**여름방학 5주차 회의록**

안세훈

**1. 프레임 워크 조사**

- Flutter => 앱 / 웹 개발 활용성 조사

- Spring => DB / 서버 활용성 조사

**2. 도커 활용방안 조사**

- 문제점: 서로 환경이 다르기 때문에 모델링 협업에서 호환성 등의 이슈가 생길 것으로 예상

- 도커 데스크탑을 이용해 공동의 작업 환경 조성 가능

- 이미지 파일을 이용한 환경 공유 스터디

**8월 1주차 활동 결과 요약**

**도커 환경 설정**

- 개발 환경 세팅 오류가 많아 임시 보류

- 모델링 작업 전까지 컨테이너 생성 후 배포 예정

**프레임 워크 조사**

- **하이브리드 앱**: htm,l css javascript, 웹 뷰를 감싸고 있는 웹사이트

Pros: 네이티브 언어 몰라도 개발 가능

Cons: UI를 수동으로 만들어야함, 스마트폰의 기능 사용 제한(치명적 단점)

- **크로스플랫폼 앱 / 웹**: react -> react(java script) / flutter -> dart

Pros: 하나의 코드로 2개 플랫폼에서 사용가능

Cons: 퍼포먼스 낮아질 수 있음

- **네이티브 앱**: android -> java / ios -> swift

Pros: 다양한 API 사용 가능, 퍼포먼스 높음

Cons: 두개의 언어 사용, 두배의 개발시간

**Flutter**



구글에서 출시한 모바일/웹/데스크톱 크로스플랫폼 프레임워크

**언어** : Dart (멀티 플랫폼 상에서 동작되도록 하는 앱을 위해 디자인 된 언어)

**공식 지원 IDE** : 안드로이드 스튜디오, VS CODE 등

(vscode 사용시에도 안드로이드 스튜디오는 인증용, VMD 가져오기용으로 필수)

**플러터 설치방법**

<https://blog.naver.com/chandong83/221839261430>

**특징**

코드 하나로 웹, 안드로이드, ios 동시 개발 가능

파이어베이스 등을 통해 서버 연동 가능

AWS에서 플러터 지원

**장점** : 네이티브에 비해 진입장벽이 낮으며 성능도 우수함(네이티브와 비슷한 속도)

UI 디자인에 용이함

**단점** : 네이티브에 비해 제공되는 기능이 적은 편(패키지를 통해 계속 업데이트 중)

커뮤니티가 크게 활성화 되지 않았으며 리액트네이티브에 비해 사용 빈도가 적음

**DartPad**  
<https://dartpad.dev/?null_safety=true>

간단한 코드는 DartPad 웹에서 바로 실행 가능

어도비 XD를 통해서도 디자인 가능

**Spring**

<framework와 library의 차이점 >

내가 코드를 컨트롤 하는 것 vs 누군가의 규칙을 따라 코딩하는 것.

library ex : JQuery – 코딩을 하다가 내가 제이쿼리가 필요할 때 call

framework ex :django - call하지 x. framework가 나를 call. 내가 프레임워크의 규칙을 따라야함.

어디에 템플릿,컨트롤러,뷰를 넣을지.. 장고에서 어드민 패널을 만들고 싶다면, 무조건 코드를 admin.py에 써야함.

< spring framework >

* 자바를 이용해서 웹개발 시 사용
* dependency injection, transaction management를 깔끔하게 제공
* enterprise application을 만들때 중요한 요소

텍스트, 자연이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

EE영역을 SE+spring으로!

아직까진 선택가능하긴함



* 웹개발을 할 때, 3가지 범주

-MVC

-트랜잭션

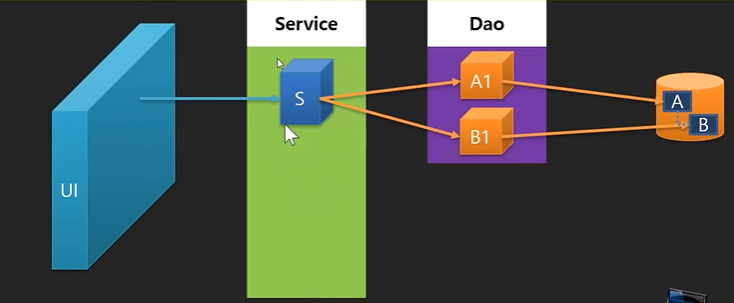
-인증과 권한

* 🡪 spring의 기본적 덕목 : DI , AOP, Servlet Filter , MVC+ DB ( 오라클 SQL )

