멀티모듈 설계

현재 최근 프로젝트는 단일 프로젝트로 구성되는일이 없으며 일정수준의 트래픽을 감당하기 위해서는 사용자와의 매핑을 담당하는 서버, DB와의 매핑되는서버등 으로 구성되게 됩니다.

그리하여 이번에 프로젝트를 진행할때 단일 프로젝트로 진행하지 않고 서비스를 분리하되 각각의 서비스에 최소한 의존성을 가질수 있는 방법을 보다가 멀티 모듈 관련 내용이 있는 블로그를 보게 되었습니다.

위의 블로그등의 보며 공부하였던 내용인 멀티 모듈의 개념과 장점 그리고 제가 진행하였던 방식을 정리해 보겠습니다.

멀티 모듈 프로젝트?

멀티 모듈 프로젝트라고 처음들었을때 하나의 프로젝트를 여러개의 모듈로 쪼개놓은 프로젝트라고 생각했습니다.

멀티모듈 프로젝트는 공통으로 사용하는 코드를 모아놓고 모듈로써 개발을 진행하는 형태입니다.

멀티 모듈 프로젝트 구조의 중요성

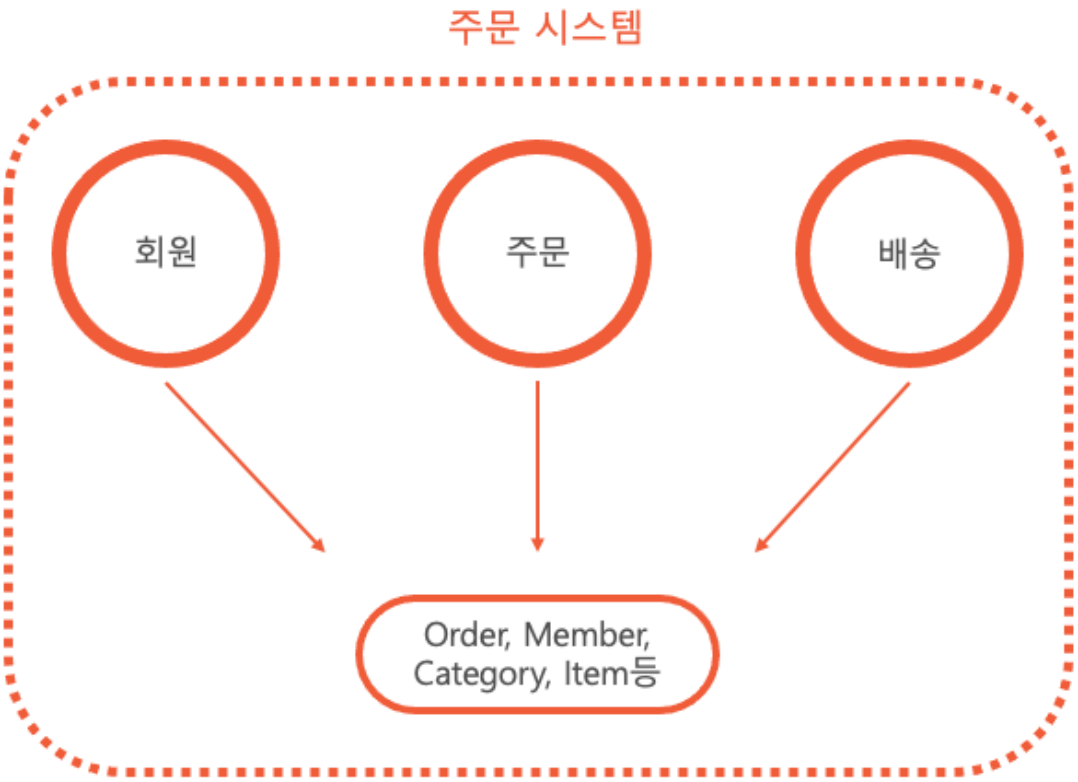
아키텍처는 프로젝트 초기에 이루어져야 하는 일련의 설계 결정이다. -마틴 파울러-

위의 말처럼 **프로젝트 구성는 초창기에 이루어져야 하는 과정**입니다. 시스템이 커질수록 빌드와 배포 프로세스등이 복잡해 지게 되고 이러한 상황에서 구조를 변경할시 어마어마한 힘듬과 고통이 따라오게 됩니다.

또한 **멀티 모듈 프로젝트는 개발 생산성에 막대한 영향을 끼칩니다.** 여러 프로젝트로 진행할시 복사 붙여넣기 등의 작업이 있을 뿐만 아니라 여러개의 IDE를 사용하여 작업을 진행하게 되므로 매우 힘든 과정이 될것입니다.

주문 시스템를 예로 들어보겠습니다.

Order클래스, Member 클래스, Category클래스, Item클래스, Delivery 클래스, OrderStatus 클래스등의 데이터베이스와 매핑되는 도메인 클래스가 들어간다고 가정하였을때 여러프로젝트에서 해당 도메인 클래스들을 사용할려면 위에서 말한 클래스 파일들이 각 해당 프로젝트에 있어야 합니다.



연동되는 프로젝트가 많아진 상황에서 위의 클래스 파일의 수정이 필요할때는 각 프로젝트의 수많은 코드를 수정해야하고 실수할 경우도 많아 집니다.

