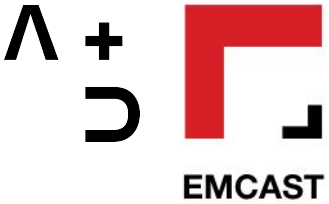
ABOVE+
BEYOND



Session 1

Leveling up from Monolithic to Multi-module

Software Architecture Designing experience

Jack Jungnam Lee
EMCAST Software Technology R&D Team

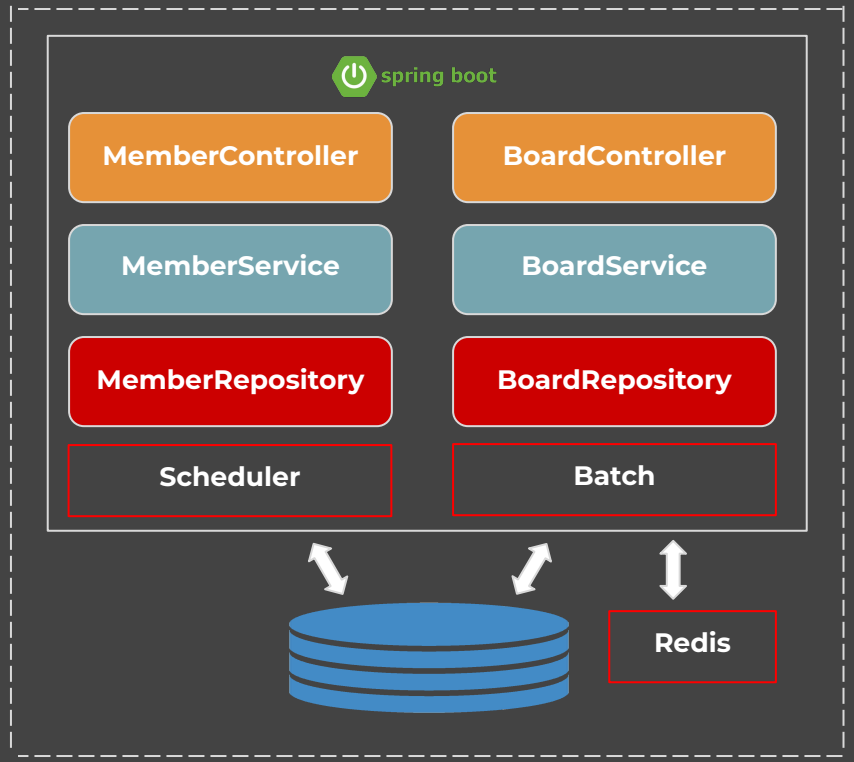
Contents Table

Monolithic Architecture

A traditional software architecture style where the entire application is designed and deployed in a single unified unit.

Pros

1. Simplicity
2. Performance
3. Simplicity of testing
4. Low Latency
5. Cost efficiency

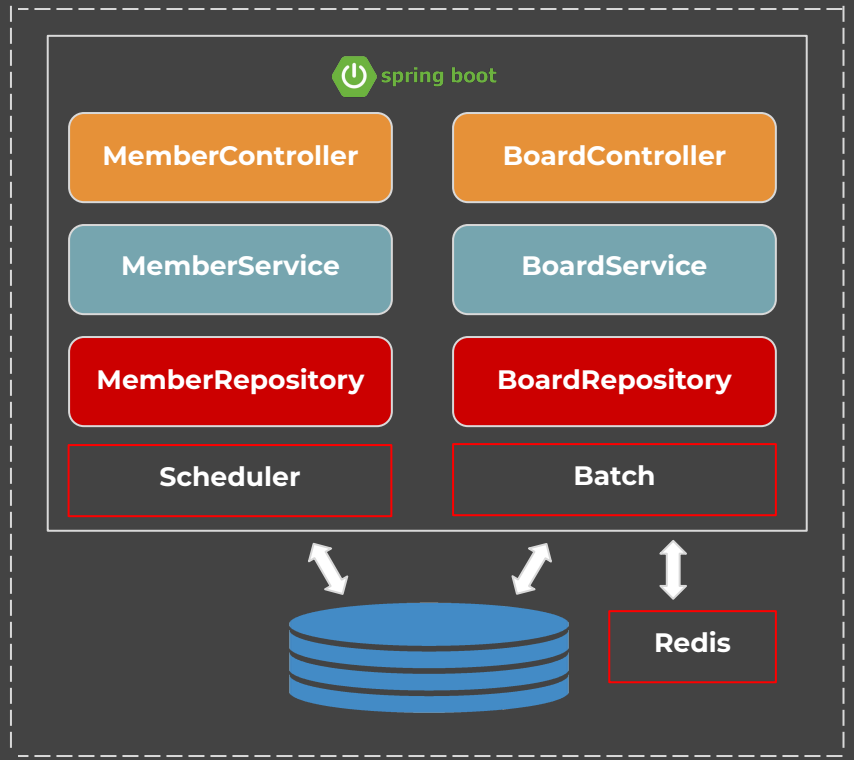


Monolithic Architecture

A traditional software architecture style where the entire application is designed and deployed in a single unified unit.

