


Home > Course > Design Your Software Architecture Using Industry-Standard Patterns > Service-Oriented Architecture


Design Your Software Architecture Using Industry-Standard Patterns

4 hours  Hard

Last updated on 6/29/20



Service-Oriented Architecture

 [Log in](#) or [subscribe](#) for free to enjoy all this course has to offer!

In this chapter, we are going to cover the service-oriented architecture, and what problems it solves. At the end of the chapter, I will show you a real-word example (GlobalWeather company), and then you will solve a similar case applying what you've learned.

What Is Service-Oriented Architecture?



Imagine you are buying an item at the retail site of a widely-known brand. You select the item, size, color, and then check out from the site. You pay for the item with your credit card or other payment method and choose delivery to your home, perhaps by a delivery service like FedEx.

Based on your delivery address, the website gives you a tracking number for your package right away. It is a unique, long number (20 digits) used to track your package. From now on, you can access the retail website and check the location of your package at any time.

If we examine this situation, it looks like the retail website “spoke” with FedEx when you made the purchase and then was able to tell you your tracking number immediately. The two companies communicated with each other when you bought the item, so you could track your package with the given number.

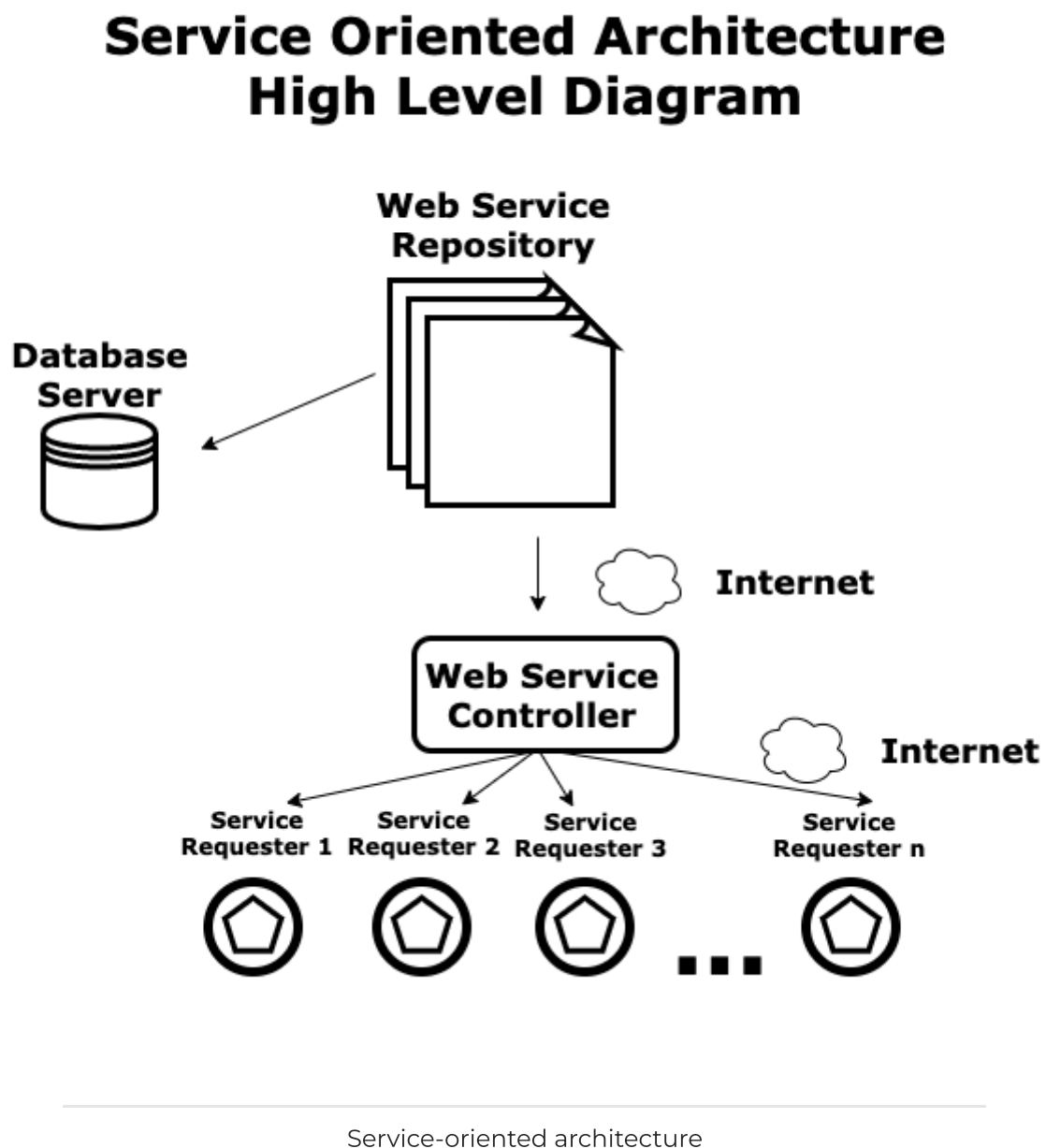
The two sites made this happen thanks to a **service-oriented architecture**. In this architecture, **services** are provided to external consumers through a communication process over a network (the internet). In our example, FedEx is the **service provider**, and the retail site is the **service**

