초격차 패키지 Online.

IT 대기업 이직을 위한 팁

Clip 1 | 지원 동기의 함정

Clip 2 | 읽기 좋은 자소서 작성법

IT 대기업 이직을 위한 팁

1 지원 동기의 함정

지원 동기의 함정

서비스 회사로의 이직을 희망하시는 분들에게
지원 동기를 물어보면 90% 이상 나오는 키워드가 다음과 같습니다.
ex) “대규모 트래픽 경험”, “개발 문화”

다시 말해 현재 회사의 기술 스택과 환경이 맘에 들지 않고
그런 부분들을 해소하고자 서비스 회사로 이직을 희망하는 것은
아마 면접관분들도 알고 있을 것이고 공감할 것입니다.

그런데 여기서 중요한 포인트는
단순히 불평불만만 하는 지원자를 **굳이 뽑을 이유**가 없다는 점입니다.

회사는 학원이 아니다.

회사의 본질은 지원자의 꿈과 희망을 이뤄주는 곳이 아닙니다.

철처하게 비즈니스 관계로 계약을 맺고
본인의 능력을 회사의 발전을 위해 사용하고 그에 맞는 보상을 받는 곳입니다.

다만 그 과정에서 운이 좋게 본인이 희망하는 부분들이 충족되는 곳이지
단순히 “당신의 꿈과 희망을 이뤄드립니다.”가 아니란 뜻입니다.

그러므로 지원 동기로
“대규모 트래픽 경험을 하고 싶습니다.” “좋은 개발 문화를 접해보고 싶습니다.”
라고만 말하는 지원자는 굳이 뽑을 이유가 없습니다.

그러면 어떤 지원자를 뽑고 싶어 할까?

당연하게도 **뛰어난 능력**을 갖춘 사람입니다.

여기까지 보고 나면 이런 생각이 들 수 있는데요

‘ 나도 뛰어난 개발자가 되고 싶어. ’

‘ 그런데 현재 회사에서는 그게 불가능하니까 지원을 했으니 나를 뽑아줘 ’

‘ 그러면 나도 뛰어난 능력을 갖춘 개발자가 될 수 있다는 걸 보여줄게 ! ’

역질문

역으로 질문을 해볼까요?

회사는 지원자의 어떤 점을 보고 뽑아야 할까?
단순히 의지만 갖고 있다고 뽑아야 할까?

그런 의지는 솔직히 모든 지원자가 다 갖고 있습니다.
그렇다면 그 모든 지원자를 다 채용해야 할까요?
당연히 그건 불가능하겠죠.

관점의 전환

관점을 바꿔서 생각해봅시다.
“ 회사를 통해 본인의 갈증을 해소하고 싶어요. ” 가 아니라
“ 나의 능력을 회사의 발전에 이바지할 수 있어요. ”

그러므로 합격을 하고 싶다면
본인의 노력이 뒷받침되어야 하고
내가 노력한 부분들을 잘 어필해야합니다.

그래야지 남들과는 차별화된 지원자가 될 수 있으며
면접관이 이런 생각을 할 수 있게 만들 수 있습니다.
“ 지원자가 능력과 태도는 갖추었으나 환경이 못 받쳐주는구나. ”
“ 환경만 제공되면 뛰어난 개발자가 될 수 있겠다. ”

N년차의 내가 이직을 준비한다면?

1 ~ 1.5년 차라면 – 학습 태도

지원하고자하는 회사의 기술 스택을 공부했다는
학습 태도를 많이 어필했으면 좋겠습니다.
“ 나는 계속해서 뛰어난 개발자가 되기 위해 많은 노력을 하고 있다. ”
ex) 인터넷 강의를 듣고 정리한 기록을 볼 수 있는 Public 한 링크
(“개인 노트에 정리했어요.” 는 믿을 수 없겠죠?)

아직은 사회 초년생이므로 학습 태도가 되어있다면
채용하는 입장에서도 큰 부담 없이 채용할 수 있습니다.

N년차의 내가 이직을 준비한다면?

1 ~ 1.5년 차라면 - 사용한 기술 스택에 대한 이해도

회사에서 개발하면서
사용한 기술 스택을 잘 설명할 수 있어야 합니다.
비록 원하는 기술 스택이 아니더라도
사용한 기술에 관해 설명을 잘 못 한다면
“ 잘 모르고 사용했구나 ” 라는 이미지가 생길 수밖에 없습니다.