텍스트, 자연, 일몰, 구름이(가) 표시된 사진

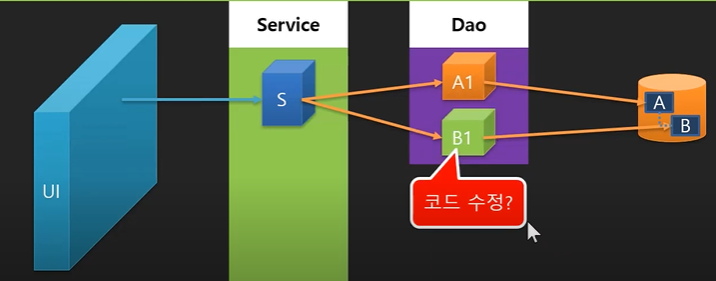
자동 생성된 설명

< 느슨한 결합력과 인터페이스 >

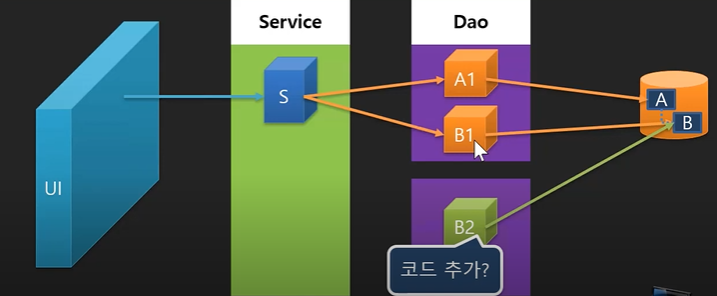


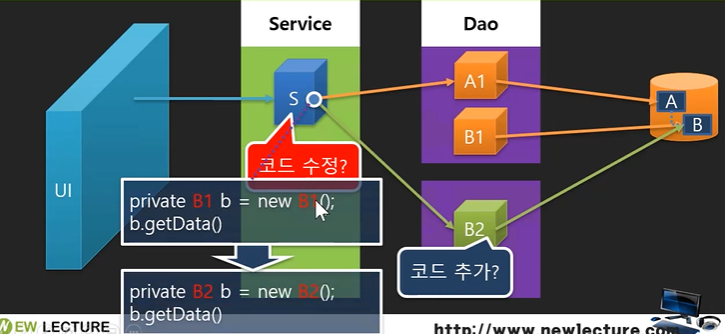
-어떤 객체를 수정해야 한다면?

