂ führt die Anwendung aus

  }

}
```

Die Klasse `TacoCloudApplication` enthält zwar nur wenig Code, doch der hat es in sich. Eine der leistungsfähigsten Codezeilen ist gleichzeitig eine der kürzesten. Die Annotation `@SpringBootApplication` macht deutlich, dass es sich um eine Spring-Boot-Anwendung handelt. Doch hinter `@SpringBootApplication` steckt mehr, als es scheint.

`@SpringBootApplication` ist eine Verbund-Annotation, die drei andere Annotationen zusammenfasst:

- `@SpringBootConfiguration` – kennzeichnet diese Klasse als Konfigurationsklasse. Obwohl die Klasse kaum etwas an Konfiguration enthält, können Sie bei Bedarf dieser Klasse eine Java-basierte Spring-Framework-Konfiguration hinzufügen. In der Tat ist diese Annotation eine spezialisierte Form der `@Configuration`-Annotation.
- `@EnableAutoConfiguration` – aktiviert die automatische Spring-Boot-Konfiguration. Zur Autokonfiguration später mehr. Momentan genügt es zu wissen, dass diese Annotation Spring Boot anweist, alle Komponenten, die mutmaßlich benötigt werden, automatisch zu konfigurieren.
- `@ComponentScan` – aktiviert die Komponentensuche. Damit können Sie andere Klassen mit Annotationen wie `@Component`, `@Controller`, `@Service` und anderen deklarieren und Spring veranlassen, diese Klassen automatisch zu entdecken und sie als Komponenten im Spring-Anwendungskontext zu registrieren.

Der andere wichtige Teil von `TacoCloudApplication` ist die Methode `main()`. Diese Methode wird aufgerufen, wenn die JAR-Datei ausgeführt wird. Meistens enthält diese Methode Standardcode; jede Spring-Boot-Anwendung, die Sie schreiben, bekommt eine ähnliche oder sogar identische Methode (ungeachtet unterschiedlicher Klassennamen).

Die Methode `main()` ruft eine statische Methode `run()` der Klasse `SpringApplication` auf, die das eigentliche Bootstrapping der Anwendung durchführt und dabei den Spring-Anwendungskontext erzeugt. Der Methode `run()` werden zwei Parameter übergeben: eine Konfigurationsklasse und die Befehlszeilenargumente. Die an `run()` übergebene Konfigurationsklasse muss nicht dieselbe Klasse wie die Bootstrap-Klasse sein, doch ist das die zweckmäßigste und typische Wahl.

Aller Wahrscheinlichkeit nach brauchen Sie in der Bootstrap-Klasse gar nichts zu ändern. Bei einfachen Anwendungen mag es zweckmäßig sein, eine oder zwei andere Komponenten in der Bootstrap-Klasse zu konfigurieren, doch bei den meisten Anwendungen kommen Sie besser, für alles, was nicht automatisch konfiguriert wird, eine separate Konfigurationsklasse zu erstellen. Im Rahmen dieses Buches werden Sie mehrere Konfigurationsklassen definieren. Seien Sie also gespannt auf die Details.

Die Anwendung testen

Testen ist ein wichtiger Bestandteil der Softwareentwicklung. Deshalb gibt Ihnen der Spring Initializr für die ersten Schritte eine Testklasse. Listing 1.3 zeigt die grundlegende Testklasse.

Listing 1.3 Ein grundlegender Anwendungstest

```
package tacos;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)    ◂ Verwendet den Spring-Runner
@SpringBootTest    ◂ Ein Spring Boot-Test
public class TacoCloudApplicationTests {

  @Test    ◂ Die Testmethode
  public void contextLoads() {
  }

}
```

In der Klasse `TacoCloudApplicationTests` ist nicht viel zu sehen: Die einzige Testmethode in der Klasse ist leer. Dennoch führt diese Testklasse eine entscheidende Prüfung durch, um sicherzustellen, dass der Spring-Anwendungskontext erfolgreich geladen werden kann. Wenn Sie durch irgendwelche Änderungen verhindern, dass der Sprint-Anwendungskontext erstellt wird, scheitert dieser Test, und Sie können darauf reagieren, indem Sie das Problem beheben.

Beachten Sie auch die Klasse, die mit `@RunWith(SpringRunner.class)` annotiert ist. Bei `@RunWith` handelt es sich um eine JUnit-Annotation, die einen Test-Runner bereitstellt, der JUnit beim Ausführen eines Tests steuert. Stellen Sie sich das so vor, als würden Sie ein Plug-in auf JUnit anwenden, um ein benutzerdefiniertes Testverhalten bereitzustellen. In diesem Fall wird JUnit `SpringRunner` übergeben, ein von Spring bereitgestellter Test-Runner, der für das Erstellen eines Spring-Anwendungskontextes zuständig ist, gegen den der Test ausgeführt wird.

Ein Test-Runner mit anderem Namen ...

Wenn Sie bereits damit vertraut sind, Spring-Tests zu schreiben, oder sich vielleicht einige vorhandene Spring-basierte Testklassen angesehen haben, ist Ihnen möglicherweise ein Test-Runner namens `SpringJUnit4ClassRunner` begegnet. `SpringRunner` ist ein Alias für `SpringJUnit4ClassRunner` und wurde in Spring 4.3 eingeführt, um die Verbindung mit einer spezifischen Version von JUnit zu beseitigen (zum Beispiel JUnit 4). Und es lässt sich nicht leugnen, dass der Alias einfacher zu lesen und zu schreiben ist.

Die Annotation `@SpringBootTest` weist JUnit an, den Test mit Spring-Boot-Fähigkeiten hochzufahren. Stellen Sie sich dies fürs Erste als Testklassenäquivalent für den Aufruf von `Spring-Application.run()` in einer Methode `main()` vor. In diesem Buch werden Sie `@SpringBootTest` mehrmals sehen und wir werden einen Teil seiner Leistung aufdecken.

Schließlich haben wir die Testmethode selbst. Obwohl die Annotationen `@RunWith (SpringRunner.class)` und `@SpringBootTest` dafür zuständig sind, den Spring-Anwendungskontext für den Test zu laden, würden sie nichts zu tun haben, wenn es keine Testmethoden gäbe. Selbst ohne irgendwelche Assertionen oder sonstigen Code besteht diese leere Testmethode auf den beiden Annotationen, damit sie ihren Job erledigen und den Spring-Anwendungskontext laden kann. Treten dabei irgendwelche Probleme auf, scheitert der Test.

Soweit an dieser Stelle der Überblick über den vom Spring Initializr bereitgestellten Code. Nunmehr kennen Sie etwas grundlegenden Standardcode, mit dem Sie eine Spring-Anwendung entwickeln können, doch Sie selbst haben noch keine einzige Codezeile geschrieben. Es ist nun an der Zeit, Ihre IDE anzuheizen, die Tastatur zu entstauben und der Taco-Cloud-Anwendung ein paar eigene Codezeilen hinzuzufügen.

1.3 Eine Spring-Anwendung schreiben

Da Sie gerade die ersten Schritte unternehmen, beginnen wir mit einer relativ kleinen Änderung an der Taco-Cloud-Anwendung, die aber viele der Vorzüge von Spring demonstrieren wird. Da Sie gerade beginnen, erscheint es passend, als erstes Feature der Taco-Cloud-Anwendung eine Startseite hinzuzufügen. Dabei erzeugen Sie zwei Codeartefakte:

- Eine Controller-Klasse, die Anfragen an die Startseite verarbeitet
- Eine View-Vorlage, die das Aussehen der Startseite definiert

Und weil Testen wichtig ist, schreiben Sie auch eine einfache Testklasse, um die Startseite zu testen. Aber eins nach dem anderen ... zuerst schreiben wir diesen Controller.

1.3.1 Web-Requests verarbeiten

Spring bringt ein leistungsfähiges Web-Framework namens Spring MVC mit. Den Kern von Spring MVC bildet das Konzept eines *Controllers*. Das ist eine Klasse, die Anfragen und Antworten mit bestimmten Informationen verarbeitet. Bei einer Browser-orientierten

Anwendung reagiert ein Controller, indem er optional Modelldaten füllt und die Anfrage an eine View (Sicht) übergibt, um HTML zu erzeugen, das an den Browser zurückgegeben wird.

In Kapitel 2 werden Sie eine ganze Menge über Spring MVC lernen. Zunächst aber schreiben Sie eine einfache Controller-Klasse, die Anfragen nach dem Root-Pfad (zum Beispiel /) verarbeitet und diese Anfragen an die View der Startseite weiterleitet, ohne irgendwelche Modelldaten zu füllen. Listing 1.4 zeigt die einfache Controller-Klasse.

Listing 1.4 Der Controller für die Startseite