이 이미지가 왜 중요할까요?
과거에도 기술 스택에 대한 이해 없이 개발했으니
우리 회사 기술 스택도 똑같이 이해 없이 사용하겠네? 라고 해석할 수 있기 때문이죠.
즉 **과거의 태도를 보고 미래를 예측**할 수밖에 없습니다.

N년차의 내가 이직을 준비한다면?

1.5 ~ 3년 차라면 – 사이드 프로젝트

1 ~ 1.5년 차에서 언급한 “학습 태도, 사용한 기술 스택에 대한 이해도” 뿐만 아니라
실제 프로젝트 경험이 필요합니다.

1.5 ~ 3년이라면 ‘ 시간이 없어서 못했어요.’ 라고 말하는 건 **핑계**입니다.
어떻게든 시간을 내서 해야만 합니다.
‘ 그냥 회사 일만 하다 보니 시간이 이렇게 흘렀네요. ’
‘ 그런데 지금이라도 변화를 하고 싶어요 ! ’ 라고 말만 하는 지원자는 경쟁력이 없습니다.

그러므로 지원하고자 하는 회사의 기술 스택을 사용한 사이드 프로젝트 경험이 필요하고
그래야 합격 가능성이 조금이라도 더 높아질 수 있습니다.

N년차의 내가 이직을 준비한다면?

4 ~ N년 차라면 – 설계 능력

“학습 태도, 사용한 기술 스택에 대한 이해도, 사이드 프로젝트”
3 요소는 기본적으로 갖춰야 하고
추가적으로 설계 경험이 중요합니다.

그런데 설계 경험은 동아리 활동 혹은 사이드 프로젝트 경험이 없다면
절대로 이야기할 수 없는 부분이라 생각합니다.

N년차의 내가 이직을 준비한다면?

4 ~ N년 차라면 – 설계 능력

아무래도 연차가 있는데
우리가 사용하는 기술 스택과 거리가 먼 커리어라면
채용에 있어 더 조심스러울 수밖에 없고
연차에 맞는 실력을 요구하는 건 어쩔 수 없습니다.

그러니 말로만 자신을 보여주는 게 아니라
실제 프로젝트 경험을 기반으로 본인을 보여주는 게 맞습니다.

이곳만은 꼭 !

다시 한 번 말하자면
현재 회사의 기술 스택과 개발 문화가
맘에 안 들어서 이직을 하려는 건 면접관도 다 압니다.
여기서 중요한 포인트는 **어떠한 노력을 하였는가**입니다.

예를 들어 볼게요.

```
현재 거주하는 지역에서 전쟁이 일어났다.
그래서 사랑하는 가족의 안전을 위해 특별 경호원을 고용하기로 했고
경호를 하고 싶다는 지원자와 일정을 잡고 면접을 진행했다.
자세도 좋고 인성도 좋아 보였다.
그런데 한 가지 걸리는 게 있다.
실제로 누군가를 경호해본 경험이 없고 경호에 대한 기본적인 개념도 모른다.
하지만 지원자는 경호를 너무 해보고 싶다고 강력하게 어필한다.
```

이곳만은 꼭 !

만약 저런 상황이라면 그 경호원을 뽑을 수 있을까요?

저라면 뽑지 않을 것 같습니다.

왜냐하면 실전 경험도 없으며

그것과 관련된 기본적인 학습의 노력도 없기 때문에 신뢰가 많이 떨어지기 때문이죠.

이곳만은 꼭 !

위 이야기를 별다른 노력 없이
이직을 희망하는 개발자로 변형을 해보자.

```
현재 대규모 트래픽이 예상되는 신규 서비스를 개발하려고 한다.
그래서 그 서비스를 개발할 수 있는 개발자를 고용하기로 했고
지원자와 일정을 잡고 면접을 진행했다.
자세도 좋고 인성도 좋아 보였다.
그런데 한 가지 걸리는 게 있다.
실제로 대규모 트래픽 개발 경험이 없고
대규모 트래픽을 다루기 위한 최소한의 개념도 모른다.
하지만 지원자는 개발을 너무 해보고 싶다고 강력하게 어필한다.
```

이곳만은 꼭 !

본인이 면접관이라고 한다면

지원자가 대규모 트래픽을 핸들링하기 위한

최소한의 개념(ex. HA구성)도 학습하지도 않고

“ 대규모 트래픽을 경험해보고 싶어서 지원했습니다! ” 라고 말한다면

그 지원자를 합격시킬 수 있을까요?

그 지원자가 우리 팀에 온다면 환영을 해줄 수 있을까요?

이곳만은 꼭 !

본인이 면접에서 대규모 트래픽 경험이라는 키워드를 사용했다면

그 개념을 위해 어떠한 노력을 했는가를 반드시 체크해봤으면 좋겠습니다.

