Microservice Architecture

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia https://twitter.com/tutorialspoint

About the Tutorial

Microservice Architecture is a special design pattern of Service-oriented Architecture. It is an open source methodology. In this type of service architecture, all the processes will communicate with each other with the smallest granularity to implement a big system or service.

This tutorial discusses the basic functionalities of Microservice Architecture along with relevant examples for easy understanding.

Audience

This tutorial has been prepared for beginners to help them understand the basic concepts of Microservice Architecture.

Prerequisites

This is a very basic tutorial and to make the most out of it, a reasonable knowledge of basic computer programming and Service-oriented architecture is required.

Copyright & Disclaimer

Table of Contents

1. MSA - Introduction

Microservice is a service-based application development methodology. In this methodology, big applications will be divided into smallest independent service units. Microservice is the process of implementing Service-oriented Architecture (SOA) by dividing the entire application as a collection of interconnected services, where each service will serve only one business need.

The Concept of Going Micro

In a service-oriented architecture, entire software packages will be sub-divided into small, interconnected business units. Each of these small business units will communicate to each other using different protocols to deliver successful business to the client. Now the question is, how Microservice Architecture (MSA) differs from SOA? In one word, SOA is a designing pattern and Microservice is an implementation methodology to implement SOA or we can say Microservice is a type of SOA.

Following are some rules that we need to keep in mind while developing a Microservice-oriented application.

Independent: Each microservice should be independently deployable.

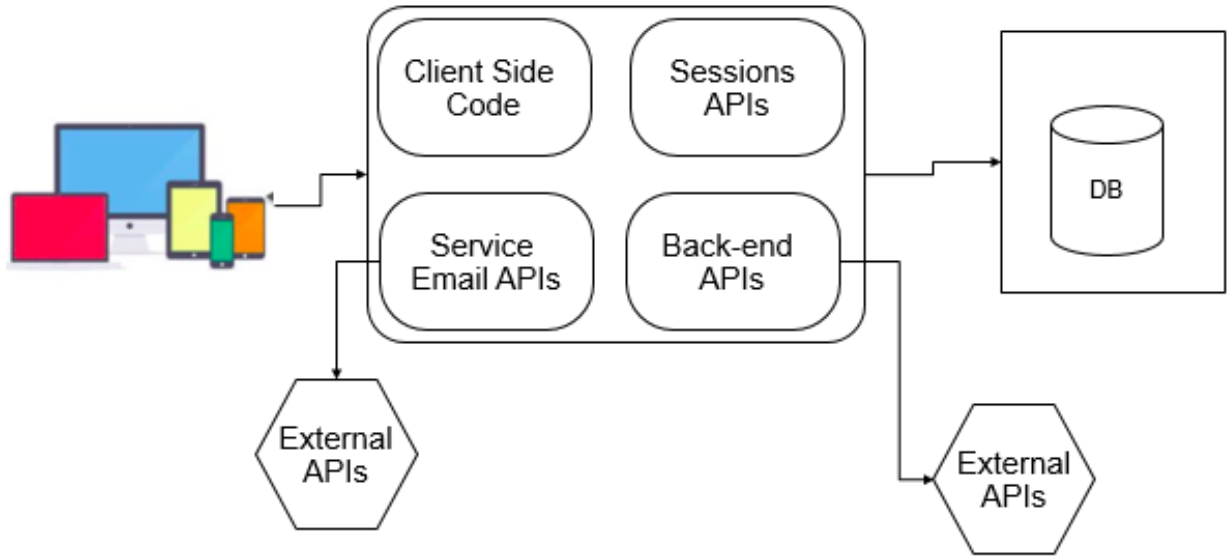
Coupling: All microservices should be loosely coupled with one another such that changes in one will not affect the other.

Business Goal: Each service unit of the entire application should be the smallest and capable of delivering one specific business goal.

Let us consider an example of online shopping portal to understand microservice in depth. Now, let us break this entire E-commerce portal into small business units such as user management, order management, check-in, payment management, delivery management, etc. One successful order needs to proceed through all of these modules within a specific time frame. Following is the consolidated image of different business units associated with one electronic commerce system.

Each of these business modules should have its own business logic and stakeholders. They communicate with other third party vendor softwares for some specific needs, and also with each other. For example, order management may communicate with user management to get user information.