```
package tacos;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller    ◂ Der Controller
public class HomeController {

  @GetMapping("/")    ◂ Behandelt Anfragen für den Root-Pfad /
  public String home() {
    return "home";    ◂ Gibt den View-Namen zurück
  }

}
```

Wie aus dem Listing hervorgeht, ist diese Klasse mit `@Controller` annotiert. Für sich allein genommen bewirkt `@Controller` nicht viel. Diese Annotation soll in erster Linie die Klasse als Komponente für die Komponentensuche identifizieren. Da `HomeController` mit `@Controller` annotiert ist, erkennt die Komponentensuche von Spring den Controller automatisch und erstellt eine Instanz von `HomeController` als Bean im Spring-Anwendungskontext.

Tatsächlich erfüllt eine Handvoll anderer Annotationen (einschließlich `@Component`, `@Service` und `@Repository`) einen ähnlichen Zweck wie `@Controller`. Den `HomeController` hätten Sie genauso gut mit einer der anderen Annotationen versehen können – er hätte trotzdem genauso funktioniert. Die Wahl ist allerdings auf `@Controller` gefallen, weil diese Annotation die Rolle der Komponenten in der Anwendung besser beschreibt.

Die Methode `home()` ist so einfach, wie Controller-Methoden nur sein können. Die Annotation `@GetMapping` zeigt an, dass diese Methode eine HTTP-GET-Anfrage für den Root-Pfad / verarbeiten soll, wenn eine solche Anfrage empfangen wird. Die Methode tut dann nichts weiter, als `home` als `String`-Wert zurückzugeben.

Dieser Wert wird als logischer Name einer View interpretiert. Wie diese View implementiert wird, hängt von mehreren Faktoren ab, da aber Thymeleaf nicht in Ihrem Klassenpfad verzeichnet ist, können Sie diese Vorlage mit Thymeleaf definieren.

Warum Thymeleaf?

Vielleicht fragen Sie sich, warum Sie Thymeleaf als Template-Engine auswählen sollen. Warum nicht JSP? Warum nicht FreeMarker? Warum nicht eine von mehreren anderen Optionen?

Einfach ausgedrückt, musste ich irgendetwas nehmen und ich mag Thymeleaf und bevorzuge es im Allgemeinen gegenüber diesen anderen Optionen. Und selbst wenn JSP wie eine nahe-

liegende Wahl aussehen mag, sind einige Hürden zu überwinden, wenn man JSP mit Spring Boot verwenden möchte. In Kapitel 1 wollte ich allerdings nicht gleich zu tief eintauchen. Warten Sie es ab. Andere Template-Optionen, einschließlich JSP, sehen wir uns in Kapitel 2 an.

Der Template-Name ergibt sich aus dem logischen View-Namen, indem er mit dem Präfix */templates/* und dem Suffix *.html* versehen wird. Der resultierende Pfad für die Vorlage lautet dann */templates/home.html*. Demzufolge müssen Sie die Vorlage in Ihrem Projekt unter */src/main/resources/templates/home.html* platzieren. Erstellen wir nun die Vorlage.

1.3.2 Die View definieren

Um die Startseite einfach zu halten, sollte sie nicht mehr tun, als die Benutzer der Site zu begrüßen. Listing 1.5 zeigt die grundlegende Thymeleaf-Vorlage, die die Taco-Cloud-Startseite definiert.

Listing 1.5 Die Vorlage für die Taco-Cloud-Startseite

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">
  <head>
    <title>Taco Cloud</title>
  </head>

  <body>
    <h1>Welcome to...</h1>
    <img th:src="@{/images/TacoCloud.png}"/>
  </body>
</html>
```

Zu dieser Vorlage gibt es nicht viel zu sagen. Einzig die Codezeile mit dem `<img>`-Tag, die das Taco-Cloud-Logo anzeigt, ist erwähnenswert. Hier wird mit dem Thymeleaf-Attribut `th:src` und einem `@{...}`-Ausdruck über einen kontextrelativen Pfad auf das Bild verwiesen. Abgesehen davon ist es nicht mehr als eine Hello-World-Seite.

Aber befassen wir uns etwas näher mit dem Bild. Ich überlasse es Ihnen, ein Taco-Cloud-Logo nach Ihren Vorstellungen zu definieren. Achten Sie darauf, dass Sie es an der richtigen Stelle innerhalb des Projekts platzieren.

Das Bild wird mit dem kontextrelativen Pfad */images/TacoCloud.png* referenziert. Wie Sie aus der Erläuterung der Projektstruktur wissen, werden statische Inhalte wie zum Beispiel Bilder im Ordner */src/main/resources/static* gespeichert. Deshalb muss auch das Taco-Cloud-Logo innerhalb des Projekts unter */src/main/resources/static/images/TacoCloud.png* untergebracht sein.

Nachdem Sie nun über einen Controller verfügen, um Anfragen an die Startseite zu verarbeiten, und eine View-Template, um die Startseite wiederzugeben, sind Sie fast fertig, um die Anwendung zu starten und in Aktion zu sehen. Doch zuerst sehen wir uns noch an, wie Sie einen Test gegen den Controller schreiben können.

1.3.3 Den Controller testen

Das Testen von Webanwendungen kann knifflig sein, wenn Assertionen in Bezug auf den Inhalt einer HTML-Seite getroffen werden. Erfreulicherweise bringt Spring eine leistungsfähige Testunterstützung mit, die das Testen einer Webanwendung leicht macht.

Für die Startseite werden Sie einen Test schreiben, der in seiner Komplexität mit der Startseite selbst vergleichbar ist. Er führt eine HTTP-GET-Anfrage nach dem Root-Pfad / durch und erwartet ein positives Ergebnis, in dem der View-Name gleich `home` ist und der resultierende Inhalt den Ausdruck »Welcome to...« enthält. Listing 1.6 zeigt, wie sich das anstellen lässt.