“ 지금 회사에서는 트래픽이 없어서 몰라요 □□ ” 라고 말하는 건 핑계입니다.

이미 시중 도서 + 인터넷에는 다양한 자료들이 있기 때문이죠.

이곳만은 꼭 !

지원한 회사의 팀원분들은

신규 입사자분의 실력이 뛰어났으면 좋겠단 생각을 하고

그런 개발자와 함께하길 희망하지

아무것도 모르는 사람 + 노력도 하지 않은 사람을

내 동료로 맞이하고 싶어하는 사람은 아마도 없을 것이기 때문이죠.

이곳만은 꼭 !

그러니 **감성팔이 지원자**가 아닌

실력을 겸비한 지원자가 되도록 꾸준히 노력하는 개발자가 될 수 있도록

꾸준히 노력을 합시다 !

IT 대기업 이직을 위한 팁

2 읽기 좋은 자소서 작성법

읽기 좋은 자소서 작성법

IT 대기업 기준으로 공채를 진행한다면

접수되는 서류의 수는 최소 100~1000개는 된다.

수많은 자소서 속에서 내 자소서가 돋보이게 하기 위해선 어떻게 해야할까?

읽기 좋은 자소서 작성법

현실적으로 채용 담당자가

모든 서류를 하나하나 꼼꼼히 확인할 수 있을까?

가능하겠지만 채용 담당자는 채용“만”하는 게 아니라

본인의 업무와 채용까지 담당하게 된다.

그러므로 서류 하나하나 꼼꼼하게 확인은 사실 쉽지 않다.

읽기 좋은 자소서 작성법

그렇다면 어떻게 판단을 할까?

빠르게 훑으면서 우리가 원하는 기술 스택을 보유하였는지?

혹은 괜찮은 인재인지 빠르게 판단을 해야한다.

그렇게 하기 위해서 지원자는 **읽기 좋은 자소서**를 작성해야한다.

읽기 좋은 자소서 작성법

어떤 부분이 읽기 좋은 자소서라고 표현할 수 있을까?

이 부분에 대해선 주관적이므로

뭐가 절대적으로 맞다라는 건 없다.

하지만 그럼에도 최소한의 몇가지 조건만 충족시킨다면

평소에 글을 잘 쓰지 못하더라도

잘 작성된 읽기 좋은 자소서는 작성할 수 있다.

읽기 좋은 자소서 작성법

공통된 포맷

문항마다 공통된 포맷으로 작성을 하면

글이 좀 더 구조화되어 있고

보는 입장에서도 한 눈에 파악하기 쉽다.

읽기 좋은 자소서 작성법

공통된 포맷

추천하는 포맷은 다음과 같다.

1. 기간
2. 주최
3. 기술 스택
4. 프로젝트 설명
5. 어필하고 싶은 부분

공통된 포맷 - 기간

프로젝트에 사용한 기간을 명시하면 된다.

간단 명료 하지만 기간이 길다면

왜 그렇게 길었는지?

그 기간동안 어떤 것들을 개발하였는지?

어필할 부분이 어떤게 있는지를 **반드시** 정리해야한다.

공통된 포맷 - 주최

주최는 2가지로 볼 수 있다.

1. 회사
2. 사이드 프로젝트

만약 회사에서 진행한거라면 회사라고 작성하면 된다.

공통된 포맷 - 주최

1. 회사
2. 사이드 프로젝트

그런데 이런 케이스가 있을 수 있다.
“현재 회사 기술 스택 != 지원하고자하는 회사 기술 스택”

그래서 모든 프로젝트가 사실 사이드 프로젝트 위주이다.
게다가 경력 이직을 희망한다.

이럴 경우엔 “주최” 항목을 빼는 것도 하나의 전략이다.

공통된 포맷 - 주최

1. 회사
2. 사이드 프로젝트

아무래도 경력직을 채용하는 회사 입장에서는

사이드 프로젝트 경험 보다는

실제 회사에서 고객을 대상으로 운영했던 프로젝트를 희망할 수 있으므로

본인의 상황에 따라서 적절하게 빼는 것도 좋은 선택이라 할 수 있다.

공통된 포맷 - 주최

1. 회사
2. 사이드 프로젝트

혹은 전략을 이렇게도 세울 수 있다.

“비록 회사에서 사용하는 기술 스택은 아니지만

따로 시간을 내어서 꾸준히 개발하고 기술 스택에 대한 이해도가 있다.”

라는 식으로 태도를 어필하는 것도 좋은 전략이다.

