이력서

OOO

인적사항	**성명**	
	휴대전화	
	E-mail	
	생년월일	
학력사항		
병역사항	**군별**	육군(의무경찰)
	계급	병장(수경)
	복무기간	0000.00.00 ~ 0000.00.00
웹페이지	**Blog**	https://1-7171771.tistory.com/
	Github	https://github.com/ROMANIAPEOPLE
보유 기술	#JAVA #Spring #MySQL #JPA #MyBatis #Redis #Git	
보유 자격증	**자격/면허명**	정보처리기사
	발급기관명	한국산업인력공단
	취득일	2019.11.22

자기소개서

OOO

대학교 저학년 시절, '프로그래밍' 이라는 것은 코드를 작성해서 원하는 기능만 구현할 수 있다면 끝이라는 착각을 하면서 대부분의 강의 시간을 의미없이 보냈습니다. 프로그래밍을 한번도 제대로 해본적이 없는 저에게 단순히 글만 읽는 학습은 지루하게만 느껴졌고, 이러한 학습을 통해 얻는 이론의 중요성을 느끼지 못했습니다. 무작정 책에 있는 예제를 똑같이 만들어보면서 예제 코드가 정상적으로 작동하면 그 코드를 완벽하게 이해했다는 착각을 했었고 학교 시험 성적 또한 형편없이 저학년 시절을 보냈습니다.

그렇게 군대를 다녀와서 복학 후 첫 수업으로 '자바 프로젝트'라는 과목을 수강하게 되었습니다. 지루하다고 느꼈던 이론 수업을 넘어 무언가를 개발 한다는 생각에 이전과는 다른 흥미가 생기게 되었고 자바 Swing을 이용해 평소에 좋아하던 리듬게임을 만들기로 다짐했습니다. 인터넷에 있는 예제를 보면서 프로젝트를 점점 완성시켜갔고, 3일 밤낮으로 붙잡고 있어도 풀리지 않던 문제를 해결했을 때 극도의 짜릿함을 느꼈습니다. 이 과정에서 내가 원하는 대로 동작하는 프로그램을 만든다는 점이 나만의 집을 건설하고 꾸미는 것과 같은 기분을 느끼게 해 주었습니다.

더 나아가 웹 애플리케이션을 만들어 보고자 하는 목표가 생겼고 대학 동기들과 4명의 팀을 이루어 프로젝트를 시작하게 되었습니다. 하지만 막상 프로젝트를 시작하고 나니 프로젝트의 범위를 너무 크게 정의했다는 사실을 알게 되었습니다. 그 결과 정해진 기간에 맞추기 위해 오로지 기능 구현에 초점을 둔 코딩을 하게 되었습니다. 또한 로컬 환경에서만 애플리케이션을 실행했기 때문에 JPA를 사용하면서 발생하는 N+1 문제와 같은 성능 적인 측면은 전혀 고려하지 않은 채 프로젝트를 완성하게 되었습니다.
이 프로젝트를 통해 기능 구현 만을 위한 코딩은 누구나 할 수 있지만 중요한 것은 향후 유지 보수를 위한 간결하고 깨끗한 코드와 성능의 최적화 라는 것을 배우게 되었고 이러한 노하우는 그동안 중요하지 않게 생각했던 이론적인 부분에서 나온다는 것을 알게 되었습니다.

성을 쌓기 위해서는 기초를 튼튼히 다져야 한다고 생각했습니다. 자바 기본 서적부터 차례대로 책을 읽어가며 공부하기 시작하였고 공부한 내용들을 기술 블로그를 만들어서 정리하였습니다. 모든 내용을 정리하는 것이 아닌 의문을 가졌다가 해소한 내용을 위주로 블로그에 정리하였습니다. 글을 쓰는데 많은 시간이 소요됨에도 불구하고 블로그에 정리하는 저만의 이유는 더 다양한 의문을 가질 수 있기 때문입니다. 머릿속에서만 이해하고 넘어갈 때와는 다르게 글로 정리하다 보면 이해하기 쉬운 글을 위해 그 이유를 찾게 됩니다. 그 과정에서 다양한 의문이 생겨나는 것을 경험했습니다. 또한 타인에게 공부한 지식을 공유한다는 점에서 정확한 내용을 전달할 수 있도록 해당 지식을 더 깊게 공부하는 습관을 가지게 되었습니다.