Listing 1.6 Ein Test für den Controller der Startseite

```
package tacos;

import static org.hamcrest.Matchers.containsString;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.
                                                                    get;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.
                                                                  content;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.
                                                                  status;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.view;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;
import org.springframework.test.context.junit4.SpringRunner;
import org.springframework.test.web.servlet.MockMvc;

@RunWith(SpringRunner.class)
@WebMvcTest(HomeController.class)    ◂ Web auf HomeController testen
public class HomeControllerTest {

  @Autowired
  private MockMvc mockMvc;    ◂ Injiziert MockMvc

  @Test
  public void testHomePage() throws Exception {
    mockMvc.perform(get("/"))    ◂ Führt GET / aus

      .andExpect(status().isOk())    ◂ Erwartet Sichtname „home“

      .andExpect(view().name("home"))    ◂ Erwartet Sichtname „home“

      .andExpect(content().string(    ◂ Erwartet den String „Welcome to...“
          containsString("Welcome to...")));
  }

}
```

An diesem Test wird Ihnen als Erstes auffallen, dass er sich hinsichtlich der angewendeten Annotationen leicht von der Klasse `TacoCloudApplicationTests` unterscheidet. Statt mit `@SpringBootTest` ist die Klasse `HomeControllerTest` mit `@WebMvcTest` annotiert. Dies ist eine spezielle Testannotation von Spring Boot, die dafür sorgt, dass der Test im Kontext einer Spring-MVC-Anwendung ausgeführt wird. Genauer gesagt veranlasst sie, dass `HomeController` in Spring MVC registriert wird, damit Sie Anfragen gegen ihn auslösen können.

Die Annotation `@WebMvcTest` richtet auch die Spring-Unterstützung für das Testen von Spring MVC ein. Obwohl sich damit ein Server starten ließe, genügt es für unsere Zwecke, die Mechanik von Spring MVC nachzubilden. In die Testklasse wird ein `MockMvc`-Objekt injiziert, damit der Test über das Mock-Objekt läuft.

Die Methode `testHomePage()` definiert den Test, den Sie gegen die Startseite durchführen wollen. Zuerst führt sie mit dem `MockMvc`-Objekt eine `HTTP GET`-Anfrage nach / (den Root-Pfad) aus. Von dieser Anfrage erwartet sie Folgendes:

- Die Antwort sollte den Status HTTP 200 (OK) melden.
- Für die View sollte der logische Name `home` zurückgegeben werden.
- Die gerenderte View sollte den Text »Welcome to...« enthalten.

Wenn mindestens eine dieser Erwartungen nicht erfüllt wird, nachdem das `MockMvc`-Objekt die Anfrage ausgeführt hat, gilt der Test als fehlgeschlagen. Doch Ihr Controller und die View-Vorlage sind so geschrieben, dass diese Erwartungen erfüllt werden, sodass der Test mit Bravour bestanden werden sollte – oder zumindest ein Grün den Erfolg signalisiert.

Nunmehr haben Sie den Controller geschrieben, die View-Vorlage erzeugt und einen bestandenen Test durchgeführt. Es scheint, als ob Sie die Startseite erfolgreich implementiert haben. Doch selbst wenn der Test erfolgreich gelaufen ist, dürfte es zufriedenstellender sein, die Ergebnisse in einem Browser zu sehen. Schließlich werden es die Taco-Cloud-Kunden so sehen. Erstellen Sie also die Anwendung und führen Sie sie aus.

1.3.4 Die Anwendung erstellen und ausführen

So wie es mehrere Möglichkeiten gibt, eine Spring-Anwendung zu initialisieren, gibt es auch mehrere Möglichkeiten, sie auszuführen. Wenn Sie möchten, können Sie im Anhang lesen, welche gängigen Verfahren es gibt, eine Spring-Boot-Anwendung auszuführen.

Da Sie sich für die Spring Tool Suite entschieden haben, um das Projekt zu initialisieren und daran zu arbeiten, steht Ihnen mit dem Spring Boot Dashboard ein praktisches Feature zur Verfügung, das Ihnen hilft, Ihre Anwendung innerhalb der IDE auszuführen. Das Spring Boot Dashboard erscheint als Register, in der Regel im IDE-Fenster links unten. Bild 1.7 zeigt einen kommentierten Screenshot des Spring Boot Dashboards.

Ich möchte hier nicht sämtliche Funktionen erläutern, die das Spring Boot Dashboard beherrscht, die nützlichsten Details sind jedenfalls in Bild 1.7 dargestellt. Wissen sollten Sie jetzt zumindest, wie Sie das Dashboard verwenden, um die Taco-Cloud-Anwendung zu starten. Stellen Sie sicher, dass die Taco-Cloud-Anwendung in der Projektliste markiert ist (in Bild 1.7 ist es die einzige Anwendung), und klicken Sie dann auf die Schaltfläche Start (die Schaltfläche ganz links, die sowohl ein grünes Dreieck als auch ein rotes Quadrat zeigt). Die Anwendung sollte nun sofort starten.

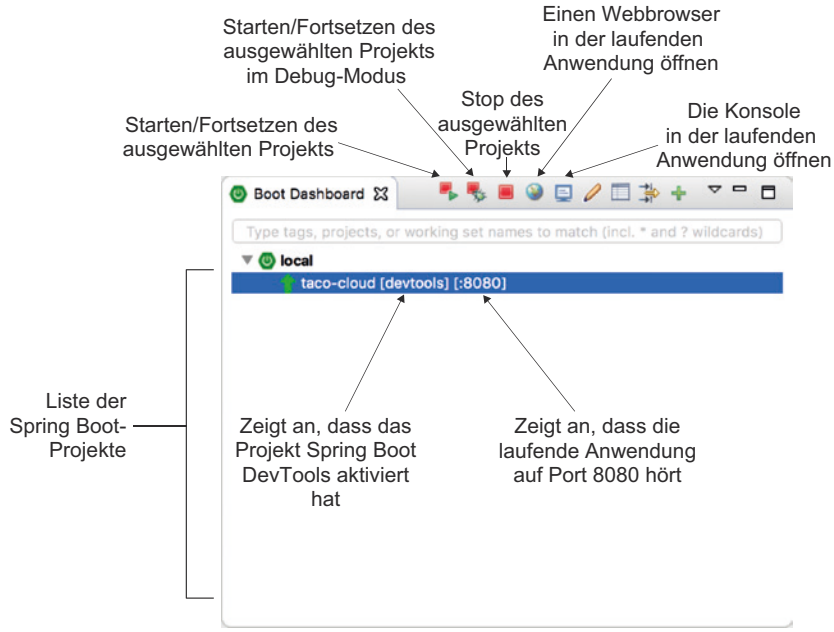

Bild 1.7 Wichtige Elemente des Spring Boot Dashboards

Wenn die Anwendung startet, sehen Sie in der Konsole das mit ASCII-Zeichen gestaltete Spring-Logo vorbeihuschen, gefolgt von einigen Protokolleinträgen, die die Schritte beim Anwendungsstart beschreiben. Bevor die Protokollierung anhält, erscheint ein Protokolleintrag, der besagt, dass Tomcat auf Port 8080 (http) gestartet wurde. Das heißt, Sie sind nun bereit, mit Ihrem Webbrowser auf die Startseite zu gehen, um die Früchte Ihrer Arbeit zu bewundern.