공통된 포맷 – 기술 스택

과유불급

기술 스택을 쓰는란에 욕심이 과하여

정말로 불필요한 기술까지 나열하는 케이스도 생각보다 많다.

ex) Java, Spring MVC, Spring Boot, MyBatis, JPA, React, JS, CSS, HTML, AWS, Hadoop, Kafka 등등

이렇게 작성된 자소서를 보면 무슨 생각이 들까?

공통된 포맷 – 기술 스택

과유불급

정말 현실적으로 저 모든 기술을 다 습득하고 사용을 했을까?
여기서 사용을 했다는건 단순히 코더 느낌으로
되어있는 기능을 사용했다가 아니라
본질을 파헤쳐보고 원리를 이해한 수준을 뜻한다.

기간과 연차에 비해 1개의 프로젝트에
너무나도 많은 기술 스택을 적는건 과유불급이다.
너무 과해서 오히려 역효과가 난다.

공통된 포맷 – 기술 스택

과유불급

또한 백엔드를 뽑는 포지션에
프론트 엔드 기술 스택도 같은 레벨로 나열을 하는 케이스가 굉장히 많은데

프론트 엔드 기술 스택이 있다고 가산점이 있거나
합격에 큰 영향을 끼치진 않으니
(개인적으로) 해당 포지션에 맞는 기술 스택만 나열하는 걸 추천한다.

그리고 나열도 내가 면접에서
잘 준비해서 잘 대답할 수 있는 범위내에서만 적는걸 추천한다.

공통된 포맷 – 기술 스택

과유불급

경력 이직같은 경우엔
아무래도 현 회사를 다니면서 준비를 하는 것이다보니
시간적으로 더더욱 부족함을 많이 느낄 수 밖에 없다.

그런 상황에서 내가 준비할 수도 없는
내가 정말 잘 알지도 못하는 기술 스택을 나열하는 건
오히려 독이 된다.

그러니 바쁜 와중에서도 최대한 시간을 내어
밀도있게 답변을 준비할 수 있는 범위를 잘 계산하여 작성을 하도록 하자.

공통된 포맷 – 기술 스택

과유불급

정리하자면 **핵심적인 키워드만** 적어도 된다.

예를 들어 Java / Spring Boot / JPA

이렇게 가장 핵심이 되는 3가지 키워드만 작성하여도

충분히 합격할 수 있다.

공통된 포맷 – 프로젝트 설명

프로젝트에 대한 간략한 설명이라도 작성을 해줘야

읽는 입장에서 어떤 서비스에서 어떤 기술 스택을 사용했는지

면접관 본인의 경험에 비추어 유추를 해 볼 수 있다.

또한 프로젝트에 대한 최소한의 공감대가 형성 되어야

내가 작성한 자소서에 대한 이해도가 높아진다.

그러니 프로젝트에 대한 설명을 간략하게라도 작성하도록 하자.

공통된 포맷 – 어필하고 싶은 부분

자소서를 작성하는 근본적인 이유를 알아야한다.

자소서를 통해 질문을 하고
그 사람의 경험과 능력을 파악할 수 있다.

그러므로 내가 정말 어필하고 싶은 부분은
따로 **강조**를 하여 작성할 필요가 있다.

내가 원하는 질문을 해주길 기다리는 건 희박한 확률에 도박을 하는 것과 같다.
내가 원하는 질문을 받을 수 있도록
자소서에 먼저 제시를 해보도록 하자.

IT 대기업 이직을 위한 팁

3 포트폴리오 작성

포트폴리오

포트폴리오를 만드는 목적

1. 개인의 만족
2. 취업

포트폴리오

포트폴리오를 만드는 목적

1. 개인의 만족
2. **취업**

포트폴리오

포트폴리오 vs 자소서

그러면 포트폴리오와 자소서는 뭐가 다를까?

포트폴리오

포트폴리오 vs 자소서

그러면 포트폴리오와 자소서는 뭐가 다를까?

분명 다르므로
회사에서도 자소서와 포트폴리오를 각각 요구하지 않을까?

포트폴리오

포트폴리오 vs 자소서

이렇게 정리를 해볼 수 있다.

시각 자료 vs 텍스트

포트폴리오

포트폴리오 vs 자소서

이렇게 정리를 해볼 수 있다.

시각 자료 vs 텍스트

시각 자료는 포트폴리오와 자소서중 어떤것에 해당할까?

포트폴리오

포트폴리오 vs 자소서

즉 포트폴리오는 자소서와 **콜라보레이션**을 이뤄야한다.