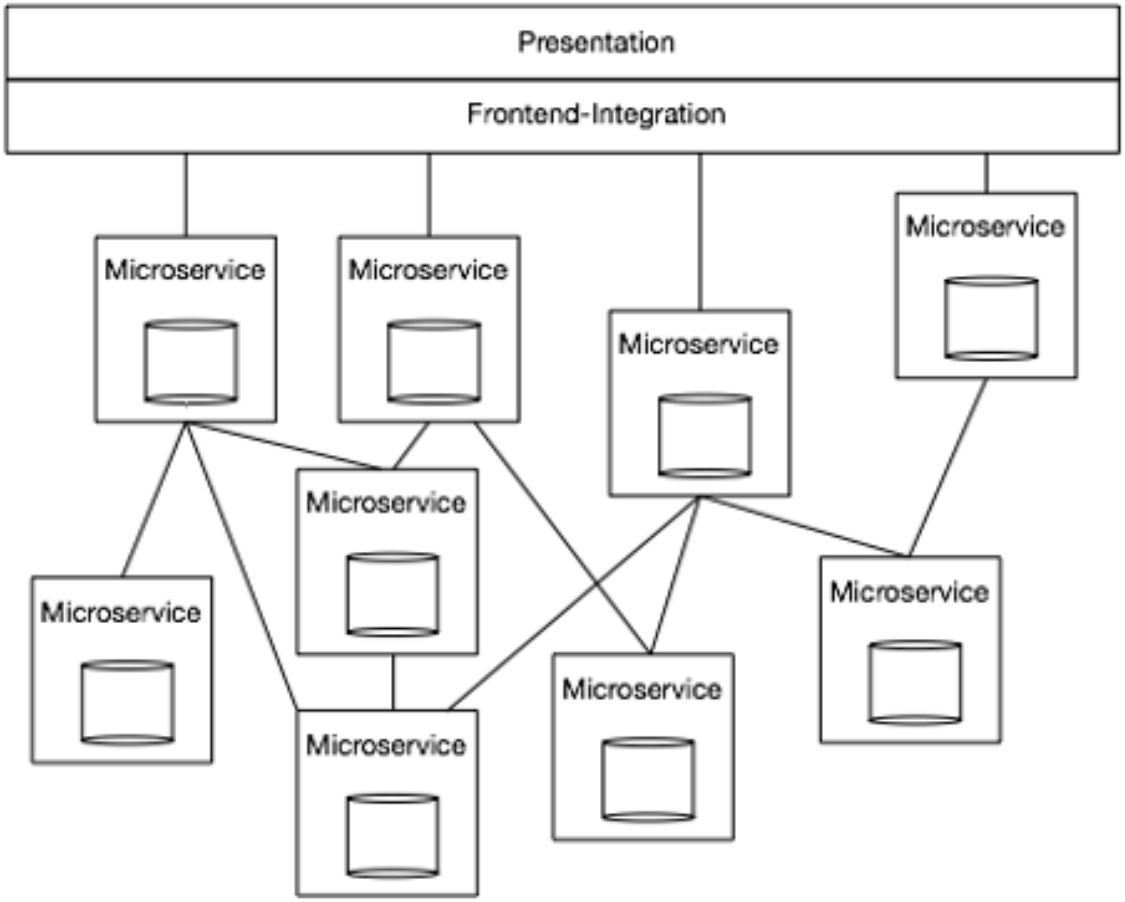
Now, considering you are running an online shopping portal with all of these business units mentioned earlier, you do need some enterprise level application consisting of different layers such as front-end, back-end, database, etc. If your application is not scaled and completely developed in one single war file, then it will be called as a typical monolithic application. According to IBM, a typical monolithic application should possess the following module structure internally where only one endpoint or application will be responsible to handle all user requests.



In the above image, you can see different modules such as Database for storing different users and business data. At the front-end, we have different device where we usually render user or business data to use. In the middle, we have one package that can be a deployable EAR or WAR file that accepts request form the users end, processes it with the help of the resources, and renders it back to the users. Everything will be fine until business wants any changes in the above example.

Consider the following scenarios where you have to change your application according to the business needs.

Business unit needs some changes in the "Search" module. Then, you need to change the entire search process and redeploy your application. In that case, you are redeploying your other units without any changes at all.