Moment mal. Tomcat gestartet? Wann haben Sie die Anwendung auf Tomcat bereitgestellt?

Spring-Boot-Anwendungen bringen in der Regel alles mit, was sie benötigen, und müssen nicht auf einem Anwendungsserver bereitgestellt werden. Ihre Anwendung haben Sie niemals auf Tomcat bereitgestellt ... Tomcat ist Teil Ihrer Anwendung! (Wie Tomcat Teil Ihrer Anwendung geworden ist, beschreibe ich ausführlich in Abschnitt 1.3.6.)

Nachdem Sie die Anwendung gestartet haben, gehen Sie in Ihrem Webbrowser auf *http://localhost:8080* (oder klicken im Spring Boot Dashboard auf die Erdball-Schaltfläche). Nun sollten Sie etwas wie in Bild 1.8 sehen. Die konkreten Ergebnisse können bei Ihnen abweichen, wenn Sie ein eigenes Logo entworfen haben. Allzu groß sollten die Unterschiede allerdings nicht sein.

Es mag nicht berauschend aussehen, aber dies ist auch kein Buch über Grafikdesign. Das bescheidene Erscheinungsbild der Startseite ist fürs Erste mehr als ausreichend. Und es bietet Ihnen einen sicheren Ausgangspunkt, um Spring kennenzulernen.

Auf ein Modul bin ich bis jetzt noch nicht eingegangen: DevTools. Bei der Initialisierung Ihres Projekts haben Sie es als Abhängigkeit ausgewählt. In der erzeugten *pom.xml*-Datei erscheint es als Abhängigkeit. Und selbst das Spring Boot Dashboard gibt an, dass das Projekt DevTools aktiviert hat. Doch was ist DevTools und was tut es für Sie? Ein kurzer Überblick über die nützlichsten Features von DevTools soll dies beantworten.

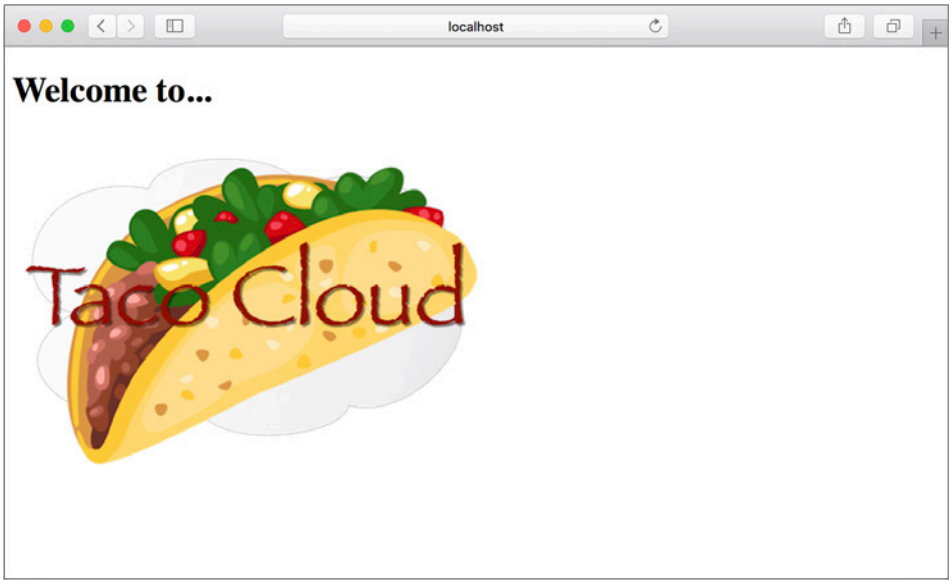

Bild 1.8 Die Startseite von Taco Cloud

1.3.5 Spring Boot DevTools kennenlernen

Wie der Name schon sagt, bietet DevTools den Spring-Entwicklern einige praktische Entwicklungswerkzeuge. Unter anderem:

- Automatischer Neustart der Anwendung, wenn sich der Code ändert
- Automatische Browser-Aktualisierung, wenn sich für den Browser bestimmte Ressourcen (wie zum Beispiel Vorlagen, JavaScript, Stylesheets usw.) ändern
- Automatisches Deaktivieren von Vorlagencaches
- Integrierte H2-Konsole, wenn das H2-Datenbankmanagementsystem verwendet wird

Beachten Sie, dass DevTools weder ein IDE-Plug-in ist noch eine spezifische IDE voraussetzt. Es funktioniert gleichermaßen gut in der Spring Tool Suite, IntelliJ IDEA und NetBeans. Da es zudem nur für Entwicklungszwecke konzipiert ist, deaktiviert es sich selbst, wenn eine Anwendung in einer Produktionsumgebung bereitgestellt wird. (Auf diesen Punkt kommen wir noch in Kapitel 19 zu sprechen, wenn es um das Bereitstellen Ihrer Anwendung geht.) Fürs Erste bleiben wir bei den nützlichsten Features von Spring Boot DevTools und beginnen mit dem automatischen Neustart.

Automatischer Neustart der Anwendung

Mit DevTools als Teil Ihres Projekts sind Sie in der Lage, den Java-Code und die Property-Dateien im Projekt zu verändern und nach einem kurzen Moment die Wirkung dieser Änderungen zu sehen. DevTools startet die Anwendung automatisch neu, sobald es eine Änderung erkannt hat.

Genauer gesagt: Wenn DevTools im Spiel ist, wird die Anwendung in der JVM (Java Virtual Machine) in zwei getrennte Klassenlader geladen. Der eine Klassenlader wird mit Ihrem Java-Code, den Property-Dateien und praktisch allem, was sich im Pfad *src/main/* des Projekts befindet, geladen. Bei diesen Elementen ist davon auszugehen, dass sie sich häufig ändern. Der andere Klassenlader wird mit Abhängigkeitsbibliotheken geladen, die sich höchstwahrscheinlich nicht oft ändern.

Wenn DevTools eine Änderung erkennt, lädt es nur den Klassenlader erneut, der Ihren Projektcode enthält, und startet den Spring-Anwendungskontext neu, lässt aber den anderen Klassenlader und die JVM unverändert. Diese Strategie ist zwar nicht weltbewegend, spart aber etwas Zeit beim Starten der Anwendung.

Nachteilig bei dieser Strategie ist, dass Änderungen an Abhängigkeiten in automatischen Neustarts nicht verfügbar sind. Das hängt damit zusammen, dass der Klassenlader, der die Abhängigkeitsbibliotheken enthält, nicht automatisch neu geladen wird. Jedes Mal, wenn Sie eine Abhängigkeit in Ihrer Build-Spezifikation hinzufügen, ändern oder entfernen, müssen Sie also einen harten Neustart der Anwendung durchführen, damit diese Änderungen wirksam werden.