포트폴리오

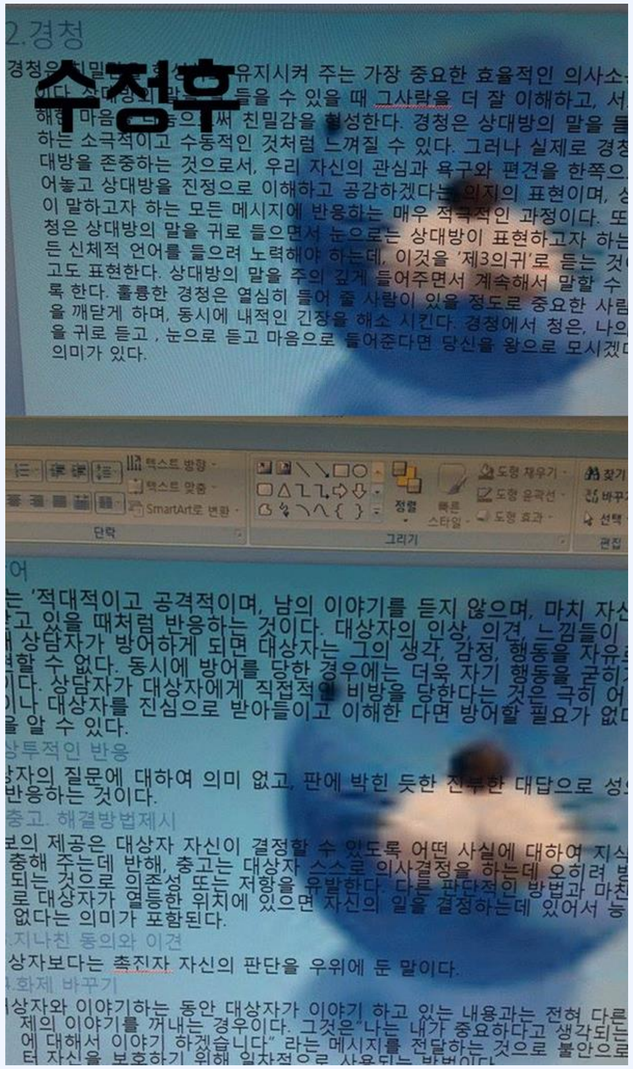
포트폴리오 작성법 – 피해야하는 스타일

1. 구구절절
2. Drop the CODE

포트폴리오

포트폴리오 작성법 – 피해야하는 스타일

1. **구구절절**
2. Drop the CODE



https://www.huffingtonpost.kr/2016/06/02/story_n_10251950.html

포트폴리오

포트폴리오 작성법 – 피해야하는 스타일

1. 구구절절
2. **Drop the CODE**

```
let ad_id = req.query.ad_id;
let date = req.query.date;
let selectQuery =
`
SELECT DATE_FORMAT(time,"%Y-%m-%d %H:%i:%s") as
time,COUNT(*) as count
FROM ad_clicked_log
WHERE ad_id =` + ad_id + ` AND time LIKE '` + date + `%'
GROUP BY UNIX_TIMESTAMP(time) DIV 3600
ORDER BY time
`
let result = [];
try {
let _result = await db.query(selectQuery);
let count = 0;
let tmp = _result[count].time.substr(11,2);
for(let i = 0 ; i <=23; i++) {
if(i <= tmp && tmp<i+1) {
result.push(_result[count].count);
count++;
if(count < _result.length) {
…
```

포트폴리오

포트폴리오 작성법 – 피해야하는 스타일

Drop the CODE

코드 = 비즈니스

비즈니스를 모르는데 코드를 본다(?)

포트폴리오

포트폴리오 작성법 – 피해야하는 스타일

Drop the CODE

코드 = 비즈니스

비즈니스를 모르는데 코드를 본다(?)

주객이 전도된 상황이다.

포트폴리오

포트폴리오 작성법 – 피해야하는 스타일

Drop the CODE

코드는 비즈니스를 컴퓨터 언어로 표현을 하는 수단이다.

그러므로 읽는 사람이 비즈니스를 파악하고 있어야

코드를 보더라도 이해가 쉽고 서로 Sync가 맞게 된다.

포트폴리오

포트폴리오 작성법 – 피해야하는 스타일

Drop the CODE

그런데 다른 회사사람에게 비즈니스를 설명할 기회도 없고

설명할 수도 없지 않나?

포트폴리오

포트폴리오 작성법 – 피해야하는 스타일

Drop the CODE

그런데 다른 회사사람에게 비즈니스를 설명할 기회도 없고

설명할 수도 없지 않나?

