Course Overview

Course Overview

Hi everyone. My name's Dustin Schultz, and welcome to the course, Spring: The Big Picture. I've been writing and developing applications with Spring for more than 10 years. It's a great and fun technology to work with. This course is a short introduction to Spring and provides a high-level, ten-thousand-foot view of all things that are Spring. Some of the major topics we'll cover in this course are, what is Spring, what is the Spring Framework, what is Spring Boot, and why would I want to use Spring? By the end of this course, you'll have a broad understanding of Spring, what it provides, and what it's used for. I hope you'll join me on this journey at Pluralsight to learn Spring with the course, Spring: The Big Picture.

What Is Spring?

Introduction

Hi. I'm Dustin Schultz, and welcome to Spring: The Big Picture. Spring is one of the most popular, if not the most popular, technologies for Java development. It's used by thousands and thousands of developers and companies worldwide, and the goal of this course is to provide you with a high-level overview of Spring. A ten-thousand-foot view if you will. So what can you expect from this course? Let's talk about expectations. Where possible, we're going to avoid the details and really try to keep things simple. I'll give you just enough detail to help you learn and understand the content, but really nothing beyond that. And this course is particularly short and is probably the exact opposite of what you would call a technology deep dive. Our goal here is to cover breadth and not depth, and that means we're going to cover a range of topics, but we'll really only scratch the surface on each of the particular topics. We will be seeing small, but easy code snippets, and don't worry, it's not that big of a deal, but it would practically be impossible to describe the benefits and usefulness of Spring without them. Again, if you don't know Java, or you don't regularly code, seriously don't worry. I'll keep things very simple and walk you through step by step each of the examples. Lastly, don't be surprised to see some hand waving. And by that I mean, don't be surprised if I gloss over some details, or I'm intentionally vague in certain areas. Remember, the goal of this course is to provide you with an overview of Spring. We want to see the whole forest and not each of the individual trees. The intended audience of this course is particularly large and broad, but generally falls into two categories, individuals working in a technical, but non-development role, and individuals working in a development role who are unfamiliar with Spring. So maybe you've heard of Spring or you've heard others talking about it and you've always wondered what it is, or maybe you're a developer and you're currently using a different technology stack and you need to learn Spring, or you're just generally curious what it is. If any of those sound like you, then this course is for you. Alright, enough with the introductions, let's get started.

What Is Spring?

Surely you're wondering what exactly is Spring, and to be honest, the answer is a bit confusing. In a technology context when someone says Spring, you can be sure that they're definitely not talking about the season, but they could be referring to one of a number of different things. Spring could mean the Spring Framework, Spring Boot, Spring Data, Spring Cloud, Spring Batch, and a whole lot more; however, most often when people say Spring, they're actually referring to the entire family of projects, or the Spring ecosystem. Let's take a closer look at what the Spring family, or Spring ecosystem, consists of.

Meet the Spring Family

The Spring family all began with the creation of the Spring Framework, which was built largely in response to the complexity of developing applications using Java's enterprise framework called J2EE, it's now called Java EE, and the Spring Framework aimed at removing the complexity and helped to make things like web development and data access easier for developers to implement. It also aimed at reducing boilerplate code, and if you're not familiar with what that is, it's repetitive code that's often needed as part of the logic, but tends to clutter the application and draw focus away from the main logic. The Spring Framework is still heavily used today, and it's the foundation upon which everything else is built. The success of the Spring Framework led to the creation of several Spring projects that were built on top of the Spring Framework, but tailored to specific needs or domains. For example, the Spring Security project was created to remove the complexity and challenges with securing a Java application, and another project, like the Spring Data project, was created to further simplify data access in Java applications. The Spring Framework and Spring projects continued to evolve and thrive, eventually resulting in the creation of a particularly impactful project called Spring Boot. Spring Boot was a real game changer. It provided a new and drastically faster way of building Spring-based applications. Up until Spring Boot, building a Spring-based application had always involved a lot of choice making, configuration, and a cumbersome deployment model. Spring Boot removed all of that by taking an opinionated view of building Spring-based applications including sensible defaults for library choices and configuration, and adding smarts for auto-detecting and auto-configuring other common configurations. It also massively simplified the deployment process and made running a Spring-based application as simple as running a single command. Lastly came the Spring Cloud project. Spring Cloud was built on top of Spring Boot and simplified the development of applications that make use of distributed architectures, such as microservice architectures, and these type of architectures often have many common patterns that are implemented, such as service discovery and distributed configuration, and Spring Cloud helped to make it easier for developers to build applications that use those patterns. Now that we've been introduced to the Spring family, let's take a look at why we would want to use it.

Why Spring?

So why Spring? And oftentimes you'll hear people asking, why would I want to use Spring? And that's probably something you may be even thinking for yourself right now. And one of the common answers you'll hear is that it's a good alternative to Java EE, and while that was true in the beginning, today Spring is so much more than just an alternative to Java EE. In fact, it's actually complimentary to Java EE and makes use of several of the standard specifications like the Java Persistence API. The question of why you would want to use Spring is a completely valid and reasonable question that actually has a very simple answer, and that's that creating software can be hard, seriously, and Spring helps make it easier. You'll notice that regardless of the project within the Spring family, they all share a common theme, and that's to make developing Java applications easier. Let's talk about some other reasons why Spring is a good choice. First, Spring is really great at being flexible, modular, and especially backwards compatible. So Spring is far from an all-or-nothing type of choice. In fact, you're given the freedom to pick and choose whichever parts of Spring make the most sense for your project. And what's really great about that is that you can be sure that the investment you're making with Spring won't be wasted when a new version is released. Spring also has a very large and active community, and in my opinion this is often one of the things that's overlooked and taken for granted when assessing technologies, but it makes a huge difference. With Spring, you can be sure that there's a wealth of knowledge that already exists out there, and if you run into a problem, it's likely that someone else has already ran into that same problem, and they can probably help you through it. And lastly, what makes Spring a good choice is that it's continually innovating and evolving. Being an open-source project but fully backed by a company called Pivotal allows Spring to remain healthy and alive, while at the same time pushing the bounds of enterprise Java development. It's not uncommon for Java's standards or specifications to be influenced or based on ideas that originated within Spring.