Automatische Browser-Aktualisierung und Template-Cache-Deaktivierung

Standardmäßig werden Template-Optionen wie zum Beispiel Thymeleaf und FreeMarker so konfiguriert, dass sie die Ergebnisse des Template-Parsings zwischenspeichern, damit diese Templates nicht bei jeder von ihnen bedienten Anfrage erneut geparst werden müssen. In einer Produktionsumgebung ist das in Ordnung, da sich dadurch ein kleiner Performancegewinn ergibt.

Zur Entwicklungszeit sind zwischengespeicherte Templates jedoch nicht so willkommen. Es ist dann nämlich nicht möglich, Änderungen an den Vorlagen vorzunehmen, während die Anwendung läuft, und sich die Ergebnisse nach einer Browser-Aktualisierung anzusehen. Selbst wenn Sie die Vorlage ändern, wird die im Cache befindliche Vorlage weiterhin verwendet, bis Sie die Anwendung neu starten.

Um diesem Problem zu begegnen, deaktiviert DevTools automatisch das Zwischenspeichern von Templates. Somit können Sie Ihre Vorlagen nach Belieben ändern und sind nur eine Browser-Aktualisierung davon entfernt, die Ergebnisse zu begutachten.

Doch wenn es Ihnen geht wie mir, ist Ihnen sogar das Klicken auf die Aktualisierungsschaltfläche des Browsers lästig. Es wäre viel komfortabler, wenn Sie die Änderungen ausführen und die Ergebnisse im Browser sofort erleben könnten. Erfreulicherweise bringt DevTools für diejenigen, die zu faul sind, auf Aktualisierung zu klicken, etwas Spezielles mit.

Wenn DevTools läuft, aktiviert es automatisch einen LiveReload-Server (http://livereload.com/) für Ihre Anwendung. Der LiveReload-Server an sich ist nicht sehr nützlich. Doch in Verbindung mit einem entsprechenden LiveReload-Browser-Plug-in sorgt er dafür, dass Ihr Browser automatisch aktualisiert wird, wenn Änderungen an Vorlagen, Bildern, Stylesheets, JavaScript usw. stattfinden – praktisch bei allem, das letztlich Ihrem Browser zugeführt wird.

LiveReload verfügt über Browser-Plug-ins für Google Chrome, Safari und Firefox. (Sorry, Fans von Internet Explorer und Edge.) Unter http://livereload.com/extensions/ erfahren Sie, wie Sie LiveReload für Ihren Browser installieren können.

Integrierte H2-Konsole

Momentan verwendet Ihr Projekt noch keine Datenbank. In Kapitel 3 ändert sich das. Wenn Sie sich in der Entwicklungsphase für die H2-Datenbank entscheiden, aktiviert DevTools automatisch eine H2-Konsole, auf die Sie von Ihrem Webbrowser aus zugreifen können. In Ihrem Webbrowser müssen Sie lediglich die Adresse http://localhost:8080/h2-console eingeben, um Einblick in die Daten zu bekommen, mit denen Ihre Anwendung arbeitet.

Nunmehr haben Sie eine vollständige, wenn auch einfache, Spring-Anwendung geschrieben. Im Verlauf dieses Buches erweitern Sie diese Anwendung. Zunächst aber bietet sich hier ein Rückblick auf das an, was Sie erreicht haben, und welche Rolle Spring dabei gespielt hat.

1.3.6 Rückblick

Denken Sie zurück, wie Sie zu diesem Punkt gekommen sind. Kurz gesagt sind es folgende Schritte, in denen Sie Ihre Spring-basierte Taco-Cloud-Anwendung erstellt haben:

- Sie haben mit Spring Initializr eine erste Projektstruktur angelegt.
- Sie haben eine Controller-Klasse geschrieben, um Anfragen an die Startseite zu verarbeiten.
- Sie haben ein View-Template definiert, um die Startseite zu rendern.
- Sie haben eine einfache Testklasse geschrieben, um Ihre Arbeit zu überprüfen.

Das sieht doch ganz einfach aus, oder? Mit Ausnahme des ersten Schritts zum Hochfahren des Projekts ist jede Aktion, die Sie unternommen haben, genau auf das Ziel fokussiert, eine Startseite zu schaffen.

Tatsächlich ist fast jede Codezeile, die Sie geschrieben haben, auf dieses Ziel ausgerichtet. Ohne die `import`-Anweisungen von Java zähle ich nur zwei Spring-spezifische Codezeilen in der Controller-Klasse und keine solchen Zeilen in der View-Vorlage. Und obwohl der Großteil der Testklasse auf die Testunterstützung von Spring zurückgreift, scheint sie im Rahmen eines Tests etwas weniger invasiv zu sein.

Das ist ein wichtiger Vorteil der Entwicklung mit Spring. Sie können sich auf den Code konzentrieren, der den Erfordernissen einer Anwendung entspricht, anstatt die Anforderungen eines Frameworks zu erfüllen. Auch wenn Sie gelegentlich etwas Framework-spezifischen Code schreiben müssen, macht er normalerweise nur einen kleinen Bruchteil Ihrer Codebasis aus. Wie bereits gesagt, kann man Spring (mit Spring Boot) als Framework-loses Framework betrachten.

Wie funktioniert das überhaupt? Wie stellt Spring hinter den Kulissen sicher, dass die Bedürfnisse Ihrer Anwendung erfüllt werden? Um zu verstehen, was Spring macht, sehen wir uns zunächst die Build-Spezifikation an.

In der Datei *pom.xml* haben Sie eine Abhängigkeit von den Startern `Web` und `Thymeleaf` deklariert. Diese beiden Abhängigkeiten haben transitiv eine Handvoll anderer Abhängigkeiten mit sich gebracht, unter anderem:

- Das MVC-Framework von Spring
- Den eingebetteten Tomcat-Webserver
- Thymeleaf und den Thymeleaf-Layout-Dialekt

Dazu gehört auch die Autokonfigurationsbibliothek von Spring Boot. Wenn die Anwendung startet, erkennt die Autokonfiguration von Spring Boot diese Bibliotheken und konfiguriert automatisch

- die Beans im Spring-Anwendungskontext, um Spring MVC zu aktivieren,
- den eingebetteten Tomcat-Server im Spring-Anwendungskontext,
- einen Thymeleaf-View-Resolver für das Rendern der Spring MVC-Views mit Thymeleaf-Vorlagen.

Kurz gesagt erledigt die Autokonfiguration die gesamte Routinearbeit, sodass Sie sich darauf konzentrieren können, den Code zu schreiben, der die Funktionalität Ihrer Anwendung realisiert. Das ist ein ziemlich cooles Arrangement, wenn Sie mich fragen!

Ihre Spring-Tour hat eben erst begonnen. Die Taco-Cloud-Anwendung hat gerade einmal an der Oberfläche gekratzt, was Spring zu bieten hat. Vor dem nächsten Schritt werfen wir einen Blick auf die Spring-Landschaft und sehen uns an, welche Sehenswürdigkeiten Ihnen auf Ihrer Tour begegnen werden.

1.4 Die Spring-Landschaft im Überblick

Um eine Vorstellung von der Spring-Landschaft zu bekommen, brauchen Sie sich nur die umfangreiche Liste der Optionen auf dem Webformular von Spring Initializr anzuschauen. Da hier über hundert mögliche Abhängigkeiten aufgelistet sind, möchte ich sie weder alle aufführen noch einen Screenshot beisteuern. Doch Sie sollten auf jeden Fall einen Blick darauf werfen. Einige herausragende Optionen werde ich aber erwähnen.

