ABOUT SHIVSHANKAR SHET

Technology enthusiast currently working in position of a Principal Architect with 12+ years of IT experience in Java/J2EE related technologies.
His main expertise includes building distributed systems, data analytics, cognitive computing.

Limitations of a Monolithic application and need for adapting Micro services Architecture

Posted by: Shivshankar Shet in Software Development May 3rd, 2016

If you get hold of any earlier software architectures, you will see that they have always been built in a Monolithic way.

1. So what is this Monolithic architecture all about?

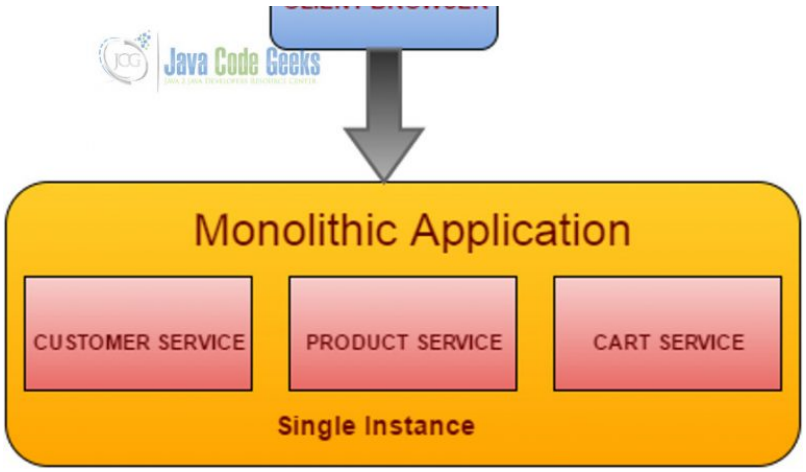
According to Wikipedia, Monolithic definition goes as below:

> *In software engineering, a monolithic application describes a single-tiered software application in which the user interface and 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.*

In layman terms, you can say that is like a big container wherein all the software components of an application are assembled together and tightly packaged .If you take a specific case of Java Programming language, this package can take various formats such as EAR, WAR, JAR etc. which is finally deployed as a single unit on the application server. In business terms you can say that all different business services are packed together as a single unit (tightly coupled). So now since we have covered definition of Monolithic Application, let's delve deeper to understand monolithic architecture with a sample use case.

Let's take a classic Use Case of a shopping Cart Application.

- Please see diagram below for better understanding of this Monolithic Application



Monolithic Architecture

If you see above, the application has following business services which are tightly coupled

1. **Customer Service**
2. **Product Service**
3. **Cart Service**

2. Does Monolithic architecture solves some of classic problems eclipsing an application?

2.1 Agility

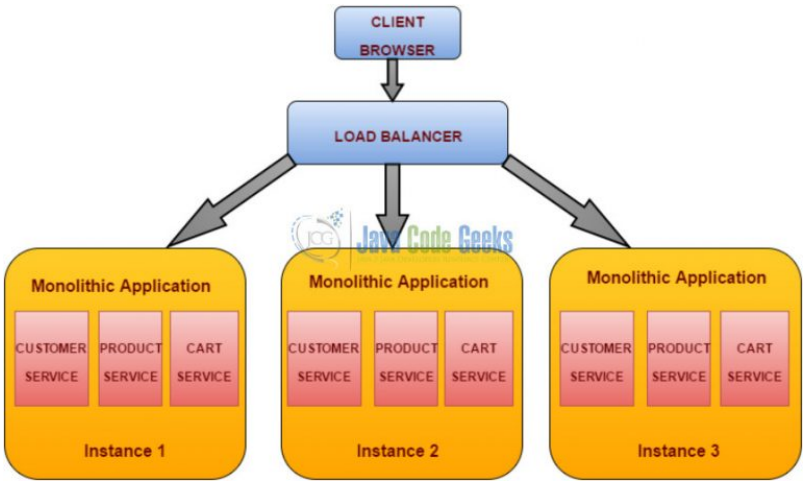
If you see nowadays, business use cases are highly agile, business processes keeps on evolving. For example, in a banking application new regulations keep on coming. Our Monolithic application can definitely accommodate this changes but at the cost of reduced agility because even if a small component in an application has to be changed, the entire application needs to be repackaged and assembled together .so here can definitely say that application has reduced agility since rebuilding the whole application takes a decent amount of time and also slows the pace at which new application features can be delivered.

