Learning Consolidation

Create Microservices by Using Spring Boot

Learning Objectives

- Explain Monolithic applications
- Explore Microservices architecture
- Create Microservices using Spring Boot

Slide Note

Monolithic software is designed to be self-contained; components of the program are interconnected and interdependent. This is Ok for small sized applications where we are aware of the application size and functionality it provides. Monoliths are OK when we know there would be very little, or no changes in functionality of the application.

This monolith architecture has several benefits:

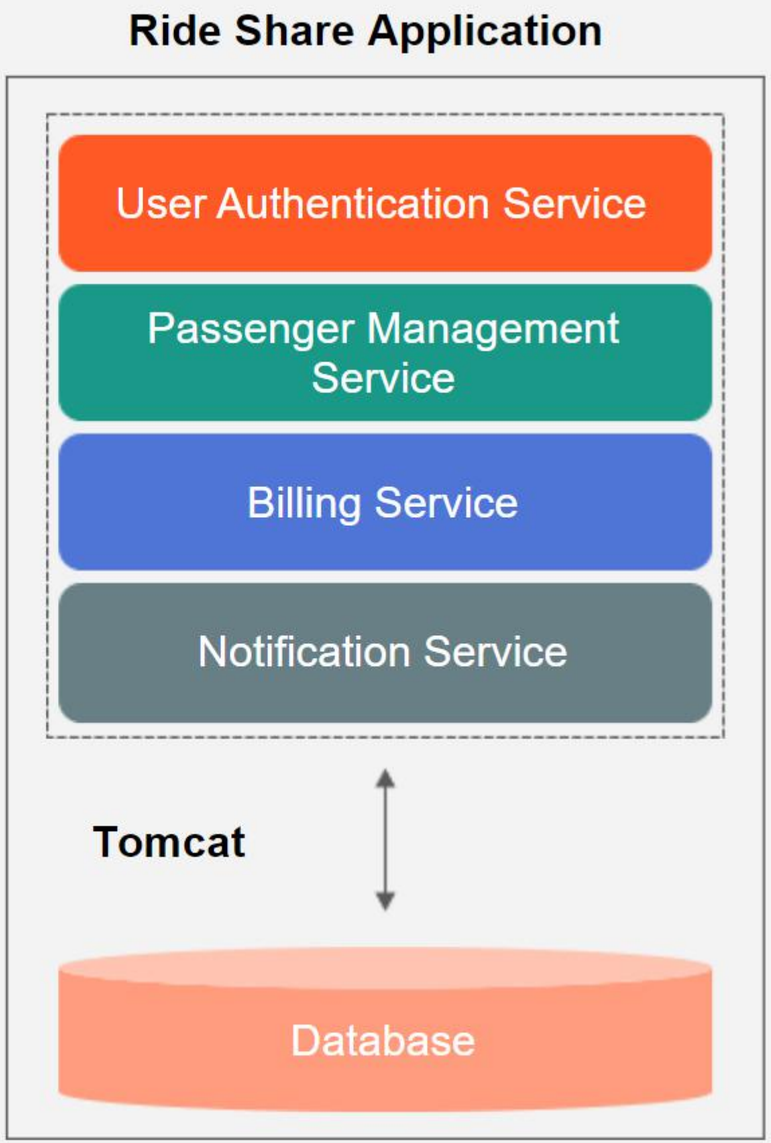
• First it is simple to develop – If you look around current development tools and IDEs, they are inherently designed to support the development of monolithic applications

• Second – It is simple to deploy - you simply need to deploy the WAR file on the appropriate runtime server.

• Also, it is simple to scale - you can scale the application by running multiple copies of the application behind a load balancer.

• So, if you have a small size application, where you expect little or no changes in functionality, monolith is the way to go.

But modern days application have far more dynamics requirements. Applications such as Facebook, Amazon, and Linkedin introduces new features every now and then.