1.4.1 Der Core des Spring Frameworks

Wie Sie sich sicherlich denken können, bildet das Spring Framework das Fundament für alles andere im Spring-Universum. Es stellt den Core-Container und das Dependency Injection Framework bereit. Es bietet aber auch einige andere wesentliche Funktionen.

Dazu gehört Spring MVC, das Web-Framework von Spring. Mit Spring MVC haben Sie bereits eine Controller-Klasse geschrieben, um Web-Anfragen zu verarbeiten. Allerdings können Sie mit Spring MVC auch REST APIs erstellen, die Nicht-HTML-Ausgaben erzeugen. Kapitel 2 befasst sich eingehend mit Spring MVC und Kapitel 6 erläutert, wie Sie damit REST APIs erzeugen.

Das Core Spring Framework bietet auch elementare Unterstützung für Datenpersistenz, insbesondere Template-basierte JDBC-Unterstützung. Wie Sie die Klasse `JdbcTemplate` einsetzen, erfahren Sie in Kapitel 3.

In der neuesten Version von Spring (5.0.8) ist die Unterstützung für reaktive Programmierung hinzugekommen, einschließlich eines neuen reaktiven Web-Frameworks namens

Spring WebFlux, das stark an Spring MVC angelehnt ist. Das reaktive Programmiermodell von Spring lernen Sie in Teil 3 kennen. Kapitel 10 befasst sich speziell mit Spring WebFlux.

1.4.2 Spring Boot

Sie haben bereits viele Vorteile von Spring Boot kennengelernt, einschließlich der Starterabhängigkeiten und der Autokonfiguration. In diesem Buch werden wir so viel Spring Boot wie möglich verwenden und jede Form von expliziter Konfiguration vermeiden, außer wenn sie absolut notwendig ist. Neben Starterabhängigkeiten und Autokonfiguration bietet Spring Boot aber auch eine Handvoll anderer nützlicher Features:

- Der Aktuator gewährt zur Laufzeit Einblick in die inneren Abläufe einer Anwendung. Dazu gehören Metriken, Thread-Dump-Informationen, Anwendungsstatus und Umgebungseigenschaften, die für die Anwendung verfügbar sind.
- Flexible Spezifikation von Umgebungseigenschaften.
- Zusätzliche Testunterstützung, die auf der Testhilfe des Core Frameworks aufsetzt.

Darüber hinaus bietet Spring Boot ein alternatives Programmiermodell, das auf Groovy-Skripten basiert und als Spring Boot CLI (Command-Line Interface, Befehlszeilenoberfläche) bezeichnet wird. Mit der Spring Boot CLI können Sie ganze Anwendungen als Sammlung von Groovy-Skripten programmieren und sie von der Befehlszeile ausführen. Auf die Spring Boot CLI gehen wir hier nicht näher ein, geben aber entsprechende Hinweise, wenn es zweckmäßig ist.

Spring Boot ist zu einem solch integralen Bestandteil der Spring-Entwicklung geworden, dass ich mir gar nicht mehr vorstellen kann, eine Anwendung ohne Spring Boot zu entwickeln. Folglich ist dieses Buch unter dem Blickwinkel von Spring Boot geschrieben und vielleicht ertappen Sie mich dabei, dass ich das Wort *Spring* verwende, wenn ich mich auf etwas beziehe, was *Spring Boot* zuzuordnen ist.

1.4.3 Spring Data

Obwohl das Core Spring Framework eine einfache Unterstützung für Datenpersistenz mitbringt, bietet Spring Data etwas ganz Erstaunliches: die Fähigkeit, die Datenspeicher einer Anwendung als einfache Java-Interfaces zu definieren und bei der Definition von Methoden eine Konvention zur Benennung anzuwenden, aus der hervorgeht, wie Daten gespeichert und abgerufen werden.

Darüber hinaus ist Spring Data in der Lage, mit verschiedenen Arten von Datenbanken zu arbeiten, darunter relationale (JPA), dokumentenorientierte (Mongo), auf Graphen basierende (Neo4j) und andere. Mithilfe von Spring Data erstellen Sie in Kapitel 3 Datenspeicher für die Taco-Cloud-Anwendung.

1.4.4 Spring Security

Anwendungssicherheit ist schon immer ein wichtiges Thema gewesen und jeden Tag scheint es wichtiger zu werden. Erfreulicherweise besitzt Spring in Spring Security ein robustes Sicherheitsframework.

Spring Security ist auf ein breites Spektrum von Sicherheitsbedürfnissen ausgelegt, einschließlich Authentifizierung, Autorisierung und API-Sicherheit. Da aber der Bereich von Spring Security zu groß ist, um ihn adäquat in diesem Buch darzustellen, beschränken wir uns in den Kapiteln 4 und 12 auf die gängigsten Einsatzfälle.

1.4.5 Spring Integration und Spring Batch

Irgendwann müssen die meisten Anwendungen mit anderen Anwendungen oder sogar mit anderen Komponenten derselben Anwendung verflochten werden. Dabei haben sich mehrere Muster herausgebildet, um diesen Anforderungen gerecht zu werden. Spring Integration und Spring Batch bieten die Implementierung dieser Muster für Spring-basierte Anwendungen.

Spring Integration realisiert die Echtzeitintegration, die Daten verarbeitet, sobald sie verfügbar sind. Im Gegensatz dazu geht es bei Spring Batch um die stapelorientierte Integration, bei der Daten eine Zeitlang gesammelt werden dürfen, bis ein Auslöser (etwa ein Zeitgeber) signalisiert, dass der Datenstapel nun verarbeitet werden sollte. Spring Batch und Spring Integration sind Themen von Kapitel 9.

1.4.6 Spring Cloud

Während ich dies schreibe, tritt die Welt der Anwendungsentwicklung in eine neue Ära ein, in der wir unsere Anwendungen nicht mehr als einzelne monolithische Bereitstellungseinheiten entwickeln und stattdessen Anwendungen aus mehreren einzelnen Bereitstellungseinheiten – den sogenannten *Microservices* – zusammenstellen.

Microservices sind ein heißes Thema, das verschiedene praktische Entwicklungs- und Laufzeitprobleme angeht. Dabei treten allerdings auch eigene Herausforderungen zutage. Diesen Herausforderungen stellt sich Spring Cloud, eine Sammlung von Projekten für die Entwicklung Cloud-nativer Anwendungen mit Spring.

Spring Cloud deckt ein weites Gebiet ab und es wäre unmöglich, in diesem Buch auf alles einzugehen. Einige der gängigsten Komponenten von Spring Cloud lernen Sie in den Kapiteln 13, 14 und 15 kennen. Für eine umfassendere Diskussion von Spring Cloud empfehle ich „*Spring Microservices in Action*“ von John Carnell (Manning, 2017, www.manning.com/books/spring-microservices-in-action).