~~1. 소스코드를 수정한다~~

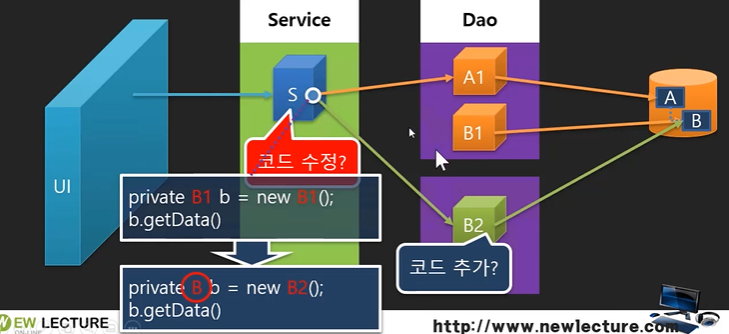


~~2. 소스코드를 추가해서 덮어쓴다.~~

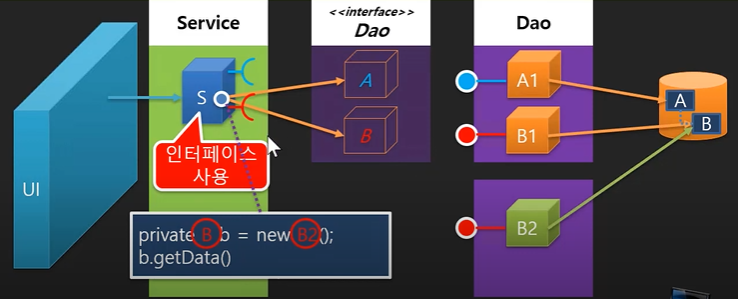


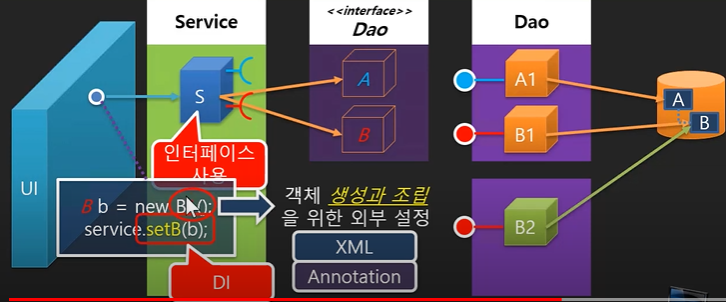
* ~~결국 서비스 코드를 수정해야함.~~
* 

3. 덮어 쓰도록 하되, 가능한 코드 수정은 없는 방법



* 인터페이스



’

<스프링이란>

* 여러가지 기술들의 모음



스프링데이터 : crud를 편리하게 사용할 수 있도록 도와줌

스프링세셔 : 세션기능

시큐리티 : 보안

rest docs : api 문서화

배치 : 데이터를 처리할 때, 배치처리에 특화된 기술

클라우드 : 클라우드

그 외에도 굉장히 많은 기술들이 있음. spring.io

* 스프링 부트 : 이 모든 기술들에 대해, 편리하게 사용하도록 해주는 것.
* 스프링 프레임워크 \*\*\*
* 텍스트이(가) 표시된 사진

  자동 생성된 설명
* 텍스트이(가) 표시된 사진

  자동 생성된 설명
* 빌드 + 서버띄우는것까지 몇줄의 코드
* 다양한 라이브러리를 starter로 한번에 편리하게 가져옴?
* 버전 궁합 ? 스프링프레임워크버전에 따른 맞는 라이브러리버전을 자동 구성해줌. 최적화.
* 모니터링
* 디폴트 설정을 편리하게 해줌

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 스프링의 핵심 컨셉

텍스트이(가) 표시된 사진

자동 생성된 설명

* 결과물
* 핵심은?
* 텍스트이(가) 표시된 사진

  자동 생성된 설명

< 웹개발 >

* web이란 : world wide web은 인터넷에 연결된 컴퓨터를 통해, 사람들이 정보를 공유할 수 있는 전 세계적인 정보 공간을 말함.
* web의 용도

1. web site : html로 구성된 여러 사이트 들
2. API web service : application programming interface
3. user interface

* web의 기본 3요소

URI : Uniform Resource Identifier 리소스 식별자. 모든 정보에 접근 할 수 있는 정보

HTTP : Hypertext Transfer Protocol 어떤 프로토콜을 통해서 주고 받을 것인가? 어플리케이션 컨트롤

HTML : Hyper Text Markup Language 하이퍼미티어 포맷, xml을 바탕으로한 범용 문서 포맷. 사용자가 알아보기 쉬운 형태로 표현

<Rest API> 백엔드,서버엔지니어는 rest api를 연동,제공할 수 있어야함. 지켜야하는 rest 규약

* REST : representational state transfer , 자원의 상태 전달. ( 네트워크 아키텍쳐)

1. client, server : 클라이언트와 서버가 서로 독립적으로 분리되어 있어야 한다. 독립적이지 않거나, 정보가 너무 밀접하면 안됨
2. stateless : 요청에 대해서 클라이언트의 상태를 서버에 저장하지 않음.

햄버거가게ex. 치즈버거주세요. 아그리고 콜라추가해주세요./ 서버는 클라이언트 요청에 대한 상태를 저장하지 않기 때문에 모든 요청에 대해 새롭게 나가야함. 🡪 치즈버거와 콜라 주세요.