consumer. FedEx's service is made available to this external customer via a service-oriented architecture.

What is the Structure of a Service-Oriented Architecture?



Let's see what this looks like:



As you can see in the diagram above, a standard client-server architecture has three parts:

- **Web service repository:** This is a library of web services built to serve external requests for information. The served information is usually a little piece of information, like a number, a word, some variables, etc. For example, a flight number, a package tracking number, the status of an order (one letter), etc. This library is usually documented in great detail since external applications will call the functions it contains.

- **Web service controller:** This module communicates the information in the web service repository with the service requesters. When an external service requester calls a certain function from the web service repository, the web service controller interprets the call and looks for the function in the web server repository. Then it executes the function and returns a value to the requester.
- **Database server:** This server contains the tables, indexes, and data managed by the core application. Searches and insert/delete/update operations are executed here.
- **Service requesters:** These are external applications that request services from the web service repository through the internet, such as an organization requesting flight information from an airline or another company asking the package carrier for the location of a package at a given moment.

Note that service-oriented architecture is **created by the companies providing the service** - not the ones consuming it, and this is done to simplify the connections to possible clients.

How does this work in practice?

Let's take the example of Visa providing a service to a popular jean company:

- The Levi's website needs information from VISA (service provider) to complete a transaction.
- Levi's website calls the function "Validate_Credit_Card_Number 5555 4567 8901 3456" from the web service repository inside the VISA site.
- The VISA website receives the request, executes the function, "Validate_Credit_Card_Number 5555 4567 8901 3456," and returns the value "OK," meaning the credit card is valid.
- Levi's website receives the value "OK" and uses it to complete the transaction.

There are several **advantages** to using service-oriented architecture:

- This model allows collaboration with external players without letting them access our systems. In the example above, VISA does not want to open its database to third parties due to security reasons, but it still wants customers to be able to get the information they need.
- It also allows the selection of the most popular site functions to be shared with the world in an ordered, controlled fashion. VISA says: "if anyone wants to know the status of Card X, call this function, give the value of X, and I will answer yes or no." The function is open to the world as a service in a very strict, controlled environment; customers cannot access anything they want.

There are also a few key **disadvantages**:

- Web services can have site weaknesses that invite hackers to clog the system. Some forms of attacks are "Denial of Service." They consist of asking for the same web service millions of times per second until the server crashes. There is now the technology to solve this, but it is still an issue to consider in web-service architectures.

- The web service owner helps other sites but gets a little return for doing this.

When Would I Use Service-Oriented Architecture?



Here are some situations this would be useful for:

Example Name	Definition	Real-World Example	Advantages
Package tracking	A package tracking web service is a function that, given a tracking number, describes the itinerary and current location of a package ordered by a client.	<ul style="list-style-type: none"> • FedEx shipping Integration (fedex.com). • DHL Express Package Tracking (dhl.com). 	Retail websites that use this web service do not have to access FedEx or DHL internal systems. The package tracking function resides on the internet and can be freely used by any retail website.
Flight information	A flight information web service accepts as input a flight number. It returns data related to the flight: departure time, estimated arrival time, point of departure, point of arrival, current altitude, etc.	<ul style="list-style-type: none"> • Flight Explorer Professional Edition (flightexplorer.com) 	Travel sites, hotels, and logistics companies that use this product can have immediate information about a flight over the internet without having to access internal systems.
Currency converter	A currency converter web service accepts a source currency and gives its value in a destination currency (for example, "how many US Dollars is a Euro").	Any major bank has this, including the Central Bank in many countries.	Money transactions need the official currency rate at any given time. This currency converter web service publishes any currency conversion given by the Central Bank.