또한 무작정 외우는 단순 암기는 기억속에 오래 남지 않을 뿐만 아니라 아무리 공부를 좋아하는 사람도 쉽게 지치게 만든다고 생각하기 때문에 프로그래밍 역량을 강화하기 위한 저만의 노하우를 탐구하기 시작하였고 좋은 개발자로 성장하기 위한 저만의 세 가지 원칙을 만들게 되었습니다.

[1. 항상 의문을 갖는 개발자]
좋은 개발자가 되기 위한 첫 번째 자세로, 항상 'Why?'라는 질문을 자기 자신에게 던지는 것입니다.
어떠한 내용을 공부할 때 단순히 그 내용을 이해하는 것만으로 끝내지 말고 '왜 이렇게 동작하는지, 왜 이렇게 설계되었는지'에 대해서 한 번 더 고민해 보는 자세가 필요하다고 생각합니다. 해당 기술이 어떤 배경에서 탄생하게 됐는지, 장단점은 무엇인지, 써본 사람들은 어떻게 평가하는지를 살펴보며 기술에 대한 전반적인 이해를 위해 노력한다면 그 기술의 기능적인 부분이나 동작 원리를 조금 더 쉽고 깊게 이해할 수 있으며 완전히 내 것으로 만드는데 도움이 될 수 있다고 생각합니다.

[2. 실천하고 기록하는 개발자]
어떠한 공부를 하더라도 직접 해보지 않으면 그것을 완전히 이해할 수 없습니다. 때문에 직접 실험하고 실천함으로써 지식을 습득하는 것이 중요하다고 생각합니다.
예를 들어 객체 지향적인 설계를 하기 위한 객체지향의 5원칙 SOLID를 책을 통해 공부했지만, 막상 코드에 적용하려고 하니 뜻대로 되지 않았습니다. 이를 위해 숫자 야구게임과 같은 작은 게임을 객체 지향의 원칙에 따라 스스로 구현해보면서 객체 지향의 전반적인 흐름을 이해할 수 있었습니다.

또한 프로젝트를 진행할 때나 개인적인 공부를 할 때 '문제 해결' 에만 초점을 두지 않고 원인을 파악하며 그 과정을 기록함으로써 내가 어떤 부분에서 실수했는지 반성하고 되짚어 보는 습관이 필요하다고 생각합니다.
항상 새로운 기술이 쏟아지는 환경에서 실천과 기록을 통해 지식을 스스로 탐구하는 습관을 가진다면 해당 기술을 더 깊게 이해하고 기억할 수 있는 좋은 개발자가 될 수 있다고 생각합니다.

[3. 소통하는 개발자]
개발자는 혼자 일하는 직업이 아닌 전적으로 협업하는 직업입니다. 팀 프로젝트 시 개인마다 요구 사항이 조금씩 달라 마찰이 일어날 수도 있습니다. 이런 상황이 발생하게 된다면 무조건적으로 자신의 생각이 옳다는 오만한 생각은 지양하고 열린 마음으로 상대방의 의견을 수용하는 자세가 필요합니다. 저는 이러한 상황을 적극적인 코드 리뷰를 통해 해결하고 있으며 이 과정이 서로에게 부족한 점을 채워줄 수 있는 보완재 같은 역할을 할 수 있다고 생각합니다.
또한 이와 별개로 업무 외 시간에 동료들과 꾸준한 소통을 나누는 것도 개발자에게 필요한 역량이라고 생각합니다. 자신의 의사를 전달하거나 단순 소통을 할 때 상대방과의 친밀감이나 신뢰 관계가 바탕이 된다면 더욱 수월한 소통이 가능하다고 생각하기 때문입니다.

포트폴리오:

OOO

Shoe-auction

2021.01 ~

#JAVA8 #Spring #Spring Data JPA # Redis #MySQL #AWS #Jenkins

https://github.com/f-lab-edu/shoe-auction

프로젝트 개요

최근 패션 업계에서 한정판 스니커즈 리셀(resell · 재판매)이 화두로 떠오르고 있습니다. 코로나로 인해 의류 매출은 전반적으로 주춤한 가운데 운동화는 꾸준히 매출이 상승하고 있을 뿐만 아니라 제한된 수량으로 발매되는 신발의 경우 웃돈을 얹은 거래가 활발하게 이루어지고 있습니다. 하지만 이러한 개인간의 신발 거래는 대부분 중고거래 카페 또는 모바일 어플리케이션에서 행해지고 있고, 개인과 개인간의 거래이기 때문에 가품 문제 및 사기의 온상지가 될 수 있다고 생각합니다. 이러한 점을 고려하여 shoe-Auction은 개인과 개인 사이에 중개자가 끼어들어 거래를 형성하고, 입찰 시스템을 도입하여 합리적인 거래를 유도하는 신발 거래 전문 플랫폼 API 서버입니다.