Monolithic Applications

- In software engineering, this application describes a single-tiered software application. Here, the user interface and the data access code are combined into a single program from a single platform.
- A monolithic application is:
  - Self-contained, and independent from other computing applications.
  - Deployed as a single application. In case, it is a Java web application, it will have a single WAR file that runs on a web container like Tomcat.

Slide Note

A severe complexity is related to Change impact.

§ Small changes in an existing component can have rippling affect, and you might end up making big changes in the overall application.

§ This happens because, as I mentioned, in monolithic applications, components are interconnected and interdependent.

§ For example, let us for a minute consider the ride share service as a monolith application.

§ If we change the User Authentication Module to incorporate a new security features – OAuth2 by replacing current JWT authentication, all the interconnected Passenger Management, Notification etc. will require change – some amount or other.

§ Even the database schemas would change.

§ This meant as a rippling effect. Such a rippling effect is common in Monolithic architecture even for very minor updates.

§ At this point, two decisions can be made:

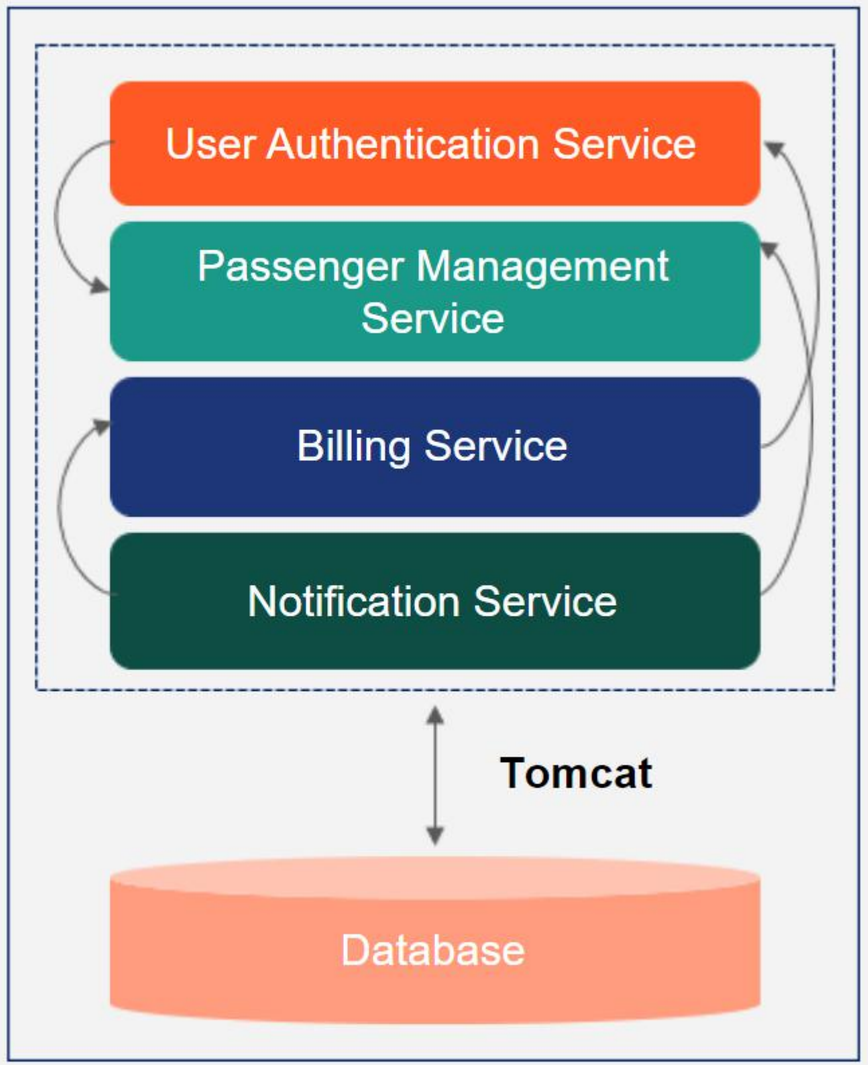
§ One, don't go for the change at all. You will find several legacy applications running with same old functionalities for years.