So we can now say it loses out on the agility front

2.2 Scalability

Let's consider our shopping cart application gains a lot of Customer traction and need arises when our application can`t handle new Customers anymore, so we need to scale our application both vertically and horizontally.

Let's put in a diagram below to understand it better.



Scaled Monolithic Architecture

If you see, just for accommodating more Customers We have created three instances of the same application keeping Customer Service into mind, but resources utilized in other services is getting wasted since they don't have a need to scale.

2.3 Dev Ops Cycle

If you bring in Continuous delivery in to picture then it fails miserably on this side since even for a small change in the application, build time increases tremendously and it definitely decreases frequency of Deployments. So from Dev Ops cycle point of view as well, it's a big No.

There are many more parameters such as **Availability, Fault Tolerance, and Resiliency** on which a monolithic application fails miserably.

3. Now lets see some advantages of a Monolithic architecture.

3.1 Single Deployment Unit

Since entire application is packaged as a single unit, deployment process is relatively easy.

3.2 One Stop shop Codebase

Source Code for entire application lies at a single location and navigating through the code base is pretty easy by making use of some IDE.

3.3 Integrated Development Environment Support

Since Monolithic architecture has been there for a long time , all IDE`s are built basically keeping monolithic architecture in mind meaning if you take an example of a Java Application you can directly create a deployment unit such as WAR , JAR or EAR file from IDE itself.

3.4 Majority

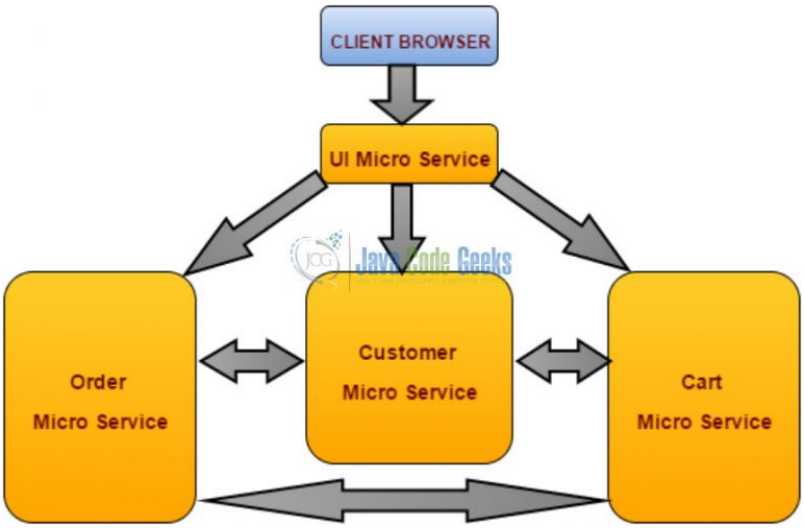
Since Monolithic architecture has been there in the scene from a long time, they have a Majority in software application space. Even if there are some advantages with implementing a monolithic architecture, disadvantages of adopting it definitely outweighs all its advantages.

4. Limitations of a Monolithic architecture has given rise to Micro Services.

Micro Services is an architecture paradigm wherein a monolithic application is decomposed into small tiny micro applications (that why the term **Micro** in Micro Services). We have all been unknowingly using Micro services from a while ago. Imagine a case wherein to support **Location tracking facility** in our application we made use of Google Maps Geo Location API. This Geo Location Service is itself a Micro Service.

Now let's take our Shopping Cart Monolithic Application above and try to decompose it into micro services. Now all its three services Customer Service, Product Service & Cart Service will go in as a Micro Service and in addition we have created a separate UI Micro service facing outside world.