1.5 Zusammenfassung

- Spring soll es Entwicklern unter anderem erleichtern, Webanwendungen zu erstellen, mit Datenbanken zu arbeiten, Anwendungen zu sichern und Microservices zu implementieren.
- Spring Boot setzt auf Spring auf, um Spring mit vereinfachter Abhängigkeitsverwaltung, automatischer Konfiguration und Laufzeitanalysen noch einfacher zu machen.
- Spring-Anwendungen lassen sich mit dem Spring Initializr initialisieren. Dieser ist webbasiert und wird in den meisten Java-Entwicklungsumgebungen nativ unterstützt.
- Die allgemein als Beans bezeichneten Komponenten lassen sich in einem Spring-Anwendungskontext explizit mit Java oder XML deklarieren, durch Komponentensuche auffinden oder mit der automatischen Konfiguration von Spring Boot automatisch konfigurieren.

2 Webanwendungen entwickeln

Die Themen dieses Kapitels:

- Modelldaten im Browser darstellen
- Formulareingaben verarbeiten und validieren
- Eine View-Template-Bibliothek auswählen

Der erste Eindruck ist immer der entscheidende. Das äußere Erscheinungsbild hat die Entscheidung des Hauskäufers bereits geprägt, lange bevor er durch die Tür tritt. Die kirschrote Lackierung eines Autos erregt mehr Aufmerksamkeit als das, was unter der Motorhaube werkelt. Und die Literatur ist voll von Geschichten über Liebe auf den ersten Blick. Was drin steckt, ist schon wichtig, doch was außen ist – was zuerst wahrgenommen wird – ist nicht weniger wichtig.

Die Anwendungen, die Sie mit Spring erstellen, erledigen alle möglichen Aufgaben, unter anderem Daten verarbeiten, Informationen aus einer Datenbank lesen und mit anderen Anwendungen interagieren. Doch der erste Eindruck, den die Benutzer von Ihrer Anwendung bekommen, stammt von der Benutzeroberfläche, dem User Interface (kurz UI). Und in vielen Anwendungen ist diese Benutzeroberfläche eine Webanwendung, die in einem Browser dargestellt wird.

In Kapitel 1 haben Sie Ihren ersten Spring-MVC-Controller erstellt, um die Startseite Ihrer Anwendung anzuzeigen. Doch Spring MVC kann weit mehr, als lediglich statische Inhalte anzuzeigen. In diesem Kapitel entwickeln Sie das erste größere Stück Funktionalität Ihrer Taco-Cloud-Anwendung – die Fähigkeit, eigene Tacos zu kreieren. Dabei dringen Sie tiefer in Spring MVC ein und lernen, wie Sie Modelldaten anzeigen und Formulareingaben verarbeiten.

2.1 Informationen anzeigen

Grundsätzlich bietet Taco Cloud einen Ort, an dem Sie Tacos online bestellen können. Darüber hinaus aber möchte Taco Cloud seine Kunden dazu ermuntern, ihre kreative Seite zu zeigen und Tacos aus einer reichen Palette von Zutaten zu gestalten.

Demzufolge benötigt die Taco-Cloud-Webanwendung eine Seite mit einer Liste der Zutaten, aus der die Taco-Künstler auswählen können. Die Zutatenauswahl kann sich jederzeit ändern und sollte deshalb nicht in einer HTML-Seite fest codiert werden. Vielmehr sollte die Liste der verfügbaren Zutaten aus einer Datenbank abgerufen und an die Seite übergeben werden, die dem Kunden angezeigt wird.

In einer Spring-Webanwendung hat der Controller die Aufgabe, Daten abzurufen und zu verarbeiten. Und es ist die Aufgabe der View, diese Daten als HTML zu rendern, das im Browser angezeigt wird. Somit erstellen Sie die folgenden Komponenten, um die Seite für die Taco-Kreation zu unterstützen:

- Eine Domänenklasse, die die Eigenschaften einer Taco-Zutat definiert
- Eine Spring MVC-Controller-Klasse, die Informationen zu den Zutaten abruft und an die View übergibt
- Eine View-Template, die eine Liste von Zutaten im Browser des Benutzers rendert

Bild 2.1 veranschaulicht die Beziehung zwischen diesen Komponenten.

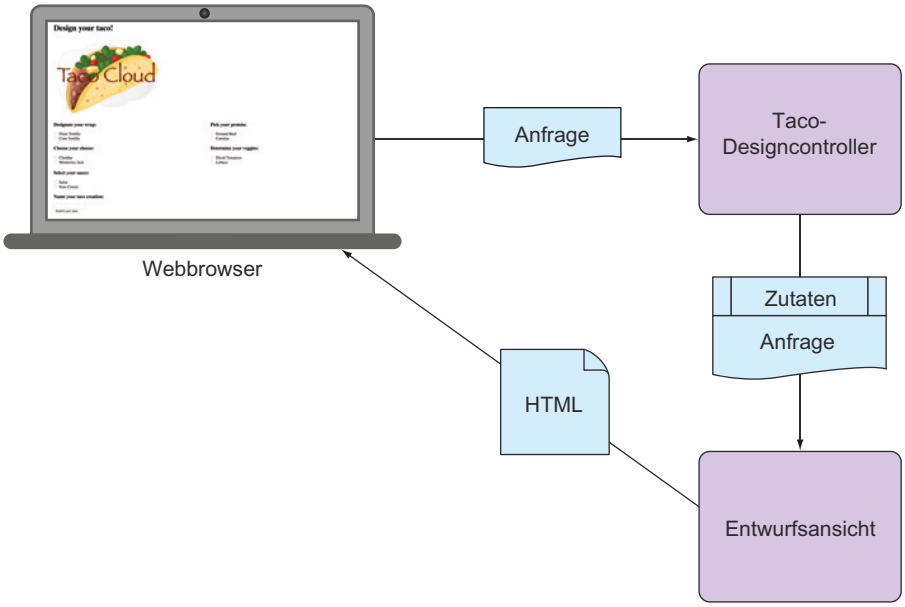

Bild 2.1 Ein typischer Ablauf einer Spring-MVC-Anfrage

Da sich dieses Kapitel auf das Webframework von Spring konzentriert, verschieben wir alle Fragen im Zusammenhang mit Datenbanken auf Kapitel 3. Momentan ist der Controller allein dafür zuständig, die Zutaten der View bereitzustellen. In Kapitel 3 überarbeiten Sie den Controller, um mit einem Repository zusammenzuarbeiten, das die Daten für die Zutaten aus einer Datenbank abruft.

Bevor Sie den Controller und die View schreiben, wollen wir den Domänentyp ausarbeiten, der eine Zutat repräsentiert. Das schafft eine gute Grundlage, auf der Sie Ihre Webkomponenten entwickeln können.

2.1.1 Die Domäne einrichten

Die Domäne einer Anwendung ist der Bereich, auf den die Anwendung abzielt – die Ideen und Konzepte, die das Verständnis der Anwendung beeinflussen.[1] In der Taco-Cloud-Anwendung umfasst die Domäne solche Objekte wie Taco-Kreationen, die Zutaten, aus denen diese Kreationen komponiert werden, Kunden und Taco-Bestellungen, die der Kunde platziert. Um loszulegen, konzentrieren wir uns auf Taco-Zutaten.