Summary

In this module, we saw the what and why of Spring. We got to understand what people mean when they say Spring and how they're referring to the Spring family, or the Spring ecosystem. We also talked about why Spring was created in the first place and how it evolved over time to where it's at today. And lastly, we discussed why we would want to use Spring, mainly that Spring strives to make building Java applications easier.

Getting to Know Spring with Spring Boot

Spring Boot Makes Spring Both Quick and Easy

Hi. My name's Dustin Schultz, and in this module of Spring: The Big Picture course, we'll learn about one of the key projects in the Spring family called Spring Boot. We already had a very brief introduction to Spring Boot in the previous module, so we'll use this module to take a slightly deeper look at Spring Boot. Again, remember that we're keeping things high level, so we don't want to be digging too deep here. We certainly won't be learning how to use Spring Boot or doing any sort of in-depth demos. Instead, we'll be focused on what Spring Boot is, and what it's about, and what it provides. We want to use Spring Boot as our first introduction to Spring and sort of get a flavor for what it's like. Learning any new technology can be both confusing and frustrating. It's really, really hard to know where to start, and you can quickly become overwhelmed. Thankfully, Spring Boot makes learning Spring both quick and easy. Developers can jump right in and get started without having to fully understand everything that's behind the scenes. Then as they get more advanced and knowledgeable, they can circle their way back and learn more about the details. I want to be absolutely clear though, easy does not mean lack of features. It's somewhat common for frameworks to boast about how great and easy their technology is, only to be disappointed when it's used in real-world scenarios and falls flat on its face. Spring Boot on the other hand is fully-featured and is heavily utilized in production by many companies today.

Understanding Spring Boot's Key Features

Remember that Spring Boot is a project within the larger Spring family, and it can be used to build both web applications and non-web applications. And also remember that Spring Boot is built on top of the Spring Framework. It has a number of key features that make it unique and easy to use. I'm going to briefly mention each of the most important features, and then we'll go into each in a bit more detail. First, and definitely one of the most notable features in Spring Boot, is auto-configurations. Spring Boot will automatically configure and set up an application based on its surrounding, as well as hints provided by the developer. Second, Spring Boot is standalone. You don't need to deploy the application into a web server, or a special environment, or anything like that. You literally just run the application with one command, just like you would any other application. And third, Spring Boot is opinionated, meaning that Spring Boot has a chosen way of doing things by default. And you're probably thinking, whoa, whoa, whoa, hold on, that doesn't sound like something I'd be interested in. Well, hold your judgment for now, as being opinionated can actually be a good thing.

Intelligent Auto-configurations

Let's talk about auto-configuration. Auto-configuration is a feature of Spring Boot that provides a best-guess configuration for an application, and as the documentation states, it attempts to automatically configure your Spring application based on the dependencies that you've added to it. And it does this by being both contextually aware and smart. Let's look at a real-life analogy. If you were to hand someone a plate, it's completely reasonable and normal to assume that you might also hand them a fork and a knife, but if you hand someone a plate with a cookie on it, you're probably not going to be using that fork and knife. So based on the context at hand, that is, whether or not the plate had a cookie on it, determined whether or not you needed to set up the plate with a fork and a knife. Well, Spring Boot works a lot in the same way. It saves the developer time by making use of context to best guess how an application should be set up. So for instance, if Spring Boot notices that an application has a certain dependency that is related to a database, it can make a reasonable assumption that it should probably configure certain things to access that database, and even more so if that dependency is for a very specific database, such as Oracle or MySQL, Spring Boot can make an even better assumption and probably set up features that may be specific to that certain database. For the developer, setting up auto-configuration is extremely easy. The developer only needs to add one annotation to their Spring Boot application. That's the @EnableAutoConfiguration annotation. And if you're not familiar with what annotations are, that's the thing with the @ symbol, think of them as additional metadata that's added to the code that can be read at runtime and you can use to make decisions upon. Really though, don't worry about the code, it's not really the point. The point here is that it's really, really easy to enable. And likewise, configurations are also really easy to disable. It's not sort of this all or nothing thing. Spring Boot tries to make them as noninvasive as possible. So if they get in the way, they can easily be removed.

Standalone Applications That "Just Run"

Next up, let's talk about the second key feature of Spring Boot, being standalone. Spring Boot makes it easy to create standalone, production-grade, Spring-based applications that just run. Now for someone who isn't intimately familiar with Java web application development, this may not seem like that big of a deal. I mean, how hard could it be? You just double-click the application or you run a single command, right? Well, not exactly. That's not how a typical Java web application is run. To really understand how impactful it is for Spring Boot to be standalone, let's look at how a typical Java web application is run. First, you package the application. You can think of this as zipping up the directory of files that make up the application. Then you have to choose a web server for the web application to run on, and there's several different choices, and each web server is going to offer many different unique features. So some offer a lot of features, while others are more bare bones. And once you've decided what server to use, you have to download and configure that web server. And again, depending on the number of features that the web server has, this configuration step could be extremely involved or relatively simple. You then deploy your application to the web server. This usually means copying the application package to a specific directory or uploading it to the web server via a web browser. And lastly, you start the web application server, which in turn starts your web application. With Spring Boot applications being standalone, they simply just run. Seriously. All you have to do is package the application and run it with a single command, java -jar, and then the name of the package that's your application. Spring Boot takes care of the rest by starting an embedded web server within your application and configures it with some sensible defaults, and then begins serving up your application.

An Opinionated View of Spring

The last key feature we'll discuss is how Spring Boot takes an opinionated view of building Spring-based applications. As I mentioned earlier, don't think of this as a bad thing, like, you must do it this way, or you must do it that way, or else. When you're building Java applications, you have tons of choices, everything from the type of web framework, to the logging framework, to the collection framework, to the build tool you use and so on. If a developer is new to this space, the choices can be certainly overwhelming. And even if they're not new, they usually end up making the same choices based on established standards and popularity. Spring Boot helps remove this burden so that the developer can get up and running as fast as possible, which in turn means they spend more time focusing on writing code that actually solves a need. And although Spring Boot is opinionated in its choices and setup, it's really more of a soft opinion because it makes it easy to override any of those pre-determined choices. To give you an example, let's open up a web browser and visit start. spring. io. This is a tool provided by Spring Boot called Spring Initializer, and Spring Initializer is a great example of how developers can take advantage of Spring Boot's opinionated setup and get started as quickly as possible. The developer simply tells it what type of functionality they need, such as web, or security, or JDBC databases, they click

Generate project, and it automatically creates and downloads a fully-functional Spring Boot application. The developer can then unzip this and use it as an initial starting point for their application.