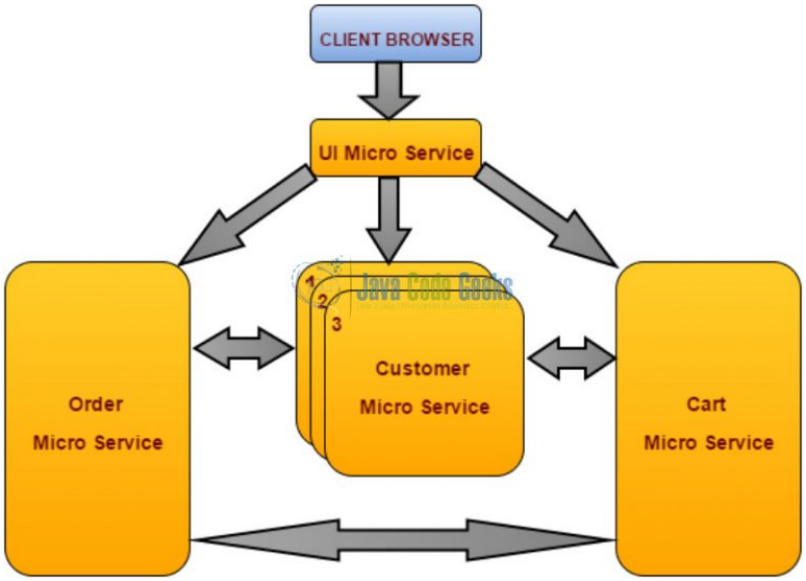
Please see diagram below for better understanding of micro services



MicroService Architecture

So from picture above one can see that we have now decomposed single Monolithic Application into four independent Micro Services. Now we will try to evaluate whether this kind of architecture paradigm solves all the limitations of Monolithic architecture?

Scalability front



Scaled MicroService Architecture

From diagram above, you can see that only **Customer Micro Service** has been scaled since there is a need to accommodate more Customers keeping the other micro services intact thus saving a lot on resources front.

4.2 Agility

Scope Changes can be done in one micro service very rapidly thus decreasing the deployment time and other micro services are not impacted from these changes since they are independent of each other, so it wins on the agility front as well.

4.3 Dev Ops Cycle

If you see , since each component operates independently, continuous delivery cycle definitely reduces a lot and deployment times reduces tremendously hence reduced Dev Ops cycle , so it's a win from Continuously Delivery cycle point of view as well.

So now we have seen that Micro Service architecture solves all the limitations of a Monolithic architecture, let's see all the advantages of adopting a Micro Service Architecture in detail.

5. Advantages of Micro Services Architecture

5.1 Scalability

Each Micro Service can scale independently without affecting other Micro Services. Thus it definitely serves an advantage over monolithic application wherein a lot of resources are wasted for scaling unrequired services since they are all packed together into one single deployable Unit.

5.2 Availability

Even if one service fails, other micro services are highly available and the failed micro service can be rectified very quickly with as minimal downtime as well. Thus it definitely serves an advantage over Monolithic Application wherein entire application has to be brought down.

5.3 Fault Tolerance

Even if one micro service has some faults with regards to say Database connection pool getting exhausted. Thus there is a very clear boundary defined with regards to any fault and unlike the monolithic way, other services operate smoothly and hence only a small part of the application is impacted instead of the entire application bogging down.

5.4 Agility

As mentioned above, changes in a particular micro service can be done very rapidly and can be deployed very quickly which makes it highly suitable architecture for ever changing business requirements meaning highly agile environment.

particular database.

5.6 Maintainability

For each business Service, a separate micro service is created. Thus business code in a micro service is very easy to understand since it caters one business functionality .also since micro service caters to single business functionality, amount of code base is also reduced a lot and this makes it highly maintainable.

5.7 Software Stack agnostic

Since a bigger application is decomposed into n number of smaller micro services, application is not tied to a single software stack and thus different software stacks can be used for different micro services.

5.8 Faster Development

Unlike Monolithic Applications, code changes in micro services can be done very quickly with changes in business requirements which results in faster development cycle.

5.9 Faster Deployment

Since Micro Service caters to a single business functionality, amount of code base is reduced considerable which makes way for faster deployment.

5.10 Clear Separation of Business Concerns

Each Micro Service caters to a unique business functionality and thus there is a very clear separation of business concern between each one of them and thus each micro service can be built in a very robust way.

6. Conclusion

In this article we have seen limitations of a Monolithic Application and need for adopting micro services architecture. In Part 2 of Micro Services Series, we will see how to construct micro services application using Spring Boot.

Tagged with: MICROSERVICES

Leave a Reply

Be the First to Comment!

Notify of new follow-up comments Email ›

Start the discussion