Case-study: Solving a Business Problem With Service-Oriented Architecture



Let's apply what you've learned and see how it was used in a real organization.

GlobalWeather is a global company that sells information about weather conditions. Here are some important facts:

- GlobalWeather is based in the U.S. and has more than 700 employees.
- GlobalWeather sells weather information to companies about countries, regions, cities, or even precise locations defined by latitude and longitude.
- GlobalWeather clients are all over the world and usually consume information using the internet. This is especially useful for transport companies: airlines, bus services, and logistics companies that have a truck fleet to distribute merchandise over a region. Additionally, any companies that coordinate outdoor events.
- Clients need information in real-time.

What's the Business Problem?

GlobalWeather sells information to clients in real-time, but it cannot let all clients access their internal systems because of security constraints.

How can we solve this problem? Let's look at the facts:

GlobalWeather has **thousands of clients**. To buy its services, each client needs to connect to GlobalWeather systems and get information manually. It is very **complex and time-consuming**, especially since the information requested is usually a little piece of information that can be transmitted via the internet.

Additionally, many clients are **websites that need to connect** to the information feed and quickly display weather information in real-time. For example, an airline that is selling a ticket from city A to city B wants to display the weather in city B once the client buys that ticket.

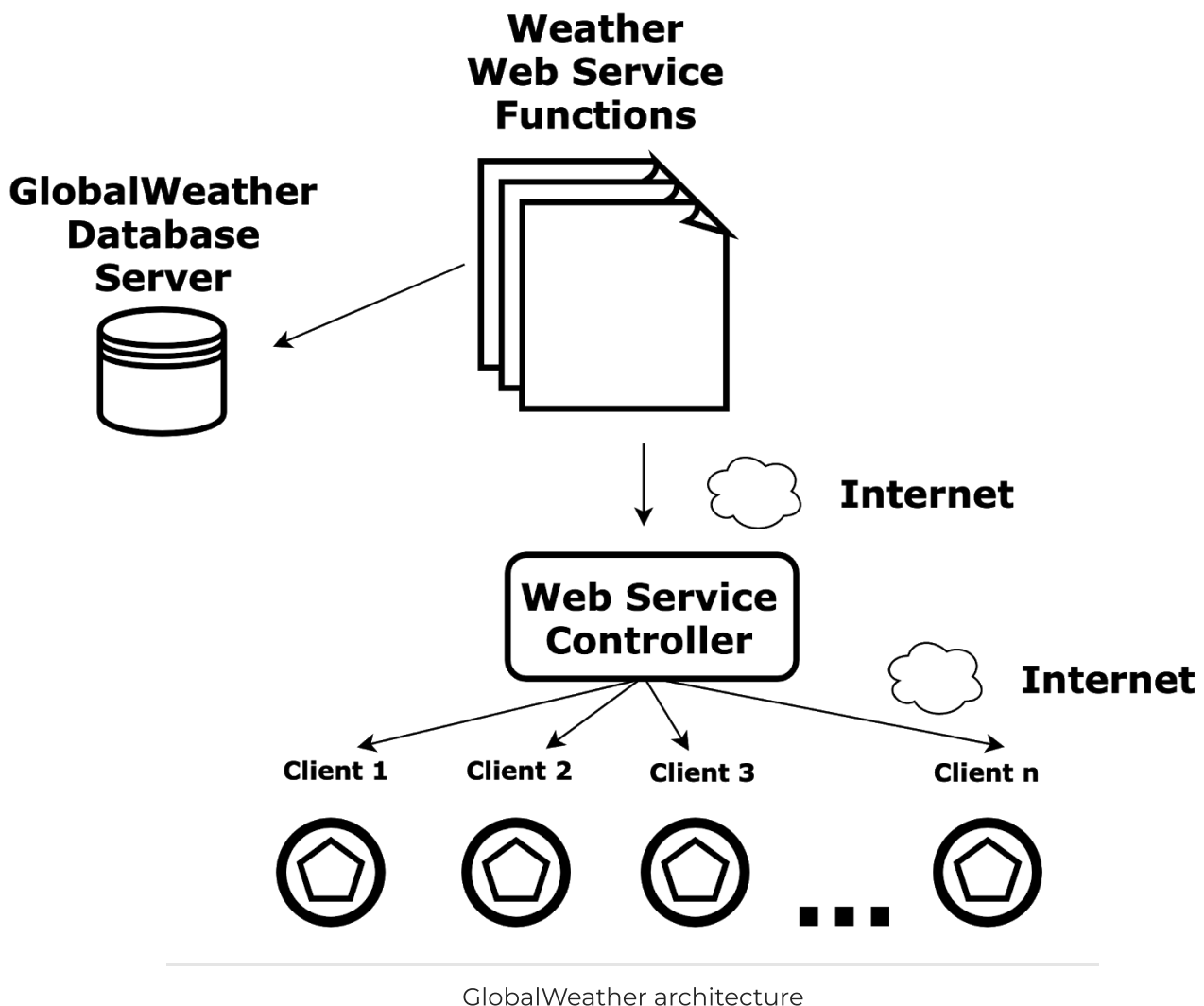
This situation is the right situation for a service-oriented architecture: GlobalWeather can build a **library of functions** and implement it as web services that their clients consume once they pay. This ensures a speedy, real-time service without allowing clients to access internal servers.