Where Can I Learn More About Spring Boot?

Since this is a very high-level course that doesn't go into any of the details, it's helpful to know where you can go to actually get those details. There's two highly-rated courses on Spring Boot at Pluralsight. The first one, Creating Your First Spring Boot Application, by Dan Bunker, is an introductory-to-intermediary course for someone who's already familiar with Java, but it's their first venture into Spring Boot. The second course is by myself, and it's called Spring Boot: Efficient Development, Configuration, and Deployment. It's an intermediate course that's a followup to creating your first Spring Boot application. It goes beyond the basics of Spring Boot and teaches the learner how to write their own auto-configurations. It also uncovers the technology behind auto-configurations, teaches the learner how to make use of Spring Boot's external configuration management, and gets them started deploying their Spring Boot applications to the cloud.

Summary

We've reached the end of this module, so let's recap what we've learned. This module was all about Spring Boot. You learned that Spring Boot makes getting started with Spring both quick and easy. Then we saw how Spring Boot uses intelligent auto-configurations to automatically set up and configure an application. Next we understood the standalone features of Spring Boot and how it drastically reduces the time it takes to run an application. And last, we covered how Spring Boot is opinionated and uses this as an advantage so that developers can get started as quickly as possible.

Understanding Spring's Foundations: The Spring Framework

The Spring Framework Is a Software Framework

Hi. My name is Dustin Schultz, and in this module of Spring: The Big Picture course, we'll get a high-level understanding of the foundations of Spring by getting more familiar with the Spring Framework. As the name suggests, the Spring Framework is a software framework, and fundamentally, a software framework is a universal, reusable software environment that provides particular functionality as part of a larger software platform to facilitate development of software applications, products, and solutions. Wow, that was a bit of a mouthful. Let's take a minute to break that down piece by piece and understand how it applies to the Spring Framework. First, a software framework is a universal, reusable software environment. So in other words, a software framework is a piece of software that can be universally used and reused and serves as the support upon which applications are built, and the Spring Framework very much meets this definition. Second, a software framework provides particular functionality, and as this would suggest, it's pretty straightforward. So for instance, in the case of the Spring Framework, the Spring Framework provides functionality for things like web development and data access. Third, a software framework is part of a larger software platform. Again, this is also pretty straightforward. It's just saying that a software framework doesn't exist entirely on its own. It's usually part of something else. So for instance, like this Spring Framework is part of the larger software platform called Java. And last, a software framework facilitates the development of software applications, products, and solutions. In other words, it makes the development of applications easier, just like the Spring Framework makes the development of applications easier. Now that we understand the purpose of the Spring Framework, let's take a moment to understand how it began and where it fits in to the larger picture that is Spring.

This Is Where It All Began

The Spring Framework is what started everything that is Spring today. It's where it all began. And remember that when we say Spring, we're usually referring to the Spring family or the Spring ecosystem. Enterprise Java development at the time was a rigid and complex process, this was around 2003, and the creator of the Spring Framework, Rod Johnson, set out to fix that by creating a framework that applications could build upon to really simplify the development process. The Spring Framework ended up being very successful in this aspect, and over time it sort of grew and evolved into what it is today, and what it is today is actually quite different than what it was when it started. In addition to simplifying Java development, it also became the foundation for which all Spring projects were built upon. For instance, like Spring Boot. Applications could then in turn utilize these projects, along with the Spring Framework, to even further simplify their development. Next, let's take a deeper look at what the Spring Framework consists of and what it provides.

The Six Key Areas of the Spring Framework

The Spring Framework utilizes a modular architecture, meaning that it's broken up into separate individual components that can be connected together. And each of those components is then responsible for providing a specific set of functionality, and together as a whole, they make up the Spring Framework. The Spring Framework can be broken up into six different key areas, core, web, AOP, or aspect-oriented programming, data access, integration, and testing. For the remainder of the module, we'll take a look at each of these key areas and understand what they are, what they provide, and how they simplify Java development.

Spring Core

The Spring Framework's Core module, or Spring Core as it's often called, is one of the most important, if not the most important pieces, of the Spring Framework. It serves as the foundational module upon which every other module is built. Spring Core is responsible for providing a number of different functionalities, such as internationalization support, validation support, data binding support, type conversion support, and a whole lot more; however, at the center of Spring Core is something called dependency injection. The topic of dependency injection is well beyond the scope of this course, but we should be able to get a rough understanding of its purpose. When software is developed using object-oriented programming, developers create objects that represent or model particular things, and objects typically don't exist by themselves. There are usually several objects that make up an application, and each of those objects may in turn have dependencies on other objects. For instance, a developer might create a computer object to represent a computer, and that computer object would likely have dependencies on other objects, like a hard disk object, or one or more memory objects, and dependency injection is about dealing with the way objects obtain those dependencies. There's mainly two different choices for fulfilling object dependencies, number one, the object can fulfill its own dependencies, or number two, the object can declare what it depends on and rely on something else to fulfill those dependencies. Choice number one may seem like the easiest and best idea, but it has some particularly bad limitations. If an object is responsible for fulfilling its own dependencies, then the object and those dependencies become strongly dependent upon each other, or tightly coupled as it's called. To relate this back to a real-life scenario, choice number one would be like buying a computer where you couldn't change or upgrade the memory. When the computer was created, it created the memory as part of the computer. In fact, many laptops are like this today. The memory was created with the computer, and if you wanted to change or upgrade the memory, you'd have to buy a whole new computer. Choice number two of declaring our dependencies is much more flexible. The object and its dependencies are still coupled, but they're considered to be loosely coupled because you can change one without having to change the other. Again, to sort of relate this back to a real-life scenario, choice number two is like buying a computer where you can change the memory. Instead of creating the memory when the computer is made, the computer and the memory are created separately and then later assembled together. Choice number two is what we would call dependency injection, and Spring Core is considered to be a dependency injection container. You can utilize it to declare objects and their dependencies, and Spring Core will create and manage and connect them all together, and this results in less things for the developer to manage. It's also not uncommon to hear Spring Core referred to as the glue of the application, as it's the thing that pieces together and manages many individual parts that form the application.