Cons

1. Scalability
2. Elasticity
3. Flexibility
4. Maintenance
5. Distribution

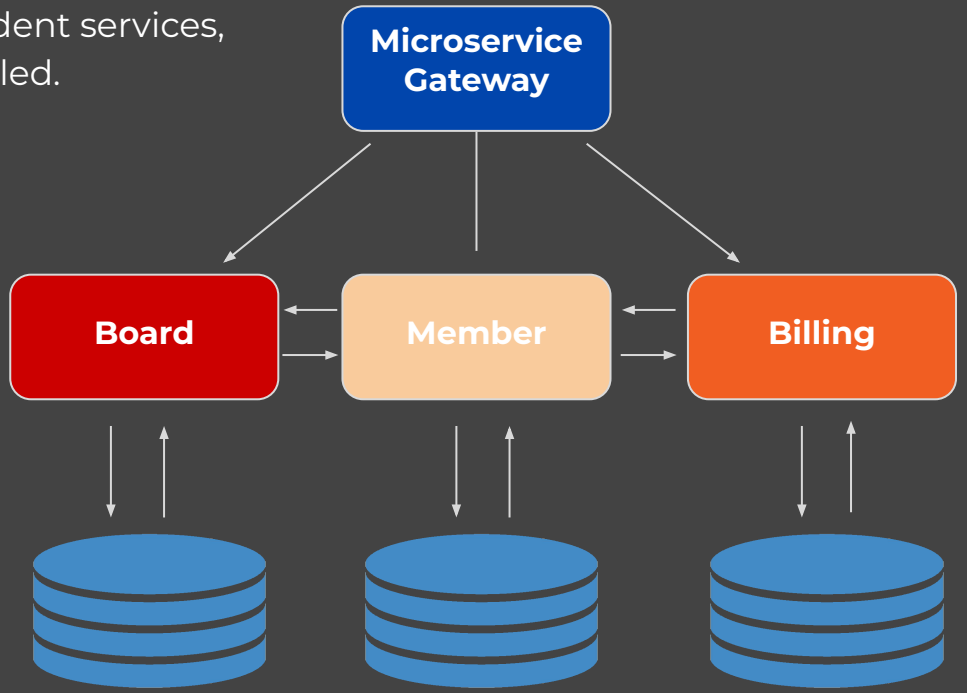


MicroService Architecture

A software development approach in which applications composed of small independent services, that are highly cohesive and loosely coupled.

Pros

1. Scalability
2. Flexibility
3. Resilience
4. Speedy development
5. Easy maintenance
6. Improved fault isolation
7. Team optimization

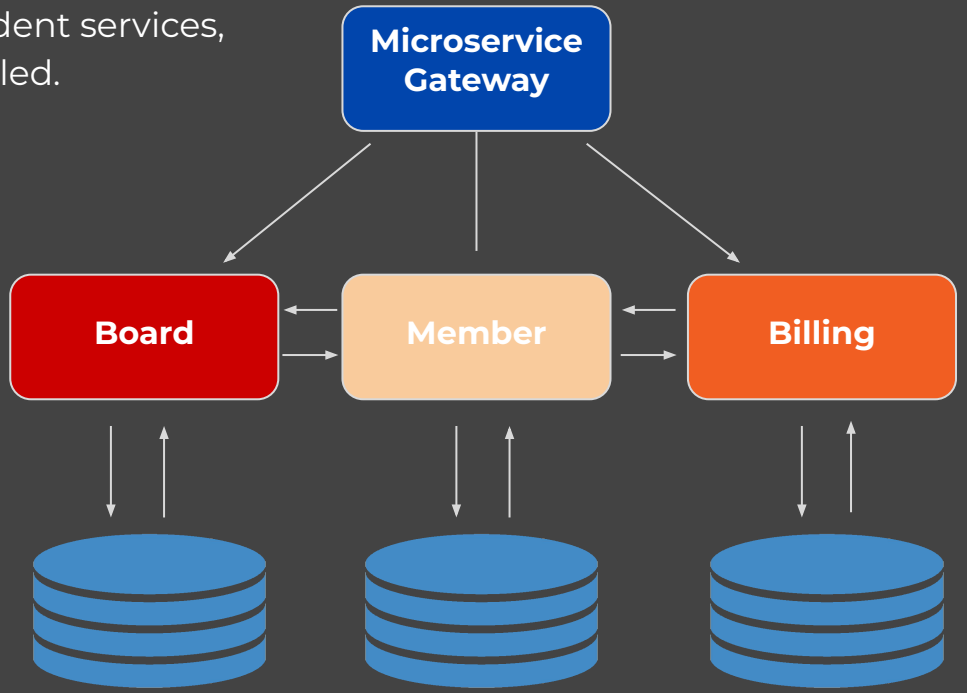


MicroService Architecture

A software development approach in which applications composed of small independent services, that are highly cohesive and loosely coupled.

Cons

1. Complexity
2. Increased overhead
3. integration problem
4. Test Complexity
5. Increased operational dressability
6. Security Issues