1. cache. 클라이언트는 서버의 응답을 임시 저장(cache)해서 재사용할 수 있어야함. 서버의 부하 낮춤.
2. 계층화. : 서버와 클라이언트 사이의 방화벽, 게이트웨이, 프록시서버 등 다양한 계층 형태로 구성 가능해야하고, 확장할 수 있어야함.
3. 인터페이스 일관성 :. 인터페이스의 일관성을 지키고, 아키텍처를 단순화 시켜서 작은 단위로 분리. 클라이언트,서버가 독립적으로 개선될 수 있어야함. 서버가 바뀌어도, 클라이언트에 지장이 없도록. 클라이언트가 바뀌어도 서버에 지장이 없도록 인터페이스의 일관성 지켜야함.
4. code on demand( optional) : javascript등 특정한 기능을 서버로부터 클라이언트가 전달받아 코드를 실행할 수 있어야함.

< REST를 잘 사용했는지 판단할 수 있는 인터페이스의 일관성 >

다음의 일관성을 잘 지켰는가

1. 자원의 식별

: 웹기반의 rest에서는 리소스 접근을 할 때, URI를 사용한다. URI에는 자원을 식별할 수 있는 정보가 담겨야 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

🡪100번째 유저라는 것을 식별하도록 사용됨.

1. 메시지를 통한 리소스 조작

: 다양한 방식으로 데이터 전달. html, xml, json, text등

* 어떤 타입의 데이터인지 http header에 content-type을 통해 지정해줌.
* 또한, 리소스 조작을 위해서 데이터 전체를 전달하지 않고, 메시지로 전달함.

클라이언트와 서버가 독립적으로 확장 가능.

1. 자기 서술적 메시지

: 요청하는 데이터가 어떻게 처리 되어져야하는지 충분한 데이터를 포함해야함.

🡪( http 기반의 REST 에서는 ) HTTP method, header정보, URI정보를 가지고 표현.

텍스트이(가) 표시된 사진

자동 생성된 설명

🡪그 외에 담지 못 한 정보들은, URI의 메시지를 통하여 표현함.

1. 애플리케이션 상태에 대한 엔진으로써 하이퍼미디어

: REST API를 개발할 때, 단순히 한가지 client요청에 대한 데이터만 응답 해주는 것이 아니라, 그 외에 서버가 가지고 있는 리소스에 대한 링크정보까지 포함해서 전달해주면, 우리는 REST Ful하다고 표현. REST API라고 부름.

🡺현업에서는 잘 사용하고있지 않기도 함. 주소정보까지 매핑하게 되면 불필요한 정보까지도 작성해야하는 경우가 생겨서.

< URI >

* URI , uniform resource identifier

: 인터넷에서 특정 자원을 나타내는 주소값. 해당값은 유일하며, 응답은 달라질 수 있음.

* URL, uniform resource locator

: 인터넷 상에서의 자원, 특정 파일이 어디에 위치하는지 식별하는 주소. [ URI의 하위 개념]

<URI 설계 원칙 >

1. 슬래시 구분자(/)는 계층 관계를 나타내는 데 사용함.
2. URI 마지막 문자로 (/)는 포함하지 않음.
3. 하이픈(-)은 URI 가독성을 높이는데 사용함.
4. 밑줄(\_)은 사용하지 않음
5. URI경로에는 소문자가 적합함.
6. 파일 확장자는 URI에 포함하지 않음. ( EX. /web-master.jsp )
7. 프로그래밍 언어에 의존적인 확장자를 사용하지 않음. ( EX. /web-master.do )
8. 구현에 의존적인 경로를 사용하지 않음. ( EX. /servlet/classes/java/… )
9. 세션 아이디를 포함하지 않는다. ( ? session-id = abcdef )
10. 프로그래밍 언어의 Method명을 이용하지 않음 ( ?action=intro )
11. 명사에 단수형 보다는 복수형을 사용해야함. 컬렉션에 대한 표현은 복수로 사용 . ( EX. /classes/java/… )
12. 컨트롤러 이름으로는 동사나 동사구를 사용함.
13. 경로 부분 중 변하는 부분은 유일한 값으로 대체한다.



1. CRUD 기능을 나타내는 것은 URI에 사용하지 않는다. – 메서드를 이용해 서버에 요청하는 것이 원칙.  
   
2. URI Query Parameter디자인

쿼리부분으로 컬렉션 결과에 대해서 필터링 할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. API에 있어서 서브 도메인은 일관성 있게 사용

🡪 도메인에서 오픈api를 제공한다면, 도메인앞에 api. or api-를 붙여준다.

1. 클라이언트 개발자 포탈 서브 도메인은 일관성있게 만듦.

* dev- or developer-를 붙여줌.