§ But you can't afford to do so in modern consumer-based applications.

§ New features must be continuously rolled out based on current business demands.

Monolithic Applications - Problems

- The Ride Share application currently has four services.
- Imagine the complexity involved if we need to extend the number of services and add a Driver Management Service, Trip Management Service, etc., in the application.
- The actual problem is not the number of services present in the monolithic application, but the interactions that happen between them.
- A severe complexity is related to the change impact.
- Small changes in an existing component can have a ripple effect because the components are interconnected and dependent on one another.

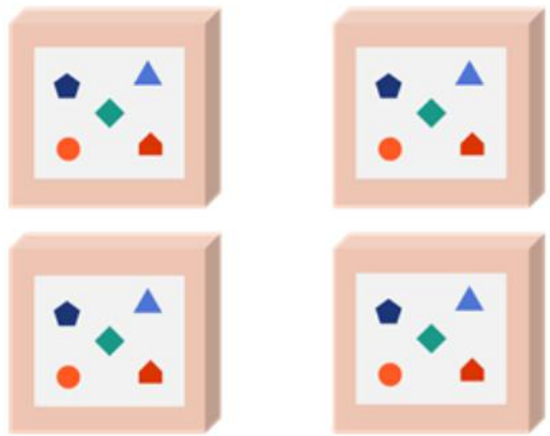


Slide Note

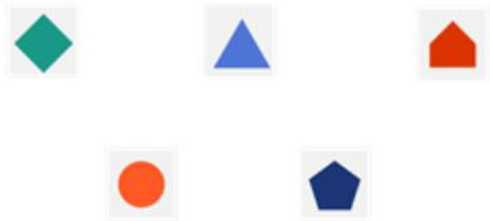
Shifting From Monolithic to Microservices

A monolithic application puts all its functionality into a single process…

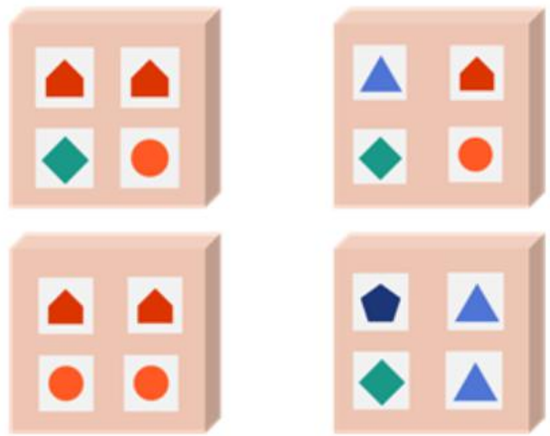
…and scales by replicating the monolith on multiple servers.



A microservices architecture puts each element of functionality into a separate service…



…and scales by distributing these services across servers, replicating as needed.



Slide Note

The client can be a mobile device or a desktop browser client.

The services can effortlessly communicate with one another.

If we want to change the Authentication type from JWT to Oauth, that can be done without changing other microservices in the application.

Change in DB schema of Billing service will not affect other services.

Microservice architecture is a method of enterprise software development, as a suite of independently deployable, small, modular services in which each service runs a unique process and communicates through a well-defined, lightweight mechanism to serve a business goal.

Microservices have been gaining ground in software development since the term came into existence. Microservice architecture is the variant of service-oriented architecture (SOA) for developing large applications where services are grained into chunks as per the business domain. It provides continuous delivery/deployment of complex applications and makes the application easier to understand, develop, test, and is more resilient to architecture erosion.

