이력서:

OOO

인적사항	**성명**	
	휴대전화	
	E-mail	
	생년월일	
학력사항	**2014.03 ~ 2021.02**	
	OOO대학교 소프트웨어공학과 졸업	
	2011.03 ~ 2014.02	
	OOOO 고등학교 졸업	
병역사항	**2015.08 ~ 2017.05**	
	육군 병장 만기전역	
웹 페이지	**Blog**	https://kerobero.tistory.com
	Github	https://github.com/Hosick
	LinkedIn	
보유 기술	#JAVA #Spring #MyBatis #JPA #MySQL #Redis #Linux #Git #Aws	
보유 자격증	**자격/면허명**	정보처리기사
	발급기관명	한국산업인력공단
	취득일	2019.11.22

자기소개서:

OOO

물음표로 하는 성장

저는 배움에 있어서 항상 스스로에게 물음을 던지며 발전하려고 노력합니다. 학부생 시절 개발에 대한 열정을 가지고 학교 수업에서 얻은 지식을 기반으로 팀 프로젝트를 진행하였습니다. 많은 시간과 노력을 들여서 어느 정도 만족할 만한 결과물을 만들 수 있었지만 들인 노력과 산출된 결과물에 비해 스스로 판단했을 때 실력이 월등히 늘었다고 생각되지 않았습니다. 그 이유에 대해 오랜 시간 고민해본 결과 저는 눈에 보이는 결과물, 즉 비즈니스 로직이 원하는대로 동작하는지 그 자체에만 집중하고 있었고 해당 코드가 동작하는 기술적인 원리와 이유에 대한 물음은 등한시하며 적극적으로 궁금해하지 않았기 때문이라는 생각을 했습니다.

지금까지 개발에 임했던 방식이 잘못되었음을 깨닫고 그 때부터 결과가 아닌 본질을 좇기 시작하였습니다. 언어 기본서부터 다시 읽어가며 동작 과정과 원리에 대해서 꼼꼼히 학습하였고, 그 동안 사용했던 기술들에 대한 책 들을 읽으며 과거의 지식들을 하나씩 교정해 나아가기 시작했습니다. 그 결과 이전에 작성했던 코드들을 어떠한 이유로 작성해야 했으며, 왜 잘못 작성되었는지 하나씩 새롭게 보이기 시작했습니다. 덕분에 이후 프로젝트를 진행할 때에는 이전보다 훨씬 효율적이고, 문제가 발생할 위험이 적은 방식으로 로직을 작성할 수 있게 되었습니다. 이러한 결과로 이전과 달리 성장에서 오는 성취감을 크게 느낄 수 있었으며 더불어 더욱 열정적으로 학습에 임할 수 있는 계기를 얻을 수 있었습니다.

속도보다 품질

저는 개발을 할 때 정확하고 꼼꼼한 구현을 가장 중요하게 생각합니다. 단지 동작만 하는 어플리케이션은 누구나 만들어 낼 수 있지만 신중함 없이 작성된 코드에서는 필시 예상하지 못한 문제들이 발생할 가능성이 높으며, 이는 실제 비즈니스 수준에서는 서비스에 심각한 피해를 입힐 수 있기 때문입니다.

따라서 저는 프로젝트 진행 시 기능을 구현할 때에도 해당 코드에서 의도치 않은 사이드 이펙트가 발생할 가능성은 없는지, 여러 방법 중 트레이드 오프를 고려했을 때 최선의 방법인지, 확장성과 유지보수를 고려했을 때에는 어떠한지 등 여러 방면에서 디테일한 고민들을 하려고 노력하고 있습니다.

물론 이러한 지향점 때문에 기능 하나당 소요되는 개발 기간이 남들보다 빠르다고 할 수는 없을 것입니다. 하지만 이전의 사례를 돌아봤을 때 깊은 고민을 하며 작성한 코드들은 시간에 쫓겨 급하게 작성한 코드들에 비

하여 문제가 발생하는 빈도가 더 낮아져 불필요한 추가 리팩토링 시간이 줄어 오히려 전체 개발 시간은 더 단축되는 경우가 많았습니다. 따라서 완성도 높은 결과물을 기대하기 위해서 고민하는 시간은 필수적으로 요구된다고 생각합니다.

커뮤니케이션을 중시

대부분의 개발자는 기업이나 프로젝트에 속해서 다른 사람과 협업을 통해서 결과물을 만들어냅니다. 이 과정에서는 개발자들 간 또는 고객과의 커뮤니케이션은 필수적인 부분이며, 이러한 커뮤니케이션 능력 또한 개발자에게 간과해서는 안 되는 덕목이라고 생각합니다. 아무리 코드 작성 능력이 뛰어나더라도 개발 의도를 잘못 파악하여 기존의 설계와 다른 방향으로 구현 한다거나 또는 팀원과 기술적 소통을 할 때에 자신의 생각과 의도를 알아들을 수 없게 전달하는 경우에는 오히려 팀에 악영향을 끼칠 수 있다고 생각하기 때문입니다.