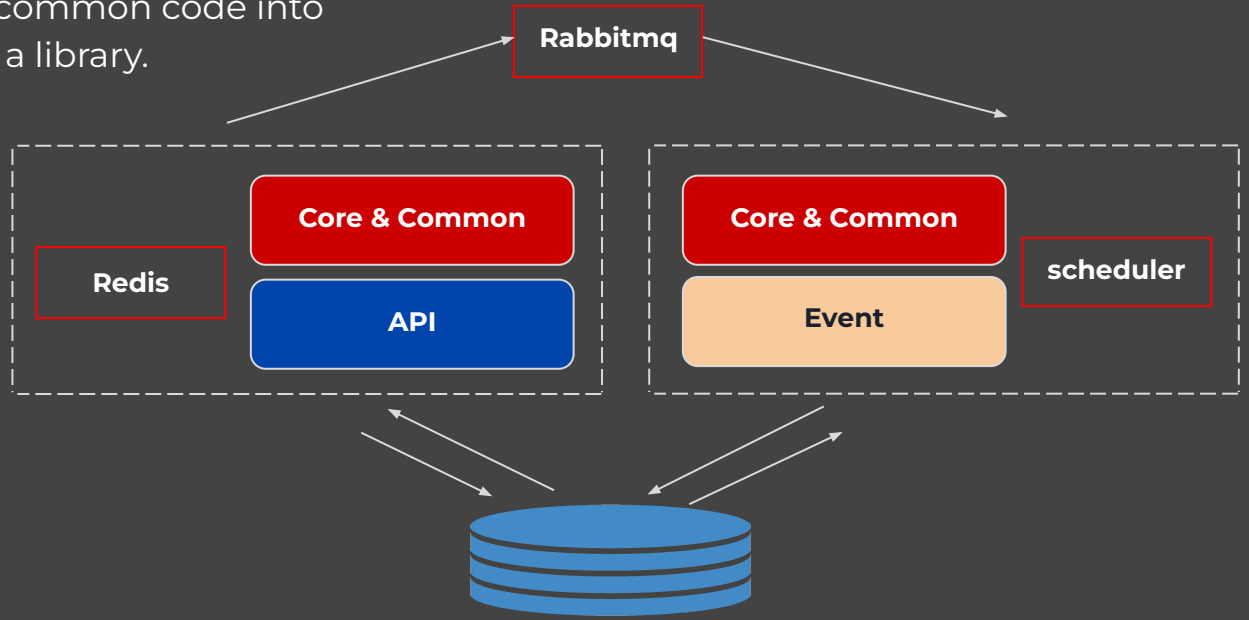


Multi-Module Architecture

An architecture that implements and uses only the business logic required by individual modules by separating the domain and common code into modules and using them as a library.

Pros

1. Scalability
2. Flexibility
3. Resilience
4. Speedy development
5. Easy maintenance
6. Improve Fault Isolation
7. Simplicity
8. Low latency

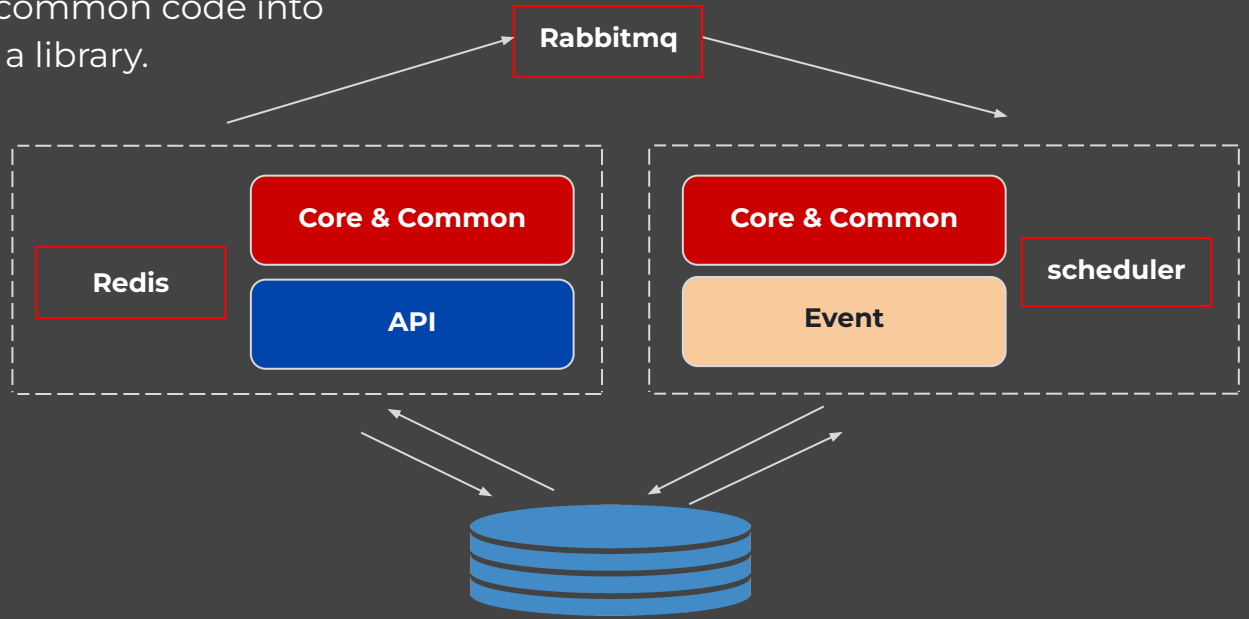


Multi-Module Architecture

An architecture that implements and uses only the business logic required by individual modules by separating the domain and common code into modules and using them as a library.