#확장에 용이한 구조 설계

본 프로젝트는 단순한 기능 구현뿐만이 아닌 애플리케이션의 사용자가 증가함에 따라 발생할 수 있는 트래픽 문제를 고려하여 진행하였습니다. 고사양 하드웨어 장비를 통해 기존 서버 자원의 성능을 증강시키는 Scale-up 방식의 경우 확장의 한계와 서버에 문제가 발생할 경우 서버가 복구될 때까지 서비스를 중단해야 하는 단점이 있다고 판단하였습니다. 따라서 비슷한 사양의 서버를 추가해 트래픽을 분산시키는 Sacle-out 방식을 고려하여 프로젝트를 진행하였습니다.

#세션 정합성 문제 해결하기

분산 서버 환경에서 발생할 수 있는 로그인 정보를 담은 세션 데이터의 정합성 문제를 해결하였습니다. Sticky session, Tomcat clustering 등 여러 가지 해결책들을 후보에 두고 각각의 trade off을 고려하여 가장 적합하다고 판단한 솔루션인 세션 저장소 서버를 구축하는 방법을 선택하였습니다. 본 프로젝트에서는 로그인

정보를 비교적 빠른 속도로 저장하고 조회할 수 있는 In-memory DB 사용을 고려하였으며 Spring에서 간단한 의존성 빌드를 통해 사용할 수 있는 Spring-redis-session을 사용함으로써 개발의 생산성을 증진시킬 수 있었습니다.

#캐시 적용으로 인한 응답속도 개선

애플리케이션의 사용자가 증가함에 따라 DB에 접근하는 횟수 또한 증가하게 되며, 이는 애플리케이션의 성능 저하로 이어지게 됩니다. 이러한 문제점을 개선하기 위해 자주 조회되는 비즈니스 로직에 캐시를 적용하여 읽기 성능을 개선하였습니다. 캐시를 적용하기에 앞서 로컬 캐싱 전략과 글로벌 캐싱 전략 중 어떤 전략이 더 적합한지 충분히 고민해 보는 시간을 가진 후 분산 서버 환경에서 서버 간 데이터를 쉽게 공유할 수 있고 캐싱 된 데이터의 일관성을 유지할 수 있는 글로벌 캐싱 전략을 선택하여 Redis를 캐시 저장소로 활용하였습니다.

#효율적인 Redis 활용을 위한 메모리 관리

본 프로젝트에서는 응답속도 개선을 위한 캐시 적용과 세션 데이터 불일치 문제를 해결하는 과정에서 Redis를 활용하였습니다. 애플리케이션을 운영하는 동안 Redis를 효율적으로 활용하기 위해 꼼꼼한 메모리 관리를 진행하였습니다.

Redis의 maxMemory를 설정하여 운영체제의 가상 메모리인 swap 영역까지 사용하는 것을 방지하였고, Redis가 지원하는 다양한 cache eviction 정책을 비교하여 해당 애플리케이션에 가장 알맞은 정책인 LFU를 적용하였습니다. 또한 캐시 데이터를 저장하는 서버와 세션 데이터를 저장하는 서버를 분리하여 서버 부하를 방지하고 보다 더 효율적인 메모리 관리를 할 수 있도록 구현하였습니다.

#중복되는 부가기능 처리

본 프로젝트는 핵심 서비스에 한하여 회원제로 운영되고 있습니다. 따라서 대부분의 서비스를 이용할 때 로그인 여부를 확인하는 작업이 필요하며, 일부 서비스는 현재 로그인 되어있는 사용자를 불러오는 작업까지 필요하게 됩니다. 해당 작업들은 대부분의 비즈니스 로직에서 중복적으로 나타나는 기능으로, 중복되는 부가기능을 따로 처리하기 위해 Interceptor와 ArgumentResolver를 활용하여 비즈니스 로직을 구현하는 데 있어서 보다 더 객체지향적 설계를 할 수 있도록 노력하였습니다.