Spring Web Overview

In this section and the following sections, we'll focus on how the Spring Framework makes it easier to develop services and applications that interact with the web. Web support within the Spring Framework is provided via the web module. And simply put, the web module, or Spring Web, is a framework for handling web requests. Web requests can be handled in one of two different ways, either via Spring Web MVC or via Spring Webflux. In the next sections, we'll take a slightly deeper look at each of these methods.

Spring Web MVC

In this section, we'll discuss Spring Web MVC, and in order to understand what Spring Web MVC is, we first need to understand what a servlet is. The Java language introduced very basic support for interacting with the web via a built-in framework called the Servlet API, and the Servlet API is named after the key component called a servlet. And in simple terms, a servlet is an object that receives a request and generates a response based on that request. Let's take a look at what a standard web request looks like using a Servlet API. We start off with a request to the web server, which is initially handled by the Servlet API, and then passed on to the actual application logic, or business logic as it's called. The business logic is executed and produces some result, and then those results are handed back to the Servlet API to generate a response. Servlets are great, but they don't come without their challenges. Specifically, the servlet API is considered to be somewhat of a low-level API, meaning that it's more detailed and offers less way of abstracting complexity, and this typically translates into something that is harder to use and is ultimately less productive. Because the Servlet API is so low level, it's easy to develop application logic that lacks proper design principles. Code can quickly become unorganized and hard to maintain. Now that we have a high-level understanding of the Servlet API, let's look at what a request looks like using the Spring Web MVC Framework. The request starts off the same, entering the web server and passing through the Servlet API; however, instead of going directly to the business logic, it's sent through to the Spring Web MVC Framework. The business logic can then make use of Spring Web MVC, generate a result, and pass control back to Spring Web MVC, which in turn can pass control back to the Servlet API, and ultimately generate a response. The advantage of the Spring Web MVC Framework is that it provides a higher-level API for the developer to interact with, and this results in easier usage and more productivity. The higher-level API also has the added benefit of making it easier to develop code that follows proper design principles. And you may have been wondering this whole time what exactly is the MVC in Spring Web MVC, well, MVC stands for model-view-controller, and it's a design principle, or a design pattern, that results in more organized and easier-to-maintain code. If you want to learn more about Spring Web MVC, be sure to check out the Pluralsight course called Introduction to Spring MVC 4, by Bryan Hansen.

Spring Webflux

Spring Webflux is another way of handling web requests supported by the Spring Framework. And in order to understand its purpose, we first need to understand something called reactive programming. Reactive programming is a declarative programming paradigm concerned with data streams and the propagation of change. So in other words, it's a way of programming that focuses on streams of data and how they change. And the key idea behind reactive programming is that you react to change rather than wait for change. As I mentioned before, Spring's reactive web programming framework is call Spring Webflux, and its primary purpose is still to handle web requests, but the way in which it handles them is different. Specifically, web requests are executed asynchronously, and they don't block or wait, and this results in better resource utilization. To better understand, let's look at what a traditional, or non-reactive request flow looks like. As a web request progresses over time, you can think of it as executing a series of steps, and some of those steps may depend on external resources like loading a file from disk or making additional requests to other servers, and since the requests must occur in order, every time a step is waiting on something else, the request is considered to be blocked. And this is less than ideal because when the request is blocked, computing resources are tied up and they can't be used to serve other requests. To sort of relate this to a real-life concept, the traditional flow is a lot like dialing a support number and having to wait until the next agent is available. While you're on hold, you're basically tied up from doing anything else. Now let's talk about the flow of a reactive request. Just like the traditional flow, you can think of the web request as progressing over time and executing a series of steps that may depend on external resources. The difference comes with how the requests are executed. Requests are executed asynchronously instead of synchronously, and they don't block or wait. This allows the valuable resources involved in the request handling to be released and reused. And you're probably wondering though, if it doesn't wait, then how the heck does it work? Well, instead of waiting, the code simply asks to be notified when the operation is complete, and continues executing other operations. When it's observed that the operation is complete, the code can continue executing those subsequent steps. Again, relating this back to our real-life example, the reactive flow is a lot like dialing a support number that allows you to put in your telephone number and receive a call back when the next agent is available. Since you're not tied up on the phone waiting for that next available agent, you're free to do other things until your phone rings. You can learn more about Spring Webflux with the Pluralsight course, Spring Webflux: Getting Started, by Esteban Herrera.