이런 경우에 하나의 공통 프로젝트를 두고 여러 프로젝트에서 바로 공통프로젝트의 코드를 사용할수 있도록 만든 구조가 멀티 모듈 설계 구조입니다.

저는 여기서 프로젝트를 만들어서 간단하게 진행하였습니다.

그후에 유튜브에 있는 [실전! 멀티 모듈 프로젝트 구조와 설계 | 인프콘 2022] 내용을 보게 되었습니다.

제가 지금 해봤던 프로젝트는 매우매우 작은 개인 프로젝트 수준이므로 위의 강의를 보고 참고 하기는 힘들지만 어쨋든 멀티 모듈에 대한 개념과 이해를 좀더 확실시 할수 있기 때문에 위에 대한 내용도 정리 해봤습니다.

멀티 모듈을 나누는 기준

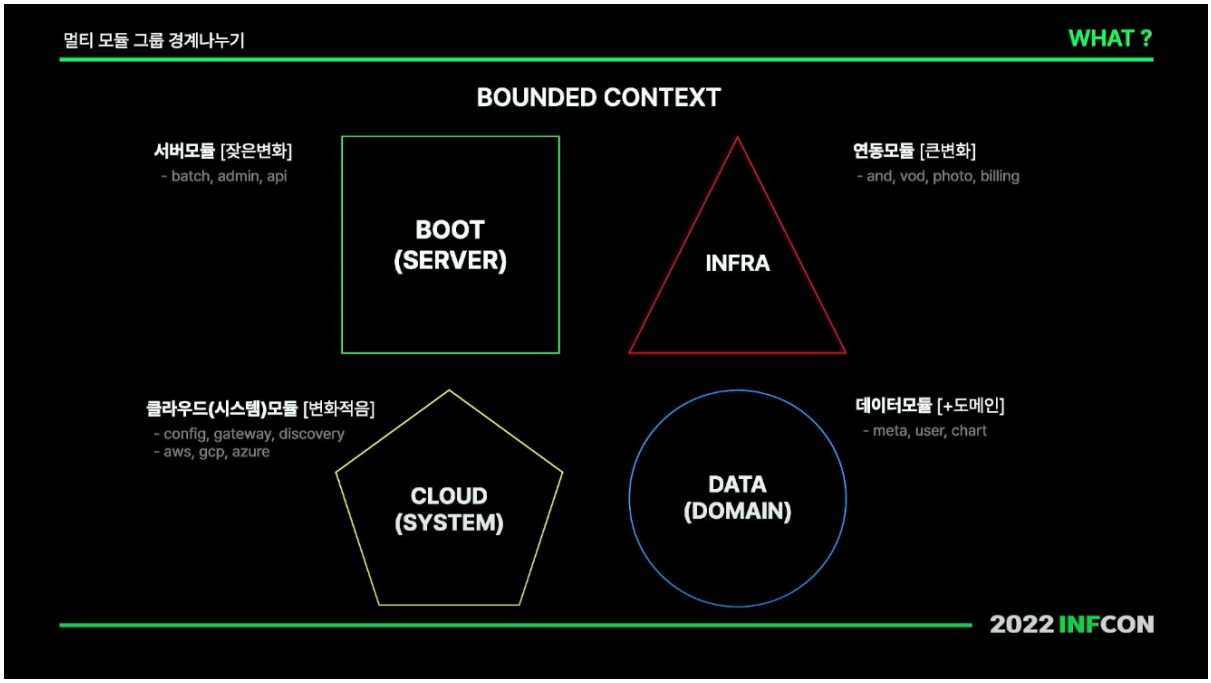
프로젝트는 역할, 책임, 협력 관계가 올바른지 생각해보면서 구조를 설계해야합니다.

저는 기존에는 구글에 공개된 여러 블로그 등을 통하여 공부하면서 Common, Core등의 모듈로 프로젝트를 나눴었습니다.

하지만 위의 강의에서 프로젝트를 진행하게 되다보면 중복된 코드를 제거해야한다는 개발자의 마인드로 인하여 Common과 Core의 사이즈가 너무 커져 있는 문제점을 가지게 된다고 말하셨고

위 강의에서 김대성 발표자님은 common과 core 모듈에서 가지고 있는 잠재적 위험성이 코드의 중복보다 더 크다고 생각하여 common과 core라는 공통 모듈을 삭제하는것이 좋은 방향인것같다 라고 발표하였으며,

바운디드 컨텍스트 (마틴파울러, 2014) * [DDD는 큰 모델을 서로 다른 Bounded Context로 나누고 상호관계를 명시해서 처리합니다. **] *** 라는 기준을 적용하여 모듈을 나누게 되었다고 소개하였습니다.



Inflearn(INFCON 2022, 김대성 Presenter[Naver])

기준을 정하였다면 어떻게 멀티 모듈 프로젝트를 실전에서 구현할까?

프로젝트가 커지고 있다면 다시 경계를 나누고 그 기준으로 소스 저장소를 분리합니다.

INFRA(외부) 라이브러리에는 DATA 관련 구현을 지향한다.

서비스 구현은 각자 역할에 맞게 각각 구현될수 있다(공통으로 한쪽에 구현하지 않습니다.)

시스템 레벨 구현이 실제 서비스 애플리케이션과 밀접하게 연관되지 않도록 격리하거나 전환한다.