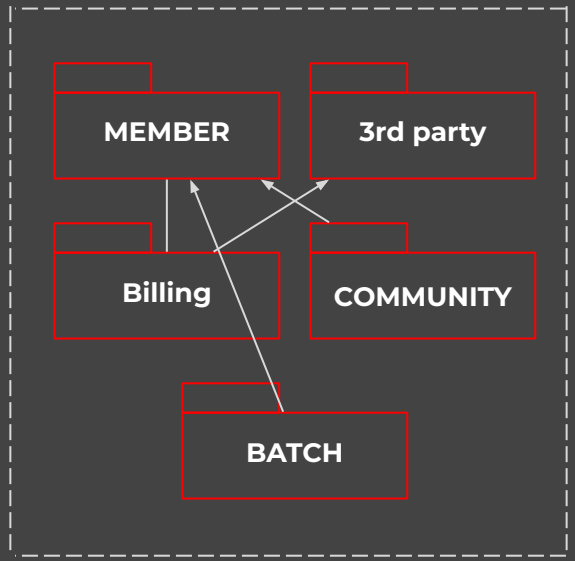
Cons

- Difficult to achieve Team autonomy



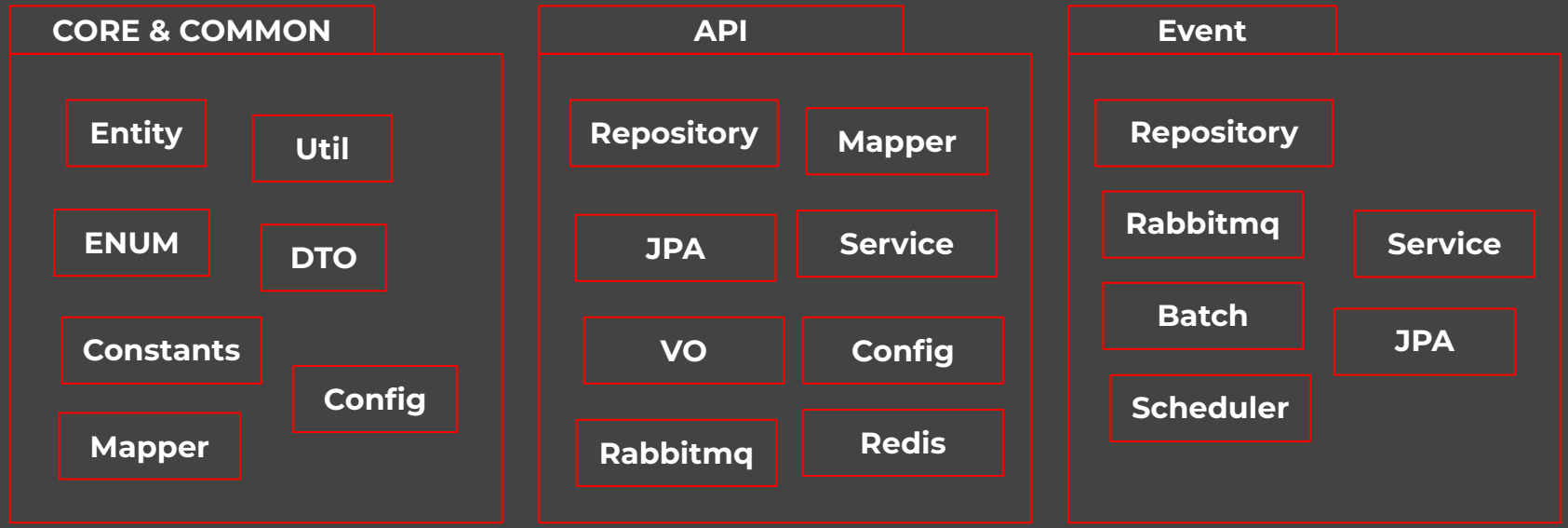
Reasons for Transitioning to Multi-Module Project

Monolithic? Microservice? Choosing the right architecture for service

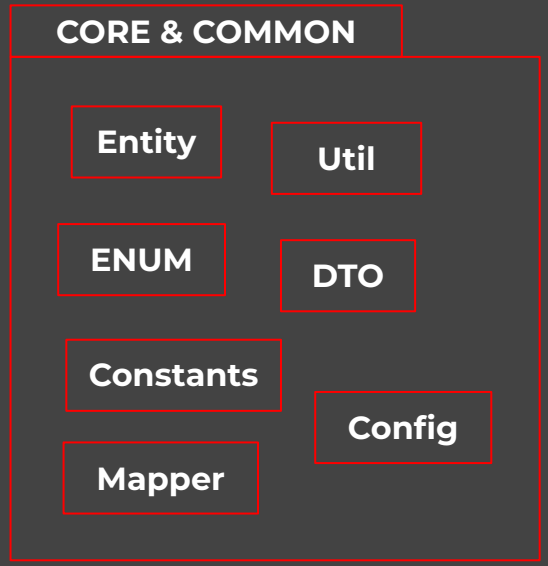


- Performance issues due to growing services
- Unable to transit to microservices due to strong coupling between entities
- **Testing one function has an effect on another**

Multi-Module Configuration

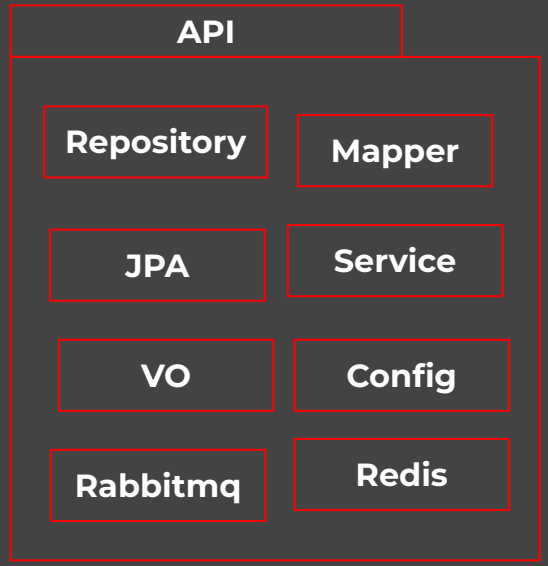


Multi-Module Project Structure (1 of 3)



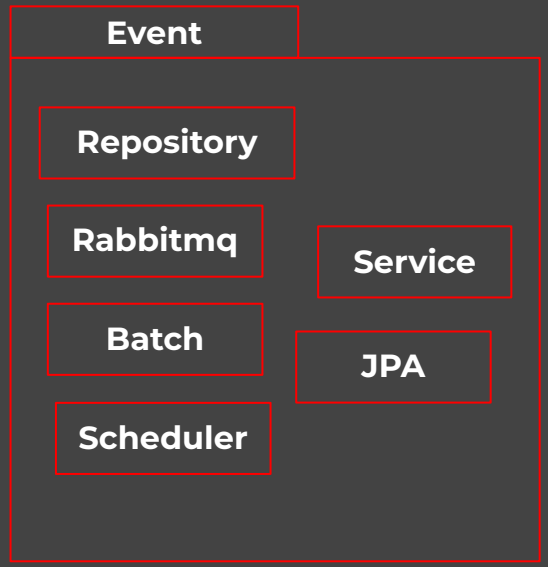
- Core and Common separation in Monolithic Architecture.
- Relocation of Util, Enum, Constants Class for use within the entire module.
- Service (Business Logic) and Repository are implemented in separate modules.
- Mapper for conversion of Entity and DTO
- Common Config settings

Multi-Module Project Structure (2 of 3)