Spring AOP

In this section, we'll learn more about Spring AOP. And in order to understand what it is, we first have to understand what AOP is. AOP stands for aspect-oriented programming, and it's a programming paradigm that aims to increase the modularity by allowing the separation of cross-cutting concerns. Now, you may be a little confused with what exactly that means, so let's break up this definition into three separate parts. First, when we say it's a programming paradigm, that simply means that it's just a way of programming or writing code, and second, the definition says that it aims to increase modularity. And in simple terms, this just means that it increases organization. And third, it allows you to separate cross-cutting concerns, or concerns that affect many parts of an application because they span multiple layers, or tiers, of the application. A classic example of a cross-cutting concern is security because security can't be contained to any one area of the application, and instead it spans multiple parts of the application. You can find it throughout various areas of the application. So just to recap, AOP is a way of programming that increases organization of code for concerns that span multiple layers, or tiers, of an application. Just to give you an idea, without AOP, solving concerns that are cross-cutting, results in scattered and duplicated code across many parts of the application. AOP is really beyond the scope of this course, but let's quickly look at the benefits that Spring AOP provides, and we'll finish with a concrete example. Spring AOP is an implementation of AOP within the Spring Framework, and it mainly has two uses, with the first being used to implement certain features within the Spring Framework itself, and the second being a valuable tool for developers that need to solve problems that span multiple layers, or tiers, of their application. It's sort of a nice tool to have in their toolbox. As I mentioned before, security is a great example of something that AOP can solve. So let's look at a typical example of a piece of code that's only allowed to be executed by persons who are authorized and authenticated. We first check if the user is authenticated and if they have the correct role, for instance, like an administrator role, and if they do, this sensitive operation is allowed to execute. And if not, we raise an error, maybe log that failed attempt, and redirect the user to a login page. This same series of operations is likely to occur in many places throughout the application. Basically anywhere there's a sensitive operation. And instead of spreading this same logic throughout all of the application, Spring AOP allows us to organize these operations into a single area that can be applied to different parts of the application. Now let's look at an example that utilizes Spring Security, a project that we'll discuss further in a future module. Spring Security utilizes the Spring Framework's AOP support behind the scenes to implement security. In our example, the sensitive method is annotated with the @PreAuthorize annotation, and it has a condition to ensure that the user has the role of admin for instance. And remember that annotations are just metadata that's added to the code. And by adding this metadata, Spring AOP can identify that this method requires security, and apply the appropriate security logic. The added benefit of this is that the logic within the sensitive operation becomes very clear, and it's not cluttered with surrounding logic that's not directly related to what the logic is trying to accomplish just to ensure that the user is secured and authorized to execute this method. It's just the sensitive operation. If you want to learn more about Spring AOP, there's an excellent course on Pluralsight by Eberhard Wolff called Aspect Oriented Programming using Spring AOP and AspectJ.

Spring Data Access

Whether it be storing it, retrieving it, or both, almost every application interacts with data these days. Data is super important. And the easier it is to develop applications that can interact with data, the better. And that's the job of the Spring Framework's Data Access module. To get started, let's start off with an example. On the left we have some example code that's used to fetch data from a database using the Java Database Connectivity API, or JDBC as it's commonly referred to. Now don't worry about the code. The specifics really aren't important. The important point to make here is that it requires a fair amount of code to retrieve data, and if you look a little bit closer, you'll realize that there's really only a small area of the code that's important to the logic. The rest of the surrounding code is required to use the API. This code that's necessary to use the API, but sort of causes clutter in our application is called boilerplate code. And as you can imagine, adding this same boilerplate code every time you needed to write or retrieve data becomes very tedious. Let's take a look at what this same code looks like, but using the Spring Framework's Data Access support. As you can see, the code on the left is significantly reduced, and not only is it reduced, it's much more clear as to the intent of this code. We don't have to weed through the boilerplate code just to understand the purpose of it. The Spring Framework's Data Access support also makes it super easy to use database transactions. Now, you may not be familiar with what a database transaction is, so let me quickly explain. A transaction is a general unit of work that must happen all together or not at all. So a database transaction is a series of database operations that must happen all together, or not at all. And just to be clear, that doesn't necessarily mean that they all happen at once, they just all happen or they don't happen at all. A classic real-life example of a transaction is transferring money between two bank accounts. You first have to subtract the amount from the first account, say it's $50, and then you add that amount to the second account. And if the subtraction of the money succeeds, but the addition fails, then the money's lost. And likewise, if the subtraction fails, but the addition succeeds, then somebody just got 50 bucks for free. So as you can see, transferring money between two accounts must be transactional. It must happen all together or not at all. Let's take a look at another example. Again using the JDBC API, the following code on the left is a pretty standard example of executing a series of statements within a database transaction. And just like with our previous example, you'll notice there's a lot of boilerplate code here that sort of draws focus away from the main core intent of the code. Let's see what this same code looks like using the transactional support in the Spring

Framework. Wow, that's a lot simpler. You simply declare the method as transactional, meaning that you want this method to execute within a transaction, and the Spring Framework takes care of the rest. Another small, but interesting feature provided by the Data Access module is something called exception translation. And exceptions are events within a program that disrupt execution. So for instance, if you tried to read data from a database and you got an error, the Java code would throw an exception. And the problem is, different database vendors have different errors and different exceptions for the same class of errors. So for instance, consider an example where you had problems with the data, and when using MySQL, a data integrity violation could return any one of the following codes, such as 630, or 839, or 1557. And that same data integrity violation in Oracle would return something like 1400, or 2292, and the Data Access module of the Spring Framework helps by taking these database-specific error codes and translating them into a well-known set of exceptions. So just to make this a bit more concrete, looking at the different error codes from MySQL and Oracle, you can see that the Spring Framework would translate those into something called a DataIntegrityViolationException. Last, but certainly not least, Data Access support within the Spring Framework helps to make testing easier. Since the Spring Framework is managing the configuration and the set up of how an application accesses its data, it's very easy to switch out that configuration for testing. So for instance, during testing it's common to utilize an embedded test database. You certainly wouldn't want to be running your tests with important production data, and Spring's Data Access support helps to set up that embedded test database and ensure that your code is configured to point to that during tests.

Spring Integration

In this section, we'll briefly take a look at the features that the Spring Framework provides for integration. Now what exactly do I mean when I say integration? Well, applications don't live in isolation. Applications often need to share data or work with other applications, and integration is all about making different systems and applications work together. So how exactly does the Spring Framework help? Well, one problem you need to solve is how one application can talk to another, or communicate. And this is a multi-sided problem. So A) you have to figure out how do you expose operations to other systems, and B) how do you invoke, or run, operations on other systems? There's certainly many different ways to expose operations to other systems, such as RMI, or Remote Method Invocation, or another way is via a messaging system, and the Spring Framework supports both of these, but one such popular way to expose operations is something called web services. And web services are simply operations that are exposed and accessible via the web. And the Spring Framework makes it easy to both expose and invoke web services. Let's take a look at an example. On the left we have a very simple example of a web service, and this can be used to retrieve account information. So you might think of this as being used in a banking application to look up your bank account. And at the core of this web service are three annotations. Remember that annotations are just metadata that's added to the code. The first annotation is the @RestController annotation, and it denotes that this code is going to be exposing operations using a style of web services called REST, and the details of what REST is aren't really important, just know that rest is a way or style of implementing web services. The second annotation, the @GetMapping annotation, is used to denote the operation and the path that need to be used to invoke this operation. And we can see that the path is just like any regular URL that we're used to seeing on the web with some additional added curly syntax. And the curly braces are related to the third annotation, and they simply indicate that this part of the path is a placeholder. In our case, it's a placeholder for the account number, and in other words, it's the way that we tell the operation what account number we'd like to retrieve. So even if you don't understand the code, you can see that with these three simple annotations, we've exposed a service that can be used to look up an account by its account number. Now that we've seen how to expose an operation via a web service, let's look at the other side of the problem and how the Spring Framework helps us to invoke our service from within another application. Remember how we utilized REST to implement our web service? Well, the Spring Framework provides support for programmatically invoking our REST service, and you utilize something called a RestTemplate. And that RestTemplate abstracts away all the tedious details of interacting with a web service, like opening a connection to the web service, sending over the command, and handling the response from the web service. In fact, it's so easy to use that it'll only take us one line of code to call a web service that looks up an account by its account number. Using the RestTemplate, we can call the GetForObject method, pass a few parameters, one of which is the path to the web service with the account ID that we want to look up, and the RestTemplate will take care of the rest. No pun intended.