그럼에도 코드를 첨부하고 싶다면?

포트폴리오

포트폴리오 작성법 – 피해야하는 스타일

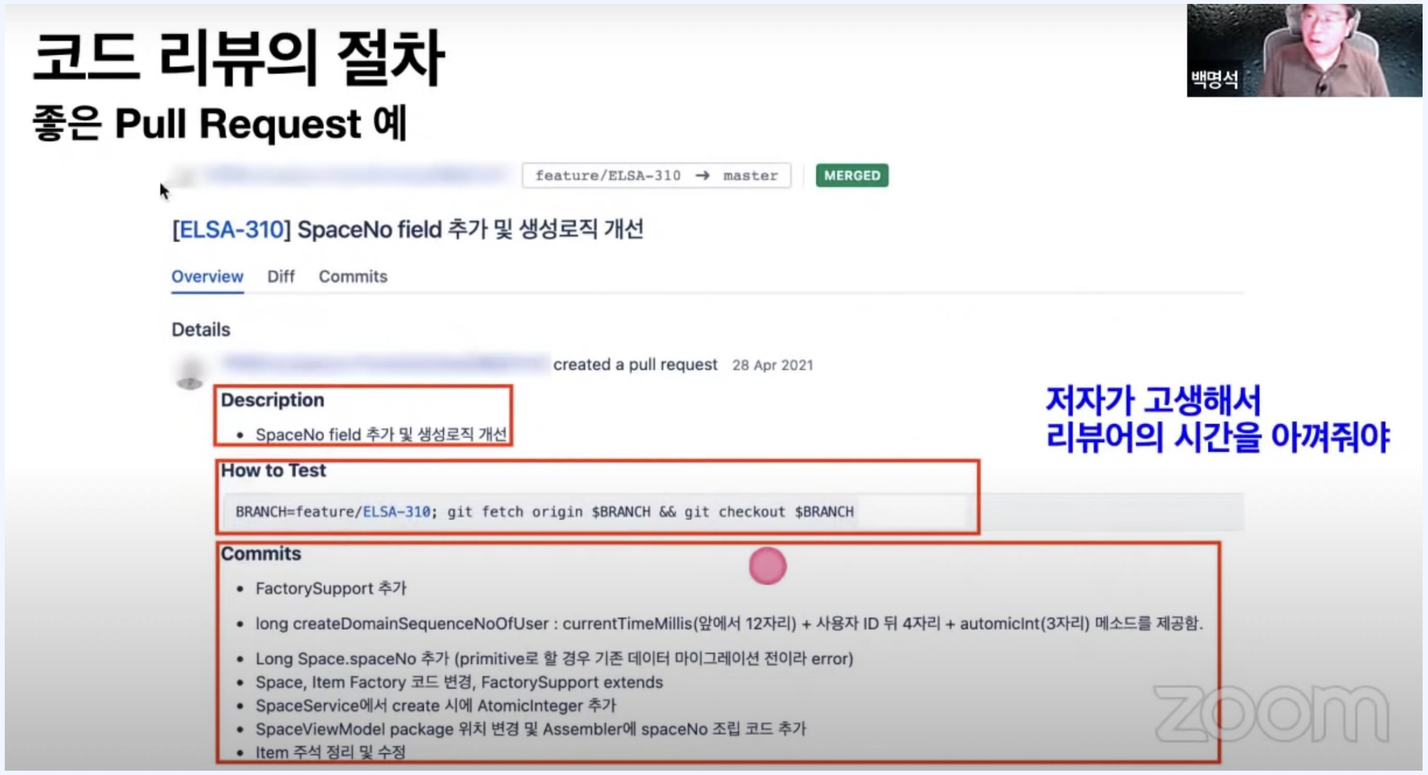
Drop the CODE

저자가 **고생**을 해야한다.

포트폴리오

포트폴리오 작성법 – 피해야하는 스타일

Drop the CODE



https://www.youtube.com/watch?v=ssDMIcPBqUE&t=6635s

포트폴리오 작성법 – 작성 스타일

1. **정적**
2. **동적**

포트폴리오

포트폴리오 작성법 – 작성 스타일 – 정적

반드시 **PDF**로 제출

PPT, Word, Hwp 등등은 시스템 환경에 따라서

디자인 + 폰트가 깨질 수 있다.

기본적으로 PDF로 파일을 제출하는게 기본 중의 기본 (+ 센스)

포트폴리오

포트폴리오 작성법 – 작성 스타일 – 동적

Notion, Github 등등

동적으로 제출 시 장점은 제출 후 수정이 가능하다.
그러므로 제출했다고 끝! 이라는 생각은 접어두고
계속해서 관리를 하도록 하자.