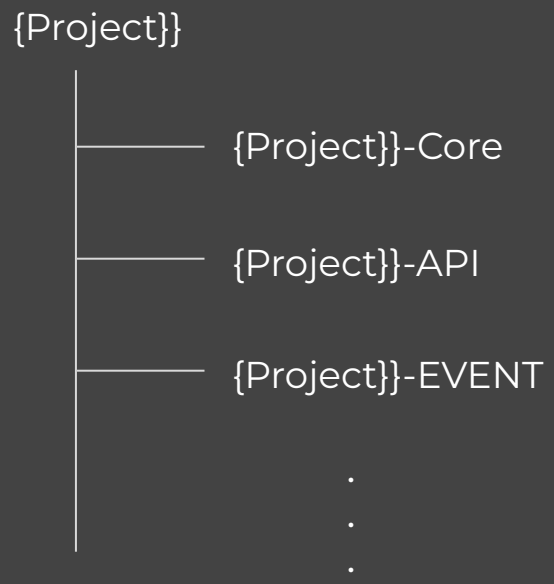
- Development of Business logic(Service) & Query.
- Mapper creates a class for converting data (VO) queried in the repository to DTO.
- Sends a message to the message broker (Rabbitmq) when an asynchronous event occurs.

Multi-Module Project Structure (3 of 3)



- Handle the asynchronous events from messages received from Rabbitmq.
- Proceed with the Scheduler (Batch) process.

Complete Project Configuration



- When core development and debugging are required, the entire module must be deployed separately
- Modules except core are distributed separately when debugging and functionality is added
- API projects can also be separated into multiple modules (Billing, Member, Admin, etc.)

Effectiveness of Multi-Module Architecture

Modularity

Reusability

Scalability

Collaboration

Effectiveness of Multi-Module Architecture

Modularity

- Breaking up the software system into smaller modules makes it easier to understand, maintain, and modify.
- Each module can be designed, tested, and implemented independently of other modules, reducing system-wide complexity.

Effectiveness of Multi-Module Architecture

Reusability

- Modules can be reused across multiple projects, saving time and effort in software development.

Effectiveness of Multi-Module Architecture

Scalability

- Multi-module architecture makes it easier to scale systems as application requirements change.
- Additional modules can be added as needed to supporting new features or accommodating growing user loads.

Effectiveness of Multi-Module Architecture

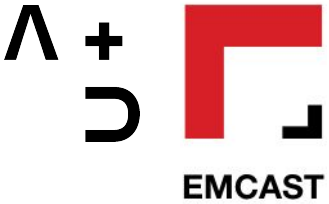
Collaboration

- Multi-module approach can facilitate collaboration between developers working in different parts of the system.
- Multiple developers can work independently on the module and integrate the modules to create the final product.

THANK YOU

감사합니다

ABOVE+
BEYOND



Session 2

Building a team consisting of complementary skilled individuals into a well rounded team

Saige Hoonyoung Choi
EMCAST Software Technology R&D Team

Hello! I'm Choi Hoon-young, the Product Owner

What does Product Owner mean?

The role is to optimize the product backlog process in order to maximize the value of the product, until the product meets the user, or until it becomes a more 'loved' product if it has already met the user.

Set directions,
Prioritize work items and product backlog.
Collect human and material resources,
Manage the surrounding environment so that everyone is fully present and engaged in the work performance.

As the product development team leader,

I always look for ways to make a better and solid organizations,
where all team members will be able to work in a healthy environment.

Work With Agility

What is Agile?

Execute fast, fail fast, and work fast to improve based on that failure.

Agile vs Waterfall

- Waterfall
  - Sequential development methodology where each task falls from top to bottom like a waterfall.
  - Clear direction & specific deliverables for upcoming product, ideal for highly detailed requirements.

- Agile
  - A resilient methodology that quickly launches products at regular intervals to add and modify their needs to meet clients needs and changing environments.
  - The biggest advantage is its speed and flexibility.

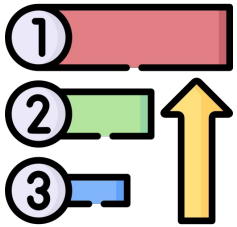
=> Agile is open to embrace changes and adapt effectively in response.

Work With Agility

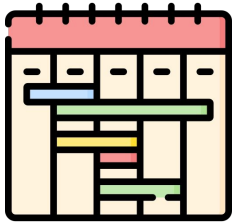
Things happening in the real world

Constant changing of events

Hard to predict hotfix

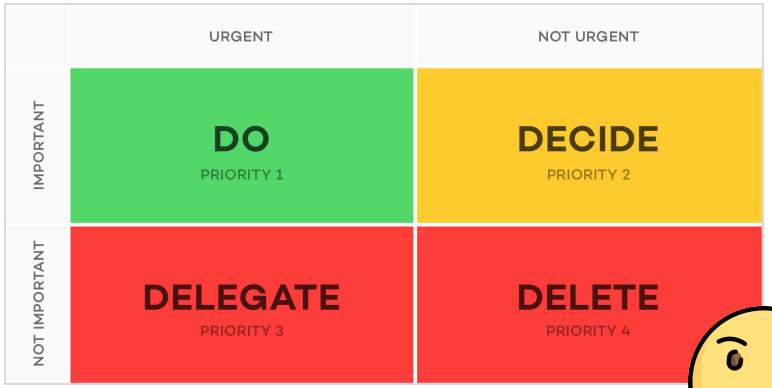


Change of priority according to business needs

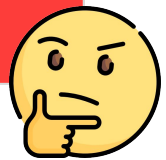


Extension of testing and debugging periods

Work With Agility



(The Eisenhower Matrix in my head)



Important and Urgent tasks must & can be done right away,
however in reality, it's constantly bombarded with **Unimportant but Urgent** tasks.
However, **Non-urgent but Important things** are often at the center of what the team wants to do.

