9) HTTP & Web Services Lesson

Introduction to Web Services

14 min to complete · By Ryan Desmond, Jared Larsen

Contents

So far, this course has covered the parts of the Spring Framework essential for the creation of local applications. These new skills have provided you with an immense amount of power. You can write complex Java applications using dependency injection to automate tasks and easily interact with a database - all with a lot less boilerplate code than before.

But, aside from the introductory *comedy* and *quotes* projects that basically threw you unknowingly into the deep end, the labs and applications you've encountered so far have basically been "one-way streets". There is no user interaction; you're just running code locally and inspecting the results. Knowing this, the question then becomes "Well, how do my applications become interactive?" You could interact using the console like you did in core Java projects, but you're here for more than that. You're here to learn how to build modern web applications. The keyword here is **web**, as in the world wide web, aka - the internet, and it is the key to interacting with your applications.

Even when running your applications locally on your machine, web services can facilitate interaction. Think back to your Spring Data `datasource` property - even though MySQL

Server is installed locally, your application can still utilize web-based protocols and communicate with it using a URL. When you ran the example comedy project, you accessed its interface using a web browser. Interaction involves communication with your application in any form - from data sent silently from one service to another (with no visual indication), to beautifully designed user interfaces that control application behavior - and there are endless options in between; the overwhelming majority of which utilize the web.

So before you can learn to build amazing interactive web applications, there is some work to be done. Get ready. This whole section covers how the internet works. At the core of this understanding are web services. This lesson will help you wrap your head around the overarching concepts the internet is built on. In this lesson, you will explore:

- What a web service is.
- Common types of web services.
- Why web service communication is so helpful.

What is a Web Service

Simply put, a web service is any software application that is capable of communicating over a network. That network might be the internet, a local network (closed off from the internet), or even virtualized within your local machine as described above. Generally, these services will communicate over the web and use standardized protocols like HTTP. They are often implemented as APIs (Application Programming Interfaces), which will be detailed in upcoming lessons.

A modern web app (service) might seem straightforward. You request a resource over the web, and it returns it. It is absolutely possible to set up a relatively simple service like this, but for most web services, there is much more going on behind the scenes. The typical web service may be pulling data from multiple sources and processing that together with data from its own database. It may be informing other services across the web of actions taken by a user. It could be asking other services to perform any number of tasks, etc.

These features make web services incredibly powerful. The crux of that power lies in their ability to communicate effectively. As a Java developer, you may be asking yourself: "How does my Java application communicate with that Python system over there? Is that even possible, or is there some translation mechanism available?" The answer to this

question is a bit more complicated than the code equivalent of Google Translate. In order to talk to systems written in different languages, web services use specific protocols and data formats to standardize how information is transferred.

The following example may help clarify this: A French speaker and an English speaker want to communicate. In real life (before translation apps) the French speaker would learn English, or the English speaker would learn French, and they could speak. But if these two individuals were to take the above-mentioned protocol route, both of them would learn Spanish in order to talk. Instead of one learning the other, they both learn something new.

These protocols are such an integral part of a web service that the protocol a service uses determines what type of web service it is.

Types of Web Services

There are two service types which have cornered most of the market:

1. SOAP Web Services: **SOAP** stands for Simple Object Access Protocol. This messaging protocol is designed specifically for scenarios that require a highly decentralized and distributed system. To do this, SOAP supports all the popular application layer protocols such as *HTTP*, *UDP*, *TCP*, or *SMTP*.

SOAP Web Services transfer information in a slightly outdated format called Extensible Markup Language or XML. This format is still used by some (checkout a Maven **pom.xml** file), but there are newer and less verbose formats out there that are taking over. SOAP's usage has become less common, but is still relevant in certain industries like banking, telecommunication, and healthcare (among others).

2. RESTful Web Services: **REST** stands for Representational State Transfer. Unlike SOAP Web Services, the RESTful protocol only supports HTTP and HTTPS (don't worry if you don't know what these are yet, you're about to do a deep dive). This may seem like the inferior web service type since it can only support one of the many protocols that SOAP Web Services can, but supporting fewer application layer protocols make RESTful services much more lightweight.

That's not all, RESTful web services also allow you to transfer data in almost any format. RESTful services can support plain text, HTML, JSON, XML, binary, YAML and more. The

trend in web service development has been moving towards RESTful services due to their simplicity, flexibility, and compatibility with modern web development practices.

Info: As you may have noticed, there was a little bit of bias in this comparison of services. This was intentional. You should prefer RESTful services. They are better in almost every use case.

When a web service adopts one of these protocols, it becomes platform and technology independent, meaning it can connect and communicate with a variety of other applications even if they are built using completely different technologies. A `C++` or `Python` application can now talk to a `Java` application and vice versa.

How Web Services Help

Let's talk a bit about the upsides of adopting a web service architecture:

- **Improved development productivity**: More and more businesses and individuals are putting services up on the web that you can call on to perform tasks. Need to verify someone's identity? Just forward the pertinent information to an identity verification service, and voilà. Minimal effort required on your part.

- **Extended life of legacy systems**: Web service protocols allow new applications to communicate with old systems. Imagine you are stuck with an outdated auto-emailing system and want to include a new component that allows you to send emails in batches. If you adopt a standardized protocol, you can add new and updated services to your legacy system without starting from scratch.

- **Ease of pulling information from many sources**: With such interoperability, web services can query information from almost any source without much difficulty. In this day and age this is common practice. The world revolves around data. The more you have, the better off your predictions and systems are.

What's Coming Up

Even though it is useful to know how to set up both web services, there is only so much one can put into a course without overwhelming students. As a result, this course focuses solely on RESTful services.

Ignoring SOAP means we avoid the common "an inch deep and a mile wide" issue. You'll be a pro at RESTful instead of mediocre at both SOAP and RESTful. And with this skill set at hand, *if* by chance you run into a SOAP service that you need to communicate with, you'll be well-prepared to conquer that challenge.

Summary: Web Services

- A web service is any software application that is capable of communicating over a network.

usage has become less common, but is still relevant in certain industries like banking, telecommunication, and healthcare.

  - RESTful Web Services: **REST** stands for Representational State Transfer. The trend in web service development has been moving towards RESTful services due to their simplicity, flexibility, and compatibility with modern web development practices.

- Web services help you:
  - Improve development productivity.
  - Integrate with external services.
  - Easily pull data from many sources.

Learn more

Previous

Next → JSON: The Preferred Data Format for Web Services

Discord