저는 이러한 이유로 협업 시 커뮤니케이션에 소홀해지지 않도록 최대한 주의하고 있습니다. 개발 방향에 대한 더 나은 의견이 떠오를 때에는 근거를 확실하게 정리하여 의견을 피력하려 하며, 반대로 팀원들의 의견을 최대한 객관적이고 적극적으로 수용하는 태도를 가지려고 노력하고 있습니다.

저의 경험상 위와 같이 건전한 커뮤니케이션을 위해 노력했을 때 다른 팀원들 또한 본인의 의견을 스스럼없이 제시하는 분위기가 조성되었으며, 저의 의견에도 더욱 귀 기울여준다는 점을 느낄 수 있었습니다. 결과적으로 이는 분명히 팀원들의 방향성 통일과 팀 분위기에 긍정적인 변화를 불러와 주었다고 생각합니다.

기록하는 습관

저는 과거의 기록들을 통해 부족한 점을 스스로 발견하고 고쳐 발전하는 삶을 지향합니다. 항상 일상 속 많은 부분들에 대하여 기록을 남기고 있으며, 이 후에 이 기록들을 보고 지난 행동들을 복기하며 어떤 점이 부족했는지 스스로 파악하여 앞으로 고쳐서 나아갈 길을 알려주는 중요한 자산으로 사용하고 있습니다.

개발을 할 때에도 이러한 습관에서 도움을 얻을 수 있었습니다. 사례로 교내에서 팀 프로젝트를 진행할 당시 팀원 모두가 처음 진행하는 프로젝트였기 때문에 진행하며 수많은 에러와 문제들을 마주할 수밖에 없었습니다. 문제가 발생할 때 마다 소모되는 시간적 비용을 줄이고자 저희 팀은 많은 고민을 했고 프로젝트 진행과정을 문서화하여 각자 맡은 부분의 경험들을 공유하기로 하였습니다.

저와 팀원들은 구글 독스 공유 기능을 사용하여 각자 맡은 파트를 개발하는 상세한 과정과 개발 중 만난 에러와 그 해결방안을 문서화하고 공유하였습니다. 그 결과 저 또는 팀원이 한 번 경험했던 문제를 만났을 때에 그 원인과 해결방안을 신속히 찾아내 수정할 수 있었고 같은 실수를 반복하는 일이 적어지는 결과를 불러왔습니다. 이러한 경험을 통해 저는 기록하는 습관에 대한 신뢰를 느낄 수 있었고, 문서화의 중요성을 체감하는 계기가 되었습니다.

포트폴리오:

OOO

Shoe-Auction

2021.01 ~ 2021.XX

#JAVA *#Spring* *#Spring Data JPA* *#MySQL* *#Redis* *#Junit* *#Mockito*

#Jenkins *#AWS Lambda* *#FCM* *#Restdocs*

개요

패션 시장에서 신발에 관한 열풍이 일고 있습니다. 의류 매출은 전반적으로 주춤한 가운데 운동화 매출은 꾸준히 상승하고 있습니다. 그 중에서는 고가의 해외 브랜드의 제품 또는 제한된 수량으로 발매되는 신발의 비중이 상당히 높으며, 생산량보다 훨씬 높은 수요로 인해 각종 중고거래 사이트에도 구매가에 웃돈을 얹은 신발들이 활발하게 거래되고 있습니다.

하지만 이러한 개인간 신발 거래는 대부분 중고거래 카페 또는 중고거래 어플리케이션에서 행해지고 있습니다. 신발에 대해 전문성을 가진 플랫폼이 아니다 보니 구매/판매 시 매물 정보나 거래 시세를 파악하기 어려우며, 가품 판매와 사기의 온상지로 변모할 수 있습니다. Shoe-Auction은 이러한 수요로 인해 만들어진 개인간의 신발 거래 중계를 전문으로 서비스하는 플랫폼입니다.

지속적인 확장이 가능한 설계

프로젝트를 설계하고 구현함에 있어서 실제 운영중인 서비스이며 빠른 증가로 인해 확장이 필요하다는 가정하에 진행하였기 때문에 서비스 확장이 용이한 방식을 고려하였습니다.

기존 서버에 자원을 추가하거나 고성능 장비로 교체하는 Scale-up 방식은 추가적인 네트워크 연결이 필요 없어 상대적으로 손쉬운 확장이 가능하고 관리 편의성 측면에서 이점이 있으나, 지속적인 확장에는 한계가 존재하며 서버에 장애가 발생할 시 장애 영향도를 크게 받을 수 밖에 없다는 단점이 있습니다.