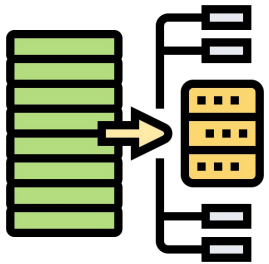
Only by doing this can break through in the long run and create opportunities.

How can we be flexible enough to meet your organization's needs and solve problems without missing the central point?

How ANCHOR team manages the project

1. Sets the work into units - not too large or not too small

- Break down the work in units that can be completed within 0.5 to 3 days.
- Too large of a work at once brings challenges in maintaining the work agility.
- Right work size that are clear, specific are more motivating.
- This is because **small accomplishments leads to the completion of larger goals and bigger rewards & achievements.**



How ANCHOR team manages the project

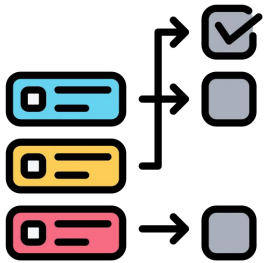
2. All members understands the context of their particular task

- One must understand
  which part of this task is and how this task that
  I'm doing leads to the next one.
- One must look & observe at the colleague's work that affects with my task.
- Likewise, one must be aware of the task of the colleagues
  who are affected by my work.

How ANCHOR team manages the project

3. Always check the remaining time to the targeted schedule and periodically rearrange the work according to priority

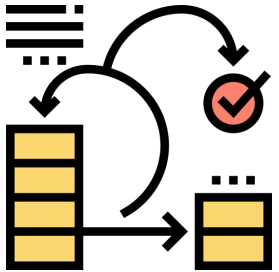
- Check for remaining issues at intervals of 2 to 3 days per week, depending on the remaining period.
- Address urgent issues related with usability issues first. Depending on the speed of progress and the duration remaining, the scope of the issue may narrow or widen.
- Priority checks and work arrangements enable all members to focus on achieving their goals.



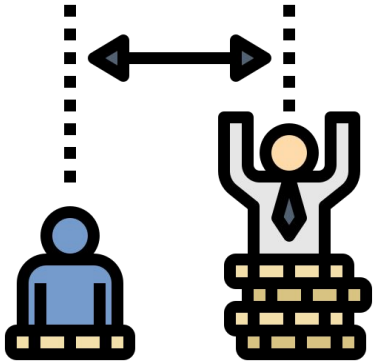
How ANCHOR team manages the project

4. Utilizes the backlog effectively

- When carrying out a project, there are times when things that were not planned become an issue.
- If it's concluded as something that needs to be solved right away, we start it right away, but if it's something that is difficult to start right away or that can be postponed, then we write it down and keep in the backlog.
- The key is to empty the head, instead write down the issues in detail so that we don't rely on memory.



Closing the Gap Between Ideal and Reality



Closing the Gap Between Ideal and Reality

Writing Daily Scrum Notes

- Daily scrum notes give an opportunity for the team to see each other's work.

1) Work planned yesterday
2) Work in progress or completed
3) Issue and schedule adjustment
4) Work scheduled for today

Closing the Gap Between Ideal and Reality

Managing the Release Board
along with the work's schedule and distribution status

- Create the distribution process in an easy-to-understand pin board as > **Merge Request - Develop - Stage - Operate** distribution

- Use the Kanban board section's name in synchronization with the **GitFlow's branch name**

Closing the Gap Between Ideal and Reality

To breakdown and number the work even if it's the same work

- Testing 1st, Debugging 1st, Testing 2nd, Debugging 2nd ...
- Output Planning 1st, Reviewing & feedback 1st,
  Output Planning 2nd, Reviewing & feedback 2nd ...
- The person in charge of each task can complete the task
  when they are done with or without feedback.

The power of taking meeting minutes!

- **All decision-making situations,** from short calls to conference calls, are considered as meetings.
- From small decisions to large directional decisions, **we take meeting minutes.**
- It has the effect of announcing and declaring to everyone.
- It helps members follow the context of what the topic is being discussed intensely at this point.
- Whenever one can't remember on how it was discussed? **we can search and recall it.**

Principles of communication & conduct

keep something to oneself

- Does not hide things that might be an issue or turned out to be an issue later on.
- Instead, we favor having honest & open communication and solve it together.

Keep mutual promises

- To build and maintain mutual trust, try to keep even small promises faithfully.

Principles of communication & conduct

Create rules, abide by, and give feedbacks

- Anyone can suggest and give feedback, also at the same time we don't forget to abide by the rules made under agreement.
- Follow and discuss again to edit the rules.

Try to stay in good condition

- Takes good care of one's physical, mental, and emotional condition so that we can focus on work at the fixed working time.
- Manage one's emotion so that one can enjoy working with the colleagues in a harmonious environment.