Spring Testing

Testing is an important part of developing software, so it's no surprise that the Spring Framework strives to help in this area as well. And there's many different levels at which software is tested, and the Spring Framework focuses on two different areas, on unit testing and integration testing. Let's start by talking about unit testing. Unit testing is a software development process in which the smallest testable parts of an application called units are individually and independently scrutinized for proper operation. And explicit support for unit testing in the Spring Framework is actually minimal. The Spring Framework support for testing largely comes as a side effect of using one of its key concepts, and that's dependency injection. And we talked about dependency injection in the section on Spring Core. So just to quickly refresh your memory, dependency injection is all about dealing with the way objects fulfill their dependencies. So you're probably wondering how exactly does dependency injection help make testing easier? Well, remember that with unit testing, the idea is to test the smallest unit of code possible, and when the code that's being tested has no dependencies, meaning that it doesn't depend on other code to do its work, this is a fairly easy process; however, when the code under test has dependencies, it's important to be able to control how those dependencies behave in order to restrict the testing to just the code and not the dependencies. If the code is written with dependency injection in mind, the testing becomes a lot easier. The developer is forced to declare what the code depends on, and the code expects to be given those dependencies, and the code doesn't necessarily care where they come from, just the fact that something fulfills them. So during testing, dependent code can be replaced with code that behaves a certain way, and this action of replacing pieces of code with controlled code is called mocking. Now just to be absolutely clear, in unit testing, we don't actually use dependency injection to inject dependencies into our unit tests, but the way that the code is set up for dependency injection, mainly that the code is forced to declare what it depends on, that helps to make testing easier. If you're confused or still not grasping the concept, hopefully it'll make sense with an example. Suppose you have a piece of code that grants people certain permissions based on their security clearance level, and suppose that there's three different types of security clearance, confidential, secret, and top secret. Now I don't expect that you'll understand this piece of code, so let's quickly walk through it. If you notice the method, the getPermissions method, is passed a parameter called the SecurityManager, and the SecurityManager is responsible for getting the security clearance level. And then we have a series of if/else if statements that just check what that level is and grant certain permissions based on that level. When testing this piece of code, you want to be able to test every condition, or what happens in every scenario, every if block, and luckily this code has been written with dependency injection in mind and declares that it depends on the SecurityManager. During testing, instead of using the real logic to determine what the user's current security clearance level is, we can mock this dependency and control the way the SecurityManager behaves, or in other words, we can control what security clearance level is returned. So for instance, if we were writing a test here, for our first test we could make it so that the SecurityManager that it uses always returns confidential, and that would ensure that we enter that first if block. And then for our second test, we could make it always return secret and it would make sure that it enters that second if block, and so on. If the SecurityManager had not been declared as a dependency, this code would have been much harder to test because we wouldn't be able to as easily control the logic of the SecurityManager to test all of the various conditions. As part of the support for unit testing, the Spring Framework includes a number of prewritten mocks that can be used in testing, and this ultimately makes testing easier and faster for a developer because it saves them from having to write their own mocks. The second area of focus for testing within the Spring Framework is integration testing, and integration testing is the phase in software testing in which individual modules are combined and tested as a group. It occurs after unit testing. And as the name suggests, integration testing is focused on how various parts of a software application integrate together. Integration tests allow you to run whole pieces of an application and test how they all work together and not just pieces of code in isolation like unit tests. Outside of explicit support for integration tests, the Spring Framework plays an important role in integration testing, as it could be used to piece together various parts of the application for testing. In terms of explicit support, the Spring Framework provides support for common integration testing scenarios, like setting up and testing data access, as well as other scenarios like setting up and testing whole parts of a web application to ensure they're functional. Another nice feature provided by the Spring Framework is the ability to clean up after tests. And let me explain what I mean by that. So when you're writing integration tests, it's very common that you'll be changing the state of the application just by running the test. For instance, maybe you do some data access tests and you insert some new data into a database. And if you leave that data in there after the test is complete, it could end up affecting how other tests run after it. So the Spring Framework provides support for helping you clean up those tests after they run.

Summary

We've reached the end of this module, so let's take a moment to recap what we've learned. We've covered a lot of content in this module. We first saw Spring Core and how it's the foundation of the Spring Framework and how it provides support for dependency injection, which is a key component of the Spring Framework. Then we got to see the web support within the Spring Framework, as we talked about Spring Web MVC and Spring

Webflux. After that, we got to learn what AOP is, or aspect-oriented programming, and how Spring AOP can be used to solve cross-cutting concerns. We then learned all about the Spring Framework's Data Access module and how it removes a ton of boilerplate code and makes it easier to access data. And then after that, we took a brief look at the Spring Framework's integration features and how they help different applications and systems work together, and we finished out the module by learning about the Spring Framework support for unit and integration testing.