#JPA 성능 최적화

JPA를 사용하면서 발생할 수 있는 문제점 중 하나로, 사용하기 편리한 만큼 성능적인 부분을 쉽게 간과할 수 있다고 생각했습니다. 이번 프로젝트에서는 모든 Fetch 전략을 LAZY로 설정하여 해당 로직에 반드시 필요한 쿼리만 실행되도록 구현하였으며 N+1 문제가 발생하는 부분을 찾아 리팩토링 하는 과정을 진행하여 불필요한 쿼리가 실행되는 것을 최소화할 수 있도록 구현하였습니다.

#Jenkins를 이용한 CI/CD 구축

NCP 이용해 Jenkins 서버를 띄우고 CI/CD를 적용하였습니다. 본 프로젝트는 협업을 통해 진행되는 만큼 코드의 충돌이나 버그가 없도록 구현하는 것이 상당히 중요한 부분이었습니다. 따라서 feature branch를 포함한 모든 branch에 git push가 발생하거나 PR이 진행되었을 때마다 GitHub에서 Webhook을 보내 테스트와 빌드가 자동 실행되도록 멀티 브랜치 파이프라인을 설계하였습니다. 또한 자동 배포를 위해 develop branch에서 git push가 발생하면 자동으로 원격 서버에 있는 배포 스크립트가 실행되도록 설정하였습니다.
Jenkins에서 발생되는 모든 결과는 협업 중인 개발자와 함께 속해 있는 Slack 채널에 알림을 보내도록 설정하여 즉각적인 피드백을 진행하고 있습니다.

#MySQL Replication을 통한 애플리케이션 성능 개선

본 프로젝트는 사용자가 증가함에 따라 많은 양의 트래픽이 발생한다는 가정 하에 진행중입니다. 따라서 데이터베이스의 부하로 인한 병목 현상을 개선하기 위해 Replication을 적용하였습니다. 프리티어의 사양 문제로 인해 AWS RDS대신 무료 크레딧을 사용할 수 있는 NCP로 서버 두 대를 띄어 직접 MySQL Replication을 적용하였습니다. 이후 동적 라우팅 설정을 통해 Read 작업은 Slave 서버에서, Write/Read 작업은 Master 서버에서 수행하도록 구현하였습니다.

#AWS Lambda를 활용한 이미지 리사이징

프로젝트에 사용되는 상품과 브랜드의 이미지 저장소로 높은 확장성과 복구 기능을 가진 AWS S3를 선택하였습니다. 많은 양의 이미지를 관리하기 위해 외부 저장소를 선택한 만큼 추가적인 통신 과정은 어쩔 수 없는 비용이지만, 애플리케이션 내에서 동일한 이미지가 다양한 크기로 사용되는 점을 고려하여 이미지 리사이징을 적용하고 요청마다 적합한 사이즈의 이미지를 사용할 수 있도록 구현하였습니다. 이 과정에서 AWS Lambda를 활용하여 서버리스한 방식으로 리사이징 작업을 처리하도록 구현하였고, 이로 인해 애플리케이션 서버의 추가 작업 부담을 덜고 기존 방식을 그대로 사용하여 주 비즈니스 로직 구현에 더욱 집중할 수 있었습니다.

#Spring Rest Docs를 활용한 API 문서화

API 문서를 GitHub wiki 또는 google docs 등을 이용하여 수동으로 관리하게 된다면 애플리케이션을 수정해야 하는 상황이 발생했을 때 API 문서도 수동으로 변경하는 작업을 거쳐야 합니다. 또한 이러한 작업은 자동화 작업이 아니기 때문에 실수할 가능성이 존재합니다. 이러한 문제를 사전에 방지하기 위해 API 문서의 자동화를 적용하게 되었습니다. Spring 환경에서는 API 문서 자동화에 주로 Swagger와 Spring rest docs가 사용됩니다. 각각 장단점을 비교해 본 결과, Swagger를 사용하게 되면 Controller Layer에 비즈니스 로직과는 전혀 관련 없는 문서화를 위한 annotation들이 추가되어 프로덕션 코드의 오염을 야기한다고 판단하여 Spring rest docs로 API 문서의 자동화를 진행하였습니다.

#테스트 코드 작성

어떠한 프로젝트를 진행하더라도 완성된 후에 발생할 수 있는 요구 사항의 추가나 변경을 고려해야 합니다. Service Layer와 Controller Layer에 모든 case마다 단위 테스트를 작성함으로써 코드의 추가 또는 변경이 있을 경우 다른 코드에 미치는 영향을 빠르게 파악하여 개발의 생산성과 유지보수성을 증진시킬 수 있도록 구현하였습니다.