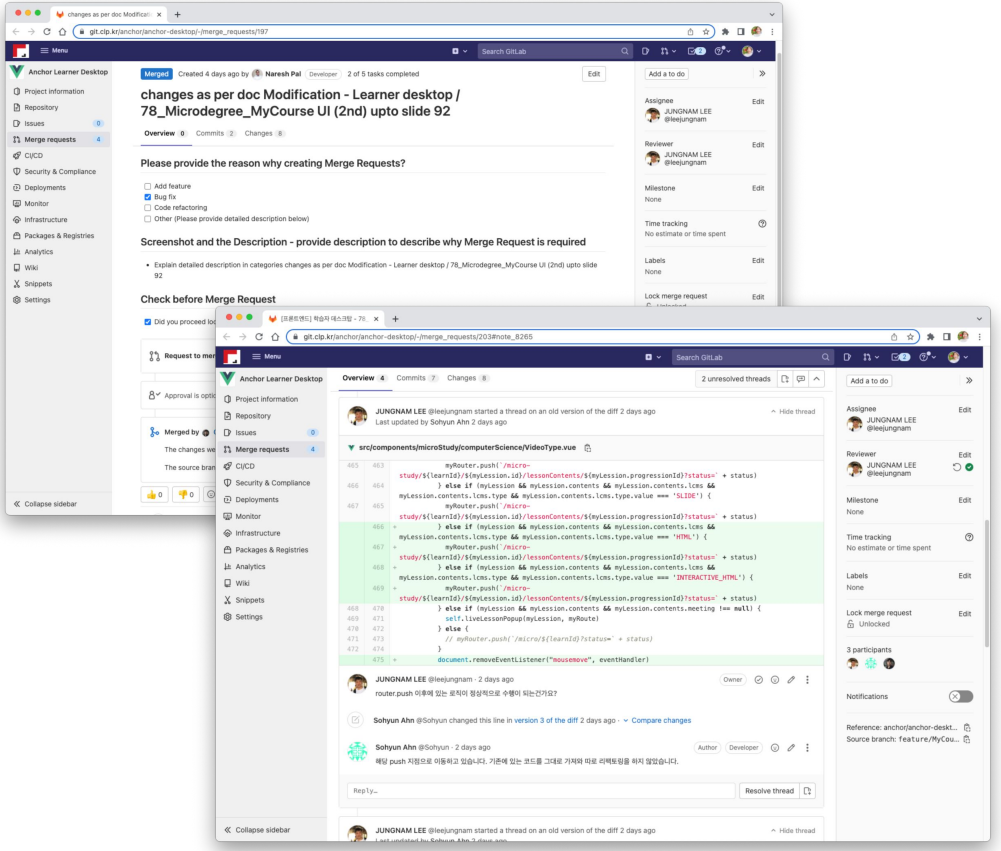
Create a Retrospective Culture

Code review

1. Code review using GitFlow policy.
2. Daily Code review meeting before end of the work.

Code Review

- Create a **merge request** and request for a review according to the GitFlow policy.
- **Reviewers can request modification by commenting** before merge, and merge to develop branch when confirmed.



Code Review

- In addition to coding reviews according to the GitFlow policy, we have **a review meeting every day** before the end of the day.
- We proceed at once without distinction between the front end and the back end.In all these process, the PO participates.

ANCHOR CODE REVIEW CORE VALUE

- Check each other for compliance with whether **the convention guide** is followed and check to see if **clean code** is created.
- **Open the code** that was written **and explain** what problem was solved and why it solved like that.
- **Suggests** if there is a better solution.
- **Review allows one to gain prior knowledge of code written by others, and analyze it quickly when the tasks is to be done later on.**
- At the end of the end, it is not at all easy to invest a little bit of time every day, however **this helps to write consistent code overall and reduce trial and error.**
- The most important thing is to share the fundamental problems of the project found and **use it as a starting point for thinking about improvement.**

R&D Study

- Anyone can suggest a topic, from what tools they're going to use, to new technologies they want to work on with.
- Usually, issues raised repeatedly through planning meetings, inspections, and code reviews leads to research topics.
- Since issues are found within the consensus that has been accumulated little by little, it naturally motivates and the results does have a direct impact on practice.

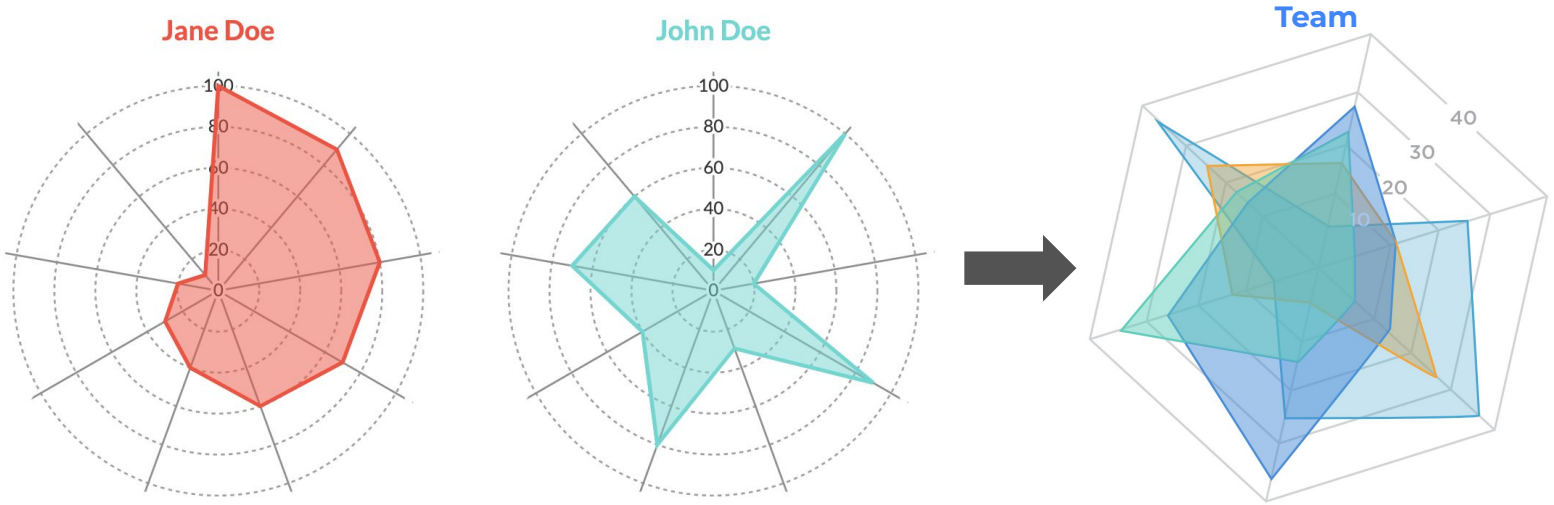
The reason why knowing and understanding the other person is important for team work

Misunderstanding each other because they don't know each other well
or being overly polite is also the cost of communication.