포트폴리오

포트폴리오 작성법 – 주의해야할 점

기술 스택 나열 형식

과연 좋은 선택일까?

포트폴리오

백엔드 개발자를 위한 포트폴리오 작성 가이드

추천하는 작성 가이드 포맷

1. Project Summary
2. Sequence Diagram
3. Code Snapshot
4. System Architecture

포트폴리오

백엔드 개발자를 위한 포트폴리오 작성 가이드

Project Summary

Frame Work
- Spring Boot 2.3.4
- Java 11

Persistence
- JPA
- QueryDSL

Client
- Feign

Dependency
- Gradle

DB
- MySQL

Project Summary
- 20.10 .01 ~ 20 .12 .23
-
-
-

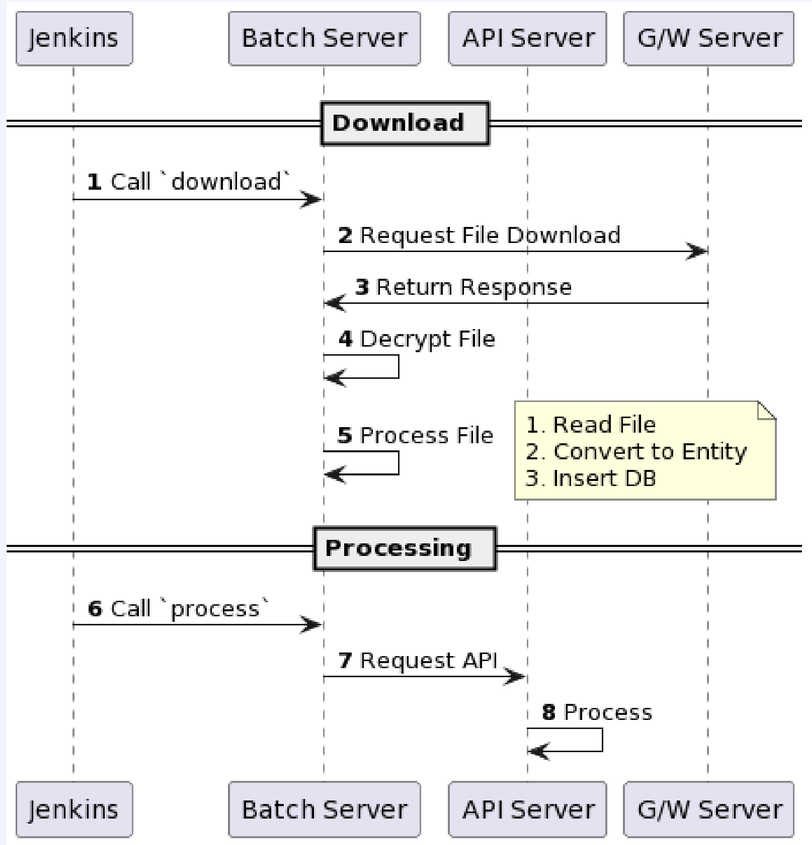
What to do
- Batch Job 개발
- Thread Count 제어

포트폴리오

백엔드 개발자를 위한 포트폴리오 작성 가이드

Sequence Diagram

- Sequence Number + 간결하게 작성



포트폴리오

백엔드 개발자를 위한 포트폴리오 작성 가이드

Code Snapshot

- 주석과 더불어
- 비즈니스를 모르더라도
- 이해할 수 있을 정도로
- 읽기 편한 코드를 첨부

```
/**
 * Intercepts calls on an interface on its way to the target. These
 * are nested "on top" of the target.
 *
 * <p>The user should implement the {@link #invoke(MethodInvocation)}
 * method to modify the original behavior. E.g. the following class
 * implements a tracing interceptor (traces all the calls on the
 * intercepted method(s)):
 *
 * <pre class=code>
 * class TracingInterceptor implements MethodInterceptor {
 *   Object invoke(MethodInvocation i) throws Throwable {
 *     System.out.println("method "+i.getMethod()+" is called on "+
 *                        i.getThis()+" with args "+i.getArguments());
 *     Object ret=i.proceed();
 *     System.out.println("method "+i.getMethod()+" returns "+ret);
 *     return ret;
 *   }
 * }
 * </pre>
 *
 * @author Rod Johnson
 */
@FunctionalInterface
public interface MethodInterceptor extends Interceptor {

    /**
     * Implement this method to perform extra treatments before and
     * after the invocation. Polite implementations would certainly
     * like to invoke {@link Joinpoint#proceed()}.
     * @param invocation the method invocation joinpoint
     * @return the result of the call to {@link Joinpoint#proceed()};
     * might be intercepted by the interceptor
     * @throws Throwable if the interceptors or the target object
     * throws an exception
     */
    @Nullable
    Object invoke(@Nonnull MethodInvocation invocation) throws Throwable;

}
```