그에 비해 기존 서버와 동일한 사양의 서버를 추가해 다중 서버에 트래픽을 분산하는 Scale-out 방식은 상대적으로 확장의 한계가 없기 때문에 본 프로젝트의 지향점에 적합하다는 판단 하에 Scale-out 확장 방식을 채택하였습니다.

로그인 정보 Session의 불일치 이슈 해결

Scale-out 확장으로 인하여 다중 서버 환경이 구축됨으로써 기존의 로그인 상태 유지를 위해 Session에 담아서 처리하고 있던 로그인 정보의 정합성 이슈를 해결하기 위해 적절한 방법을 찾아 비교 후 적용하였습니다.

Session을 처음에 생성한 서버에만 요청을 고정적으로 전송하는 Sticky Session 방식은 Session 불일치 이슈를 명확히 해결할 수 있었지만, 서버들에게 트래픽을 균일하게 배분할 수가 없기 때문에 또 다른 이슈가 발생할 수 있다는 단점이 있었습니다.

Session 생성 시 다른 서버들로 Session을 복제하도록 하는 Tomcat 설정을 통한 Session Clustering 방법 또한 Session 불일치 이슈를 해결할 수 있고, Sticky Session 방식에서 발생하는 특정 서버로의 트래픽 집중 이슈에서 또한 자유로울 수 있지만, 모든 서버의 동일한 Session을 저장해야 하기 때문에 메모리가 비효율적으로 사용될 수 있는 등 서버 확장에 따른 부하가 크기 때문에 지향하는 서버 구조와 맞지 않는다고 판단했습니다.

결국 다중 서버가 하나의 외부 Session 저장소를 공유하는 방법을 채택하여 Session 불일치 이슈를 최대한 효율적으로 해결할 수 있었습니다. Session 저장소로는 Session을 비교적 빠른 속도로 저장하고 조회할 수 있는 In-memory DB 중에서 고려하였으며, 스프링에서 제공하는 spring-data-redis를 사용해서 구현 복잡도를 최소화하여 구현할 수 있는 Redis를 채택하였습니다.

Aws Lambda를 이용한 이미지 리사이즈

사용자가 거래할 상품과 상품의 브랜드 이미지의 저장소는 다중 서버간 통신을 줄여줄 수 있으며, 높은 확장성과 복구 기능을 가진 AWS S3를 선택하였습니다. 외부 저장소를 선택한 만큼 추가되는 통신 과정은 어쩔 수 없는 부분이지만 대용량의 원본 이미지 파일을 클라이언트의 요청 시마다 전송하는 것은 비효율적이라는 판단 하에 사진을 다양한 사이즈로 리사이징하여 저장하고 요청마다 적합한 사이즈의 이미지를 전송하는 방법을 채택하였습니다.

다만 이러한 방법은 이미지 업로드 시 이미지를 리사이징하는 작업이 추가되며, 리사이징한 여러 이미지를 저장소에 업로드 해야 하기 때문에 애플리케이션 서버의 부담이 증가할 수 있다는 이슈가 있었습니다. 따라서 AWS Lambda를 사용하여 서버리스한 방식으로 리사이징과 업로드를 처리하는 방법을 적용하였습니다. 이로써 애플리케이션 서버의 추가 작업 부담을 덜 수 있었고, 기존의 간단한 업로드 방식을 그대로 사용할 수 있어 주 로직에 더욱 집중할 수 있게 되었습니다.

온디맨드 이미지 리사이징 방식을 사용한다고 해도 같은 효과를 누릴 수 있고 저장되는 총 이미지의 용량이 감소한다는 장점이 있지만, 이미지 파일 리사이징이 필요한 사이즈 수가 많지 않고 이미지 요청 시 리사이징을 시작하기 때문에 로딩 시간이 증가한다는 단점을 들어 배제하였습니다

적절한 캐싱 전략 사용

상품과 브랜드 조회 시 반복되는 DB 접근으로 인한 네트워크 병목을 줄이고자 캐시를 적용하여 읽기 성능을 개선하였습니다. 이 과정에서 각 캐싱 전략의 트레이드 오프를 고려하여 비교해보는 과정을 거쳤습니다.

서버마다 캐시를 따로 저장하는 로컬 캐싱 전략의 경우 서버 내에서 작동하기 때문에 속도가 빠르다는 장점이 있었지만, 서버의 개수만큼 저장되는 캐시 데이터의 양도 비례하여 증가하므로 서버의 지속적 확장을 예상하는 서버 구조와는 맞지 않는다고 판단했습니다.