Exploring Other Spring Projects

The Projects We'll Be Exploring

Hi. My name is Dustin Schultz, and in this module of Spring: The Big Picture course, we'll be taking a tour of several of the other Spring projects that exist, understanding what they do and how they make development easier. If you open up a browser and visit spring. io/projects, you'll be brought to a page that contains all the projects that exist within the Spring family. You can scroll through the projects and see what's available and click into any of the projects for more details. There's over 20 different projects at the time of this recording, and as nice as it would be to be able to explore all of these projects, we'll only be focusing our tour on a subset of the most popular and widely used projects. The projects we'll be exploring in this module are Spring Data, Spring Cloud, and Spring Security.

Spring Data Project

We'll start off by talking about the Spring Data project, and you can visit the Spring Data project by visiting projects. spring. io/spring-data. Spring Data's mission is to provide a familiar and consistent programming model for data access while still retaining some of those special traits about the underlying data store, and this means that regardless of the type of data that's being stored, Spring Data utilizes sort of these common patterns across all data stores with the additional goal of ensuring that any of those unique traits that are specific to a specific data store are still available. You might be wondering how exactly is the Spring Data project different than the Spring Framework's Data Access support, and that's a great question. The Spring Data project extends the data access capabilities provided by the Spring Framework, whereas the Spring Framework is focused on one particular type of database, particularly relational databases. The Spring Data project adds new ways of interacting with relational databases, as well as support from many different types of databases. Spring Data is actually an umbrella project, meaning that it itself is not actually a project, but a series of subprojects, specifically a subproject per supported data store. And if you scroll down in the main project page, you can see all the various data stores that are supported. As you can see, there's a lot of different submodules to support a lot of different data stores, and there's the Main modules, which are supported within the Spring Data umbrella, and then there's the Community contribute modules that are written and maintained by others outside of the Spring Data team. One thing that's nice about all of the Spring project pages is that they show a quick example of using the technology. So if we scroll down to the bottom, we can see a section labeled Quick Start, and without going into the details, let me give you a quick overview of that this example is doing. This particular example is showing how you can create an object that holds some data, and in this case, it's an Employee object, and then Spring Data helps make it easy to save, and delete, and find any of that data, all with built-in methods and little to no additional coding required. Spring Data, along with some other technologies, like JPA, which is called the Java Persistence API, does the hard work of taking the data that's stored in the object and translating it, or mapping it, or converting it into something that can be stored into the target data store. If you're interested in learning more about Spring Data, there's two Pluralsight courses that utilize Spring Data. The first course is the same technology that we saw in the demo, and it's called, Getting Started with Spring Data JPA, by Dan Bunker, and the second course is called Getting Started with Spring Data REST, and it's also by Dan Bunker.

Spring Cloud Project

The next project we'll take a look at is called Spring Cloud, and you can visit this Spring Cloud project page by visiting projects. spring. io/spring-cloud. The Spring Cloud project is a project that's built on top of Spring Boot, and it makes it easier to build distributed systems. Now if you're not familiar with what a distributed system is, a distributed system is just a number of different applications that are all linked together by a network. And a common and popular example of a distributed system is a microservices architecture. And again, you may or may not have heard of what a microservice is, so let me explain. A microservice is just a small application that's very well scoped to a single purpose or domain. So for instance, if you have something like an account microservice, its only job is to deal with accounts in the system. It knows how to create them, and retrieve them, and edit them, and a microservices architecture is just a bunch of microservices that are communicating with one another over a network. When you implement something like a microservices architecture, you end up running into a number of well-known problems that have well-known solutions. And one example is, how does a microservice find or discover another microservice that it needs to do its own work? And this problem is called service discovery, and it's exactly one of the things that the Spring Cloud project helps you implement. If we scroll down and take a look at the example here, we can see that there's one new interesting annotation that we've never seen before. And that's the @EnableDiscoveryClient annotation. And by using Spring Cloud and adding this annotation to an application, the application, or service as it's usually called, cannot only find other services, but it itself is also discoverable by other services. Service discovery is just one of the many things that Spring Cloud helps with when building a distributed system, and just like the Spring Data project, the Spring Cloud project is not actually a project itself. It's an umbrella project that contains a number of different subprojects, and those subprojects can help you with things like configuration and failures within a distributed system. If you want to learn more about Spring Cloud, there's a great course on Pluralsight, authored by myself, called Spring Cloud Fundamentals.

Spring Security Project

In this section, we'll take a quick tour of the Spring Security project. You can visit the project page for Spring Security by visiting projects. spring. io/spring-security. The Spring Security project is a framework for securing Spring-based applications. It handles both authentication and authorization, and it covers all the standard ways of handling security, as well as the ability to meet any sort of custom needs if needed, and it's not only for authentication and authorization. The Spring Security project also has additional features that help to protect your application against common attacks, like cross-site scripting or click jacking, and you don't really have to understand what those are to understand the benefits, just know that Spring Security is also used to make applications that are developed with it, more secure. To see a quick example of how the Spring Security project is used, let's click on the Getting Started link under the Additional Resources section, and this will take us to a page that lists a number of different getting starting guides for Spring Security. And we'll be looking at the Hello Spring Security with Boot example, which uses Spring Security along with Spring Boot. Once you're one the guide page, locate the table of contents and click the section Creating your Spring Security configuration. This is the main part of the application that demonstrates using Spring Security, and will really give us a taste of what Spring Security is like. If you scroll down a bit, you'll see some code that configures the security in a web application. As always, the details of the code aren't important, but let me explain at a high level what some of the code is doing. First, there's the @EnableWebSecurity annotation, and this, as the name suggests, sets up the application to be able to handle secure web requests. Now let's focus on the configure method. The code is configuring what paths are allowed and what paths require authorization, and it's also configuring the location of the login page for the application. And if you look at the code, it's very readable. You can see that any request for the path /CSS or /index, are always permitted, but any requests for /user path require the user to be authenticated and authorized with the role of user. This is a very real example of code that you might find in an application today using Spring Security, and it shows you just how easy it is to allow certain paths of a web application to always be allowed, or other parts to require security. You can learn more about Spring Security by watching this Spring Security Fundamentals course, by Bryan Hansen.