Now again your business unit needs some changes in "Check out" module to include "wallet" option. You now have to change your "Check out" module and redeploy the same into the server. Note, you are redeploying the different modules of your software packages, whereas we have not made any changes to it. Here comes the concept of service-oriented architecture more specific to Microservice architecture. We can develop our monolithic application in such a manner that each and every module of the software will behave as an independent unit, capable of handling a single business task independently.

Consider the following example.

In the above architecture, we are not creating any ear file with compact end-to-end service. Instead, we are dividing different parts of the software by exposing them as a service. Any part of the software can easily communicate with each other by consuming respective services. That's how microservice plays a great role in modern web application.

Let us compare our shopping cart example in the line of microservice. We can break down our shopping cart in the different modules such as "Search", "Filter", "Checkout", "Cart", "Recommendation", etc. If we want to build a shopping cart portal then we have to build the above-mentioned modules in such a manner that they can connect to each other to give you a 24x7 good shopping experience.

Advantages & Disadvantages

Following are some points on the advantages of using microservice instead of using a monolithic application.

Advantages

Small in size: Microservices is an implementation of SOA design pattern. It is recommended to keep your service as much as you can. Basically, a service should not perform more than one business task, hence it will be obviously small in size and easy to maintain than any other monolithic application.

Focused: As mentioned earlier, each microservice is designed to deliver only one business task. While designing a microservice, the architect should be concerned about the focal point of the service, which is its deliverable. By definition, one microservice should be full stack in nature and should be committed to delivering only one business property.

Autonomous: Each microservice should be an autonomous business unit of the entire application. Hence, the application becomes more loosely coupled, which helps to reduce the maintenance cost.

Technology heterogeneity: Microservice supports different technologies to communicate with each other in one business unit, which helps the developers to use the correct technology at the correct place. By implementing a heterogeneous system, one can obtain maximum security, speed and a scalable system.

Resilience: Resilience is a property of isolating a software unit. Microservice follows high level of resilience in building methodology, hence whenever one unit fails it does not impact the entire business. Resilience is another property which implements highly scalable and less coupled system.

Ease of deployment: As the entire application is sub-divided into small piece of units, every component should be full stack in nature. All of them can be deployed in any environment very easily with less time complexity unlike other monolithic applications of the same kind.

Following are some points on the disadvantages of microservice architecture.

Disadvantages

Distributed system: Due to technical heterogeneity, different technologies will be used to develop different parts of a microservice. A huge set of skilled professionals are required to support this big heterogeneous distributed software. Hence, distributed and heterogeneity stands as a number one disadvantage of using microservice.

Cost: Microservice is costly, as you have to maintain different server space for different business tasks.

Enterprise readiness: Microservice architecture can be considered as a conglomerate of different technologies, as technology is evolving day-by-day. Hence, it is quite difficult to make a microservice application enterprise ready to compare to conventional software development model.

Microservice Over SOA

The following table lists certain features of SOA and Microservice, bringing out the importance of using microservice over SOA.

Component	SOA	Microservice
Design pattern	SOA is a design paradigm for computer software, where software components are exposed to the outer world for usage in the form of services.	Micro Service is a part of SOA. It is a specialized implementation of SOA.
Dependency	Business units are dependent on each other.	All business units are independent of each other.
Size	Software size is bigger than the conventional software.	Software size is small.
Technology	Technology stack is less than Microservice.	Microservice is heterogeneous in nature as exact technologies are used to perform a specific task. Microservices can be considered as a conglomerate of many technologies.
Autonomous and Focus	SOA applications are built to perform multiple business tasks.	Microservice applications are built to perform a single business task.
Nature	Monolithic in nature.	Full stack in nature.
Deployment	Deployment is time-consuming.	Deployment is very easy. Hence, it will be less time-consuming.
Cost-effectiveness	More cost-effective.	Less cost-effective.
Scalability	Less compared to Microservices.	Fully scaled.