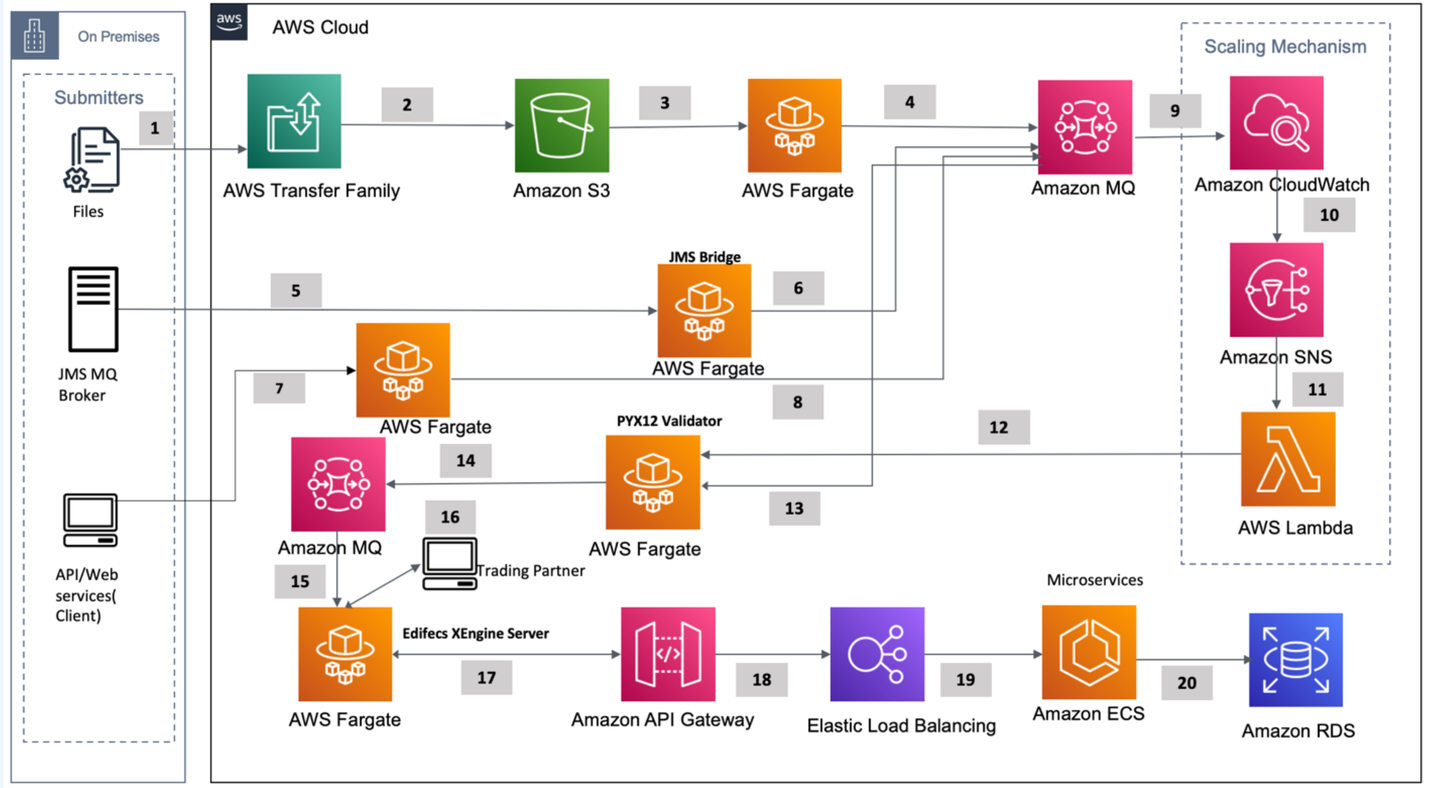
https://github.com/spring-projects/spring-framework/blob/main/spring-aop/src/main/java/org/aopalliance/intercept/MethodInterceptor.java

포트폴리오

백엔드 개발자를 위한 포트폴리오 작성 가이드

System Architecture

- 백엔드 개발자 포트폴리오의 꽃



https://aws.amazon.com/blogs/architecture/building-a-serverless-cloud-native-edi-solution-with-aws/