Summary

We're at the end of this module, so let's take a moment to recap what we learned. This was a particularly short module where we took a quick and high-level overview of some of the most popular and widely used Spring projects. We first talked about Spring Data. We saw how it helps you to access data with little or no code required, and we also explained how it's not the same as the Data Access support that exists within the Spring Framework, and instead it extends it to provide additional support for additional types of data stores. We then talked about Spring Cloud and saw that its purpose is to help build distributed systems. We talked about how microservices are a way of implementing a distributed system and how Spring Cloud helps by providing solutions and patterns, like service discovery, to these well-known problems that exist when you build distributed systems. We finished out the module by taking a look at the Spring Security project, and we saw how it can help you to easily secure an application with authentication and authorization. And we also saw that it not only helps you to implement security, but it also helps you to secure your application against common attacks.

Is Spring a Good Fit?

Is Spring a Good Fit for What I'm Doing?

Hi. My name's Dustin Schultz, and in this module of Spring: The Big Picture course, we'll be focusing on one very important question. Is Spring a good fit for my project, my team, my company, etc.? And in order to answer that question, we'll go over a number of advantages and disadvantages to using Spring. And also, let me be absolutely clear, I'm likely biased in my opinions on Spring, whether I want to be or not, but I'm going to try my hardest to be as unbiased as possible.

The Advantages of Using Spring

Let's go ahead and start off with some of the advantages of using Spring. One of the first ones is that almost everything from Spring, especially the core projects, is rock solid and well engineered. Spring has strict engineering practices and is backed by a company that invests time, money, and resources into the project. Spring has also stood the test of time. It's been around for many years, and it's continued to evolve, it's not a stale project, and it's also not the next shiny or new framework, it's here to stay. Spring has a huge community. You can be pretty sure that if you run into a problem, somebody else out there has probably already ran into the same problem. And if they haven't, you can be pretty sure that there's someone out there that can help. Spring is very well-liked by many developers, and in a recent survey from a popular developer-oriented website called Stack Overflow, Spring was part of the top most loved development frameworks, and Spring also made the list as one of the most widely used frameworks. With Spring, there's a sufficiently large talent pool and hundreds of open positions asking for engineers with Spring skills. So whether you're a hiring manager looking to fill a position, or a developer looking to learn a new skill, Spring is very well used in the industry. As I mentioned before, Spring has been around for a long time, and the other benefit of that is that there's a wealth of knowledge that already exists out there, from books, to articles, to videos, you can be sure that there are available resources to learn Spring. Spring is still very actively developed, and since it's an open-source project, you can look online and see how active the project is, and it's very much alive. Again, since Spring has been around for so long, it has a lot of built-in IDE support. Many of the popular IDEs, such as IntelliJ and Eclipse, already have built-in support and plugins available for Spring. And just in case you're not sure what an IDE is, it stands for integrated development environment, and it's the development environment in which you develop the code and having support for different technologies makes it easier to develop that code. Spring also scales, and what I mean by that is that Spring is responsible for powering many of the large and highly-trafficked applications that exist today. It's proven that it can handle load and it can scale. Now this is by no means a comprehensive list of all of the advantages of Spring, but it should give you a good sense of why you might want to use Spring. Now that we've seen the advantages, next let's take a look at some of the disadvantages, or drawbacks.

The Disadvantages of Using Spring

Everything has disadvantages, and Spring is no exception. Let's take a look at some of the disadvantages of using Spring. One of the most common disadvantages that you'll hear about Spring is that Spring is too magical. You sort of add an annotation here or there and all of a sudden you magically have a fully-working app, and this can certainly be a drawback because the reason that something happened, or the reason that something worked is not always intuitive. Spring can also have a steep learning curve, as Spring has been around for a very long time, learning everything there is to learn about it can be overwhelming. Spring can also increase the size of your final deliverable, and in most cases this isn't an issue to be concerned with, but when it matters, Spring can be considered to be a bit large and a bit bloated. Another common disadvantage that you'll hear about Spring is that it's hard to debug, and this is true in some senses as Spring hides a lot of details and it can be hard to troubleshoot when something goes wrong. You may not even know where to start, or even what went wrong. Spring also add some additional memory overhead to applications, meaning that it takes more memory to run. And in some cases, it's really quite minor, and in other case, it's more pronounced. And I should note though that this will be true of any framework. The complexity of Spring has grown over time. It started out as a way to reduce complexity, and it largely accomplishes that, but over time Spring has become complex itself, and in many ways this is why the Spring Boot project was born. Spring is sort of configurable to a fault, and this one falls in line with the complexity that we just talked about. In Spring there's many knobs and dials you can turn, and it's very configurable, and with this comes that additional added complexity. Spring is big, and by that I mean it's not a small framework, and it's not usually a good solution for problems where you need to write code to give you a quick result. Spring was meant to build long-running applications that solve complex problems. Some of the community projects from Spring can sort of be hit or miss. Sometimes they're really great and sometimes they just fall short on features. And just like I mentioned when we were talking about the advantages, this is by no means a comprehensive list of all of the disadvantages of Spring, but it should give you a good idea of what are some of the drawbacks, or disadvantages.

Making the Decision

So now that we've see the advantages and disadvantages of using Spring, we're back to our original question, is Spring a good fit for my project, my team, my company, etc.? Well, unfortunately, I'm not going to directly answer that question for you because every scenario is different, and no matter how I answer it, I'll probably be wrong. And likewise, coming to that conclusion on your own will be much more meaningful and thoughtful. When it's your decision and someone asks you why you chose Spring for your project or your team or company, you'll be in a great position to be able to back it up with well thought-out reasoning. I will say consider the pros and cons against your situation, as pros may not always be pros in different situations, and the same goes for cons. I would love to hear your comments in the discussion section on what conclusion you came to and why, or why you're considering using Spring. I try my best to read and reply to all of the comments.

Summary

In this module, we looked at answering the question, is Spring a good fit? And we talked about the advantages of Spring, like how most Spring projects are very stable and well engineered, or how Spring has been around for a long time, and it's easy to find learning resources on. And we also discussed the disadvantages of using Spring, like how it's hard to debug, or how sometimes it can be too magical, and it's not always intuitive why something worked or why something happened. And we finished out the module by discussing that there really isn't a clear-cut answer to our question and that every situation is different. We saw that we really have to weigh the pros and cons of our scenario because not every pro is a pro and not every con is a con depending on the situation.