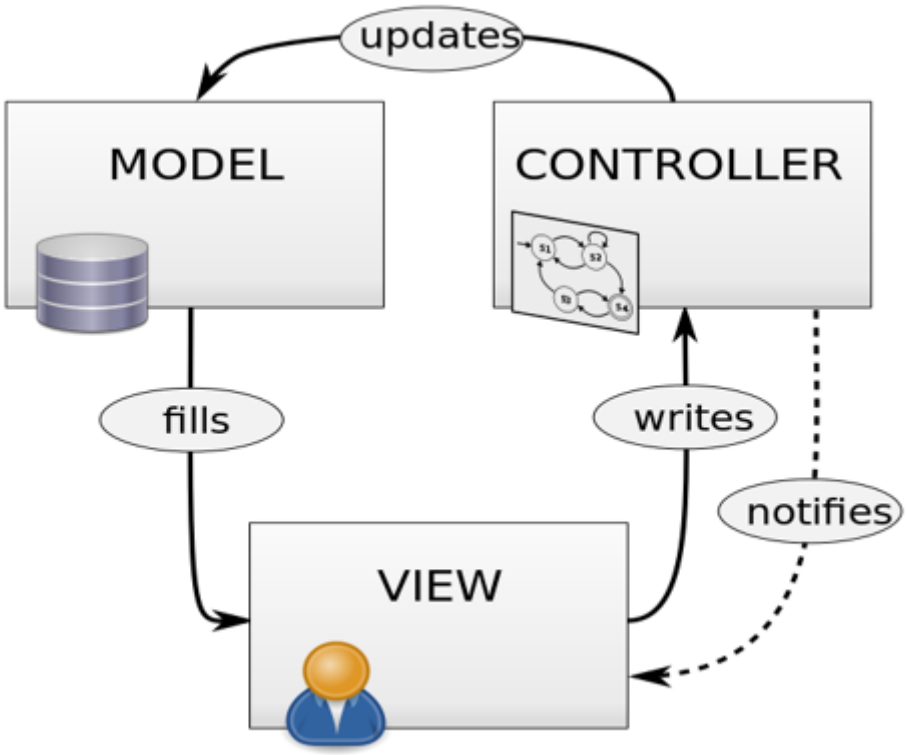
Example	Let us consider one online CAB booking application. If we want to build that application using SOA, then its software units will be - 1. GetPaymentsAndDriverInformationAndMappingDataAPI 2. AuthenticateUsersAndDriversAPI	If the same application is built using microservice architecture, then its APIs will be - 1. SubmitPaymentsService 2. GetDriverInfoService 3. GetMappingDataService 4. AuthenticateUserService 5. AuthenticateDriverService

2. MSA – Scaling

Scaling is a process of breaking down a software in different units. Scaling also defines in terms of scalability. Scalability is the potential to implement more advance features of the application. It helps to improve security, durability, and maintainability of the application. We have three types of scaling procedures that is followed in the industries. Following are the different scaling methodologies along with the corresponding real-life examples.

X-Axis Scaling

X-axis scaling is also called as horizontal scaling. In this procedure, the entire application is sub-divided into different horizontal parts. Normally, any web server application can have this type of scaling. Consider a normal MVC architecture that follows horizontal scaling as shown in the following figure.



As an example, we can consider any JSP servlet application. In this application, the controller controls every request and it will generate view by communicating with the model whenever necessary. Normally, monolithic applications follow this scaling method. X-Axis scaling is very basic in nature and it is very less time consuming. In this methodology, one software will be scaled depending on its different task that the unit is responsible for. For example, the controller is responsible for controlling the incoming and outgoing request, the view is responsible for representing the business functionality to the users in the browser, while the model is responsible to store our data and it works as the database.

Y-Axis Scaling

Y-axis scaling is also called as a vertical scaling that includes any resource level scaling. Any DBaaS or Hadoop system can be considered to be Y-axis scaled. In this type of scaling, the users request is redirected and restricted by implementing some logic.

Let us consider Facebook as an example. Facebook needs to handle 1.79 million users in every second; hence, controlling the traffic is a huge responsibility of Facebook network engineers. To overcome from any hazard, they follow Y-axis scaling which includes running multiple servers with the same application at the same time. Now in order to control this huge level of traffic, Facebook redirects all the traffic from one region to a specific server, as depicted in the image. This transferring of traffic based on the region is called load balancing in architectural language.

This method of breaking down resources into small independent business units is known as Y-Axis scaling.

End of ebook preview

If you liked what you saw…

Buy it from our store @ https://store.tutorialspoint.com