What's the Solution?

Let's see the detailed architecture diagram:

Service-Oriented Architecture

GlobalWeather



Let's review each component of the system:

- **Weather web service functions:** This is a library of web services built to serve external websites with weather information such as rain, wind direction and speed, atmospheric pressure, etc.
- **Web service controller:** This module communicates the web service repository with the service requesters. When an external service requester calls a certain function (for instance, flight information, weather information, package status) from the web service repository, the web service controller interprets the call and looks for the function in the web server repository. Then it executes the function and returns a value to the requester.
- **Clients:** These are external applications that request services from the web service repository through the internet. These requests are coded into the requester's application according to a certain syntax published by the service provider.
- **GlobalWeather database server:** This server contains the tables, indexes, and data managed by the system.

Practically speaking, an airline requests, "Give me the weather in city B tomorrow, detailed per hour, including temperature, rain, pressure, and wind." The web service executes this function and returns these values to the airline. All this is done over the internet without accessing GlobalWeather internal servers.

Try it out for yourself!

Now that you've seen one solution, see if you can apply what you've learned to another!

Context: Imagine you're a software architect at an airline. Your boss is Kim, the chief information officer (CIO).

Your mission: You work at a major airport administration department in your country. You have been asked to develop a software system to give the status of a certain flight that is in the air to some possible requesters: airlines, travel sites, hotel sites, etc.

There is a central system at the airport control area that works in real-time: it receives information for the radars and aircrafts as the flights progress.

Now, it's your job to create an architecture for them! Here are some **key questions** you can ask yourself to get started:

1. Many flights are in the air at a certain moment. How can you get their flight status from the central system?
2. How are you going to pass this information to the external requesters?
3. Why does the airline need this system? Who is going to consume this information?

Can you produce an architecture diagram for this business setting?

Once you've written out your version, check it against my **solution**.

Let's Recap!



Architecture Model	Description	Advantages	Disadvantages	When to use it
Service-oriented	A service-based architecture model that allows an external system to use a library of functionalities without having to access the internal systems.	Simple communication: works 100% over the internet. Strong security standards: no client gets into	Clients need internet to use the library. Web service controller may get overloaded and suffer performance problems. Think	When you have multiple clients for a web service. When you need to remotely communicate

the internal systems.

about how many requests the FedEx tracking web service gets per second.

with those clients.

What do you think of this course? Let us know in this [questionnaire](#).

[EVENT-DRIVEN ARCHITECTURE](#)

[PLUG-IN ARCHITECTURE](#)

Teacher

José Esterkin

Software engineer, project manager and trainer. Director of Positive, a project management consulting firm based in Buenos Aires.

- OPENCCLASSROOMS
- OPPORTUNITIES
- SUPPORT
- FOR BUSINESS
- MORE

 English