In unserer Domäne sind Taco-Zutaten ziemlich einfache Objekte. Jedes hat einen Namen und einen Typ, sodass es sich visuell kategorisieren lässt (Proteine, Käse, Soßen usw.). Außerdem erhält jedes Objekt eine ID, über die es leicht und eindeutig referenziert werden kann. Die Klasse `Ingredient` in Listing 2.1 definiert das benötigte Domänenobjekt.

Listing 2.1 Die Taco-Zutaten definieren

```
package tacos;

import lombok.Data;
import lombok.RequiredArgsConstructor;

@Data
@RequiredArgsConstructor
public class Ingredient {

  private final String id;
  private final String name;
  private final Type type;

  public static enum Type {
    WRAP, PROTEIN, VEGGIES, CHEESE, SAUCE
  }

}
```

Wie das Listing zeigt, handelt es sich um eine ganz normale Java-Domänenklasse. Sie definiert die drei Eigenschaften, die erforderlich sind, um eine Zutat zu beschreiben. Das vielleicht Ungewöhnlichste an der Klasse `Ingredient` in Listing 2.1 ist, dass ihr scheinbar die üblichen Getter- und Setter-Methoden fehlen, ganz abgesehen von nützlichen Methoden wie `equals()`, `hashCode()`, `toString()` und anderen.

Hinweis

Die Verwendung von `@RequiredArgsConstructor` ist nicht unbedingt notwendig, weil `@Data` die Generierung eines Konstruktors für erforderliche Argumente impliziert. (Unabhängig davon schadet es nicht, die Annotation `@RequiredArgsConstructor` anzugeben.)

[1] Für eine tiefer gehende Abhandlung zu Anwendungsdomänen empfehle ich das Buch *Domain Driven Design* von Eric Evans (Addison-Wesley Professional, 2003).

Im Listing sehen Sie die Methoden nicht, zum einen, um Platz zu sparen, zum anderen aber auch, weil Sie auf eine erstaunliche Bibliothek namens Lombok zurückgreifen, um die betreffenden Methoden zur Laufzeit automatisch zu generieren. In der Tat wird die Annotation `@Data` auf der Klassenebene durch Lombok bereitgestellt. Lombok generiert daraufhin sämtliche fehlende Methoden sowie einen Konstruktor, der alle `final`-Eigenschaften als Argumente übernimmt. Mithilfe von Lombok können Sie den Code für `Ingredient` schlank und kompakt halten.

Lombok ist zwar keine Spring-Bibliothek, aber so unglaublich nützlich, dass es meiner Ansicht nach sehr schwer ist, Anwendungen ohne dieses Framework zu entwickeln. Und es ist meine Rettung, wenn ich Codebeispiele in einem Buch kurz und prägnant halten muss.

Um Lombok zu verwenden, müssen Sie das Framework als Abhängigkeit in Ihrem Projekt hinzufügen. Wenn Sie mit der Spring Tool Suite arbeiten, genügt es, mit der rechten Maustaste auf die Datei *pom.xml* zu klicken und im Kontextmenü Spring und dann Edit Starters auszuwählen. Es erscheint die gleiche Auswahl von Abhängigkeiten, die Sie in Kapitel 1 (Bild 1.4) bekommen haben, und Sie haben so die Möglichkeit, Ihre ausgewählten Abhängigkeiten zu erweitern oder zu ändern. Suchen Sie den Eintrag für Lombok, aktivieren Sie das Kontrollkästchen und klicken Sie auf OK. Die Spring Tool Suite fügt das Framework automatisch Ihrer Build-Spezifikation hinzu.

Alternativ können Sie es mit dem folgenden Eintrag in *pom.xml* manuell hinzufügen:

```
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <optional>true</optional>
</dependency>
```

Diese Abhängigkeit stellt Ihnen Lombok-Annotationen (wie zum Beispiel `@Data`) zur Entwicklungszeit und automatische Generierung von Methoden zur Laufzeit bereit. Allerdings müssen Sie auch Lombok als Erweiterung in Ihre IDE hinzufügen. Andernfalls beschwert sich die IDE mit Fehlermeldungen über fehlende Methoden und `final`-Eigenschaften, die nicht gesetzt werden. Wie Sie Lombok in der IDE Ihrer Wahl installieren, erfahren Sie unter https://projectlombok.org/.

Lombok werden Sie als sehr nützlich empfinden, wohlwissend, dass es optional ist. Sie benötigen das Framework nicht, um Spring-Anwendungen zu entwickeln. Möchten Sie es lieber nicht verwenden, steht es Ihnen frei, die fehlenden Methoden selbst einzutippen. Auf geht's ... ich werde warten. Wenn Sie fertig sind, fügen Sie noch einige Controller hinzu, um in Ihrer Anwendung Webanfragen zu verarbeiten.

2.1.2 Eine Controller-Klasse erstellen

Controller sind die Hauptakteure im MVC Framework von Spring. Ihre Aufgabe besteht hauptsächlich darin, HTTP-Anfragen zu verarbeiten und eine Anfrage entweder an eine View zu übergeben, um HTML zu rendern (das der Browser anzeigt), oder die Daten direkt in den Körper einer Antwort zu schreiben (RESTful). In diesem Kapitel konzentrieren wir uns auf

solche Controller, die mithilfe von Views die Inhalte für Webbrowser erzeugen. In Kapitel 6 schreiben Sie dann auch Controller, die Anfragen in einer REST API verarbeiten.

Für die Taco-Cloud-Anwendung brauchen Sie einen einfachen Controller, der folgende Funktionen beherrscht:

- HTTP-GET-Anfragen verarbeiten, bei denen der Anfragepfad `/design` lautet
- Eine Liste von Zutaten erstellen
- Die Anfrage und die Daten für die Zutaten an eine View-Template übergeben, damit sie als HTML gerendert und an den anfragenden Webbrowser geschickt werden

Die folgende Klasse `DesignTacoController` in Listing 2.2 erfüllt diese Anforderungen.

Listing 2.2 Die Anfänge einer Spring-Controller-Klasse

```
package tacos.web;

import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

import javax.validation.Valid;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.Errors;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;

import lombok.extern.slf4j.Slf4j;
import tacos.Taco;
import tacos.Ingredient;
import tacos.Ingredient.Type;

@Slf4j
@Controller
@RequestMapping("/design")
public class DesignTacoController {
  @GetMapping
  public String showDesignForm(Model model) {
    List<Ingredient> ingredients = Arrays.asList(
      new Ingredient("FLTO", "Flour Tortilla", Type.WRAP),
      new Ingredient("COTO", "Corn Tortilla", Type.WRAP),
      new Ingredient("GRBF", "Ground Beef", Type.PROTEIN),
      new Ingredient("CARN", "Carnitas", Type.PROTEIN),
      new Ingredient("TMTO", "Diced Tomatoes", Type.VEGGIES),
      new Ingredient("LETC", "Lettuce", Type.VEGGIES),
      new Ingredient("CHED", "Cheddar", Type.CHEESE),
      new Ingredient("JACK", "Monterrey Jack", Type.CHEESE),
      new Ingredient("SLSA", "Salsa", Type.SAUCE),
      new Ingredient("SRCR", "Sour Cream", Type.SAUCE)
    );
    Type[] types = Ingredient.Type.values();
```