그에 비해 글로벌 캐싱 전략은 여러 서버에서 외부 캐시 서버를 참조하는 방식으로 동작하므로 로컬 캐시보다는 느리지만 서버간 데이터 공유가 쉬우며 서버 확장시에도 저장되는 캐시 데이터의 양은 동일하므로 본 프로젝트에 적합하다고 생각되어 글로벌 캐싱 전략을 선택하였습니다.

캐시 스토리지 또한 세션과 마찬가지로 빠른 속도를 가진 In-Memory DB이며, 이미 본 프로젝트에 사용한 경험이 있는 Redis를 선택하였습니다.

다양한 방법으로 협업 진행

본 프로젝트는 저를 포함한 총 2명의 협업으로 진행되었습니다. 프로젝트를 시작하기 전에 개발 방식의 통일성을 지키기 위해 충분한 논의 후 컨벤션을 적용하여 코딩 스타일을 통일하였으며, 설계의 내용을 서로 동일하게 이해하고 구현하기 위하여 프로토타이핑 툴인 '카카오 오븐'을 통하여 설계한 내용들을 시각화 하였습니다.

기본적인 의사소통과 프로젝트와 관련된 회의는 Slack과 google meet를 통해 진행하였으며,개발 과정에서는 Github의 Pull Request 기능을 통해서 서로가 맡은 부분들에 대한 코드 리뷰를 주고받았습니다. 최대한 꼼꼼하게 서로의 구현 방식이 적절한지 판단하며 리뷰를 나눴고, 기존의 구현 방식과 리뷰로 제시된 내용과 비교하며 더 나은 방향으로 리팩토링을 진행하였습니다.

졸업 관리 시스템

2018.09 ~ 2018.11

#JAVA #JSP #Spring #MyBatis #MySQL

개요

학교의 졸업조건은 학번, 학과, 심화 별로 다양하게 구성되어 있기 때문에 학생마다 졸업의 조건이 상이합니다. 하지만 졸업조건을 알아보기 위해서는 교내 사이트를 뒤지며, 여러 졸업요건 문서들을 확인하고, 직접 계산기를 두들기며 학점을 계산해야 하는 불편함이 있습니다. 학생들의 이러한 불편함을 해소하고자 교내 졸업 관리 시스템이라는 웹 어플리케이션을 제작하게 되었습니다.

졸업관리 시스템 사이트는 회원가입 후 학과를 선택하고 교내 종합정보시스템에서 다운받은 학점 조회 엑셀파일을 업로드하여서 자동으로 시각화 된 졸업요건을 조회할 수 있으며, 수강 중이거나 앞으로 수강할 과목들을 추가하여 시뮬레이션으로 확인할 수도 있습니다. 학번/학과 별 졸업 조건을 기준으로 남은 학점을 그래프로 제시해주며 미 수강 필수과목, 봉사활동 이수 여부 또한 조회할 수 있습니다. 기본적으로 주 전공에 관한 요건 충족 여부를 제시해 주며 추가로 복수전공과 부전공 또한 선택하여 추가할 수 있습니다.

도메인 분석

프로젝트의 주제가 졸업 관리 시스템인 만큼 팀원들 모두가 교내의 조건 별 졸업요건에 대해서 완벽하게 숙지하고 있어야만 했습니다. 그래서 모델링 이전에 도메인 분석부터 완료하기로 하였고 팀원들과 회의한 결과 각자 파트를 나누어서 교내 공식문서를 통해 졸업요건에 대해 정확하게 숙지하고 공유하기로 하였습니다. 맡은 부분을 다른 팀원들에게 설명하는 방식을 통해서 팀원 모두가 모든 졸업요건에 대해 완벽히 숙지할 수 있었으며 수월하게 도메인 분석을 마칠 수 있었습니다. 프로젝트를 시작하기 앞서 완벽하게 도메인을 분석함으로써 프로젝트에 대한 팀원들의 이해도를 높여 프로젝트 진행 간 발생할 수 있는 팀원 간의 이해 차이를 줄일 수 있었습니다.

학점 계산 및 시각화

엑셀을 통해 업로드 되어 테이블에 저장된 학점을 계산해주고, 계산된 값들을 시각화 해주는 기능을 담당하게 되었습니다. MySQL 쿼리를 이용해서 전공/교양 별 이수 학점과 평점을 뽑아내고 계정에 설정된 졸업 과정의 조건과 비교하여 이수율을 계산하였습니다. 백분율로 계산한 이수 비율과 미 이수 비율을 쉽게 표현해주기 위해 차트 라이브러리에서 CSS를 이용한 원형그래프를 사용해서 시각화 하였으며, 전체 필요 학점 대비 부족한 학점의 값 들과 함께 배치해주었습니다. 또한 졸업조건의 테이블의 필수 과목들과 나의 수강과목을 비교하여 아직 수강하지 않은 졸업 필수 강의들을 도출해내서 전공, 교양 별로 출력해주었습니다.