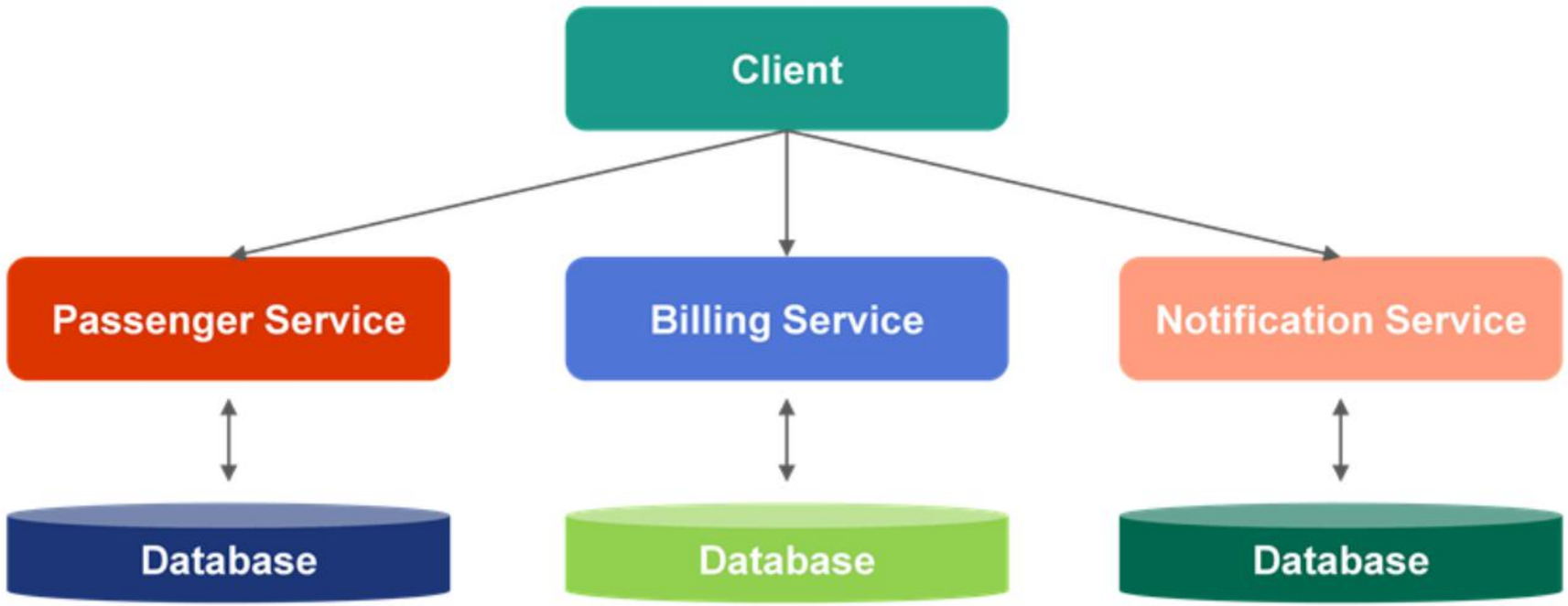
The Microservice architecture provides a new way to weave the existing system in novel ways in order to deliver software solutions quickly. Being one of the hottest topics in the software industry due to its ability to provide modularity, scalability, availability; many enterprise software development companies are keen to adopt it.

Also, Microservices architecture is suitable when it is required to support different types of clients, like desktop browsers, mobile browsers, and native mobile applications, and it is required to integrate with other applications via web services or message broker.

Microservices Architecture

The Microservice Architecture, is an architectural style that structures an application as a collection of small autonomous services modelled around a business domain.

- We can decompose the Monolith for the ride share application as shown below:



Slide Note

Steps to Create Microservices

- The Spring Boot applications created during the earlier sessions such as `CustomerService`, `ProductService`, `UserService` are all individual applications that can be stitched together to form microservices.
- Consider a streaming application that enables users to watch movies on any smart device. The application provides multiple features to all its registered users.
  - Create a `UserService` Spring Boot application that works with the user details
  - Create a `MovieService` Spring Boot application that works with movie details and links the user to the list of the user's favorite movie list .
  - Not all users can add movies to their favorite list or watch later list, thus we need to ensure that only registered users with valid login credentials are able to access this feature.
  - JWT is used for authentication purposes.

Slide Note

Only a registered user can access the data. Many applications use this token access pattern.

§ For example, during online shopping, only if you are a registered user will the application allow you to make payment or place order.

Microservices – An example