#클린 코드

비즈니스 로직을 구현하는 과정에 있어서 최대한 간결하고 읽기 좋은 코드를 만들기 위해 노력하였습니다. Modern Java In Action을 읽으며 학습했던 내용을 바탕으로 Stream API를 최대한 활용하여 코드 라인을 최적화하였고, 코드의 가독성을 증진시키기 위해 uncheckedException 예외 처리 전략을 사용하여 if문 사용을 최소화하였습니다. 또한 다양한 객체지향 관련 서적을 읽으며 객체지향의 SOLID 원칙에 입각하여 비즈니스 로직을 설계하기 위해 노력하였습니다.

#Git을 활용한 버전 관리

Git과 GitHub를 적극적으로 활용하여 프로젝트를 진행하였습니다. 해당 프로젝트는 저를 포함한 백엔드 개발자 1명과 함께 진행한 프로젝트입니다. 따라서 일관적으로 여러 상황에 대처할 수 있도록 git-flow 브랜치 관리 전략을 사용하였으며, 예상치 못한 상황에서 발생할 수 있는 side-effect 방지를 위해 새로 작성한 코드들은 서로 코드 리뷰를 주고받는 과정을 거쳤습니다. 리뷰어의 approve가 완료되지 않은 PR에 대해서는 Merge가 불가능하도록 설정하여 코드의 질을 높이는데 기여하였으며, google checkStyle로 코딩 컨벤션을 통일하였습니다.

#nGrinder를 활용한 성능테스트

#MySQL 실행계획 분석을 통한 성능 튜닝

HOT - THINK

2020.03 ~ 2020.06

#JAVA *#Spring* *#MySQL* *#Spring Data JPA*

개요

'wadis'와 같은 자신의 아이디어로 탄생한 아이템들을 판매할 수 있는 플랫폼을 만드는 것을 목표로 시작한 프로젝트입니다. 자유롭게 자신의 아이디어를 등록하고 일정량 이상의 추천 수를 받게 되면 'HotThink'라는 곳에 글을 이동시킴으로써 상품가치가 있는 아이디어로 인정받게 되는 과정의 로직을 담았습니다.
총 3개의 핵심 게시판은 'FreeThink', 'HotThink', 'RealThink'로 '좋아요' 버튼을 통한 추천 기능과 1:1 채팅 기능, 특정 유저 팔로우 기능을 구현하여 마음에 드는 아이디어를 구매하고 판매할 수 있는 환경을 만들었습니다. 또한 거래를 위한 포인트 충전 기능과 무분별한 회원가입을 방지하기 위한 휴대폰 인증 기능을 추가하여 원활한 서비스를 지원하도록 구현하였습니다.

#협업 툴 사용

본 프로젝트는 총 4명의 백엔드 개발자가 함께 진행한 프로젝트입니다. 코로나19의 영향으로 전면 비대면으로 진행하였기 도메인 설계부터 프로젝트가 완료될 때까지 '구글 행아웃' 화상 회의 툴과 Slack을 통해 진행하였습니다. 적극적으로 툴을 활용한 결과 비대면 상황에서도 불구하고 원활한 의사소통이 진행될 수 있었습니다.

#결제시스템 도입

본 프로젝트의 주된 서비스는 아이디어의 판매와 구매입니다. 따라서 아이디어 구매를 위한 결제 기능이 필요했습니다. 결제 기능을 구현하는 가장 쉬운 방법으로 '다날'과 같은 API를 도입해서 구현할 수 있지만 해당 방법은 실제로 사업자 등록이 필요했기 때문에 다른 방법을 탐구하였습니다. 그 결과 '아임포트'라는 API를 사용하여 카카오페이로 포인트를 충전할 수 있는 기능을 구현하였습니다.

#본인 인증 시스템

사이트를 운영할 때 가장 중요한 부분 중 하나는 실제 회원과 유령 회원을 구분하는 것 이라고 생각합니다. 특히 결제 기능이 포함된 애플리케이션에서는 본인인증을 도입하여 무분별환 회원가입을 방지해야 합니다. 본 프로젝트에서는 휴대폰 인증 기능을 구현하기 위해 'coolSMS'라는 API를 사용하였으며 랜덤 난수를 생성하여 사용자가 입력한 번호와 생성된 인증번호를 비교하여 본인인증 절차를 구현하였습니다