So invest time to get close.
Through this, we expect to have more effective, smooth communication and cooperation.

Hence, we aim for a round team with sharp strengths

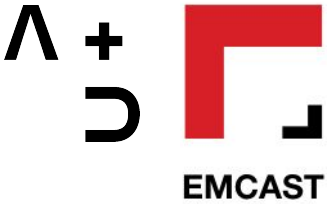
Hoping this roundly well created team will evolve for betterment.



THANK YOU

감사합니다

ABOVE+
BEYOND



Session 3

Using object-oriented programming that is easier to read and manage than SQL code

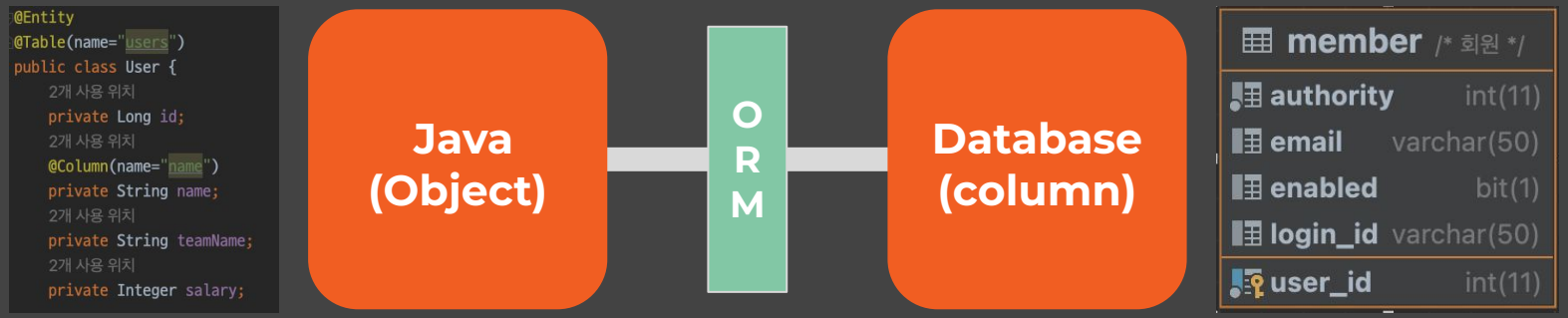
JPA Technology Introduction and implementing experience

Jack Jungnam Lee
EMCAST Software Technology R&D Team

Contents Table

What is ORM?

- ORM short for Object Relational Mapping is a programming technique used to map data between a relational database and the heap of an object-oriented programming language.
- The background of ORM's existence is due to the inconsistency between object-oriented programming and relational databases.



Differences between Mybatis and JPA

- **JPA:** Java Persistence API (JPA) is the standard interface of Object-Relational Mapping (ORM) technology for Java languages
- **Mybatis:** A framework (SQL Mapper) that maps data from objects and relational databases to SQL created by developers.

	JPA	Mybatis
Pros	• RDB-independent usability (code can be recycled when DB changes) • Basic CRUD and paging are implemented beforehand, enabling you to focus on business logic • Resolving paradigm mismatch between Object and RDB	• The running curve is gentle • SQL granularity is easy to change • Easier than JPA when using dynamic queries
Cons	• High running curve • Mistakes in mapping design results in performance degradation.	• Dependant on RDB • Mapper creation, interface design required

Key Benefits of Using JPA

1. Increase productivity
2. Improved Reusability
3. Performance Improvements
4. Improved maintenance
5. Reduce database dependencies

Key Benefits of Using JPA

1. Increase productivity

- Since JPA deals with Data around objects, it can reduce the burden of creating SQL.
- Data manipulation using objects increases the readability and maintenance of code.
- Developers productivity increases.

```
users.setName("jack");
users.setTeamName("anchor");
userJpaRepository.save(users);
```

Key Benefits of Using JPA

2. Improved Reusability

- Object-oriented code can be easily reused.
- This can help improve code reuse and reduce development costs.

```java
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
@DiscriminatorColumn
public abstract class Food {
    @Id @GeneratedValue
    @Column(name = "FOOD_ID")
    private Long id;

    private String name;
    private int price;
}
```

```java
@Entity
@DiscriminatorValue("P")
public class Pasta extends Food {
    private String restaurant;
}
```

Key Benefits of Using JPA

3. Performance improvements

- Through its persistence context, query execution results are cached and unnecessary queries are prevented leading to increase in performance.
- Since JPA supports Lazy loading, you can optimize performance by adjusting the timing of query execution.

```
@OneToMany(fetch = FetchType.LAZY,
        mappedBy = "article",
        cascade = CascadeType.ALL,
        orphanRemoval = true)
private List<ArticleTag> tags = new ArrayList<>();
```

Key Benefits of Using JPA

4. Improved maintenance

- Due to its automatic process mapping between databases and objects, it simplifies the response to database schema changes.
- This greatly helps improve in the maintenance of code.

```java
@Entity
@Table(name="users")
public class User {
    2개 사용 위치
    private Long id;
    2개 사용 위치
    @Column(name="name")
    private String name;
    2개 사용 위치
    private String teamName;
    2개 사용 위치
    private Integer salary;

    @Id
    @GeneratedValue
    public Long getId() {
        return id;
    }
```

Key Benefits of Using JPA

5. Reduce database dependencies

- Due to its reduce dependencies with databases, it makes easier to cope with database changes.
- Since JPA is responsible for mapping with databases, developers does not need to interact directly with databases. Hence, this reduces the burden on developers to handle databases.

```java
@Entity
@Table(name="users")
public class User {
    2개 사용 위치
    private Long id;
    2개 사용 위치
    @Column(name="name")
    private String name;
    2개 사용 위치
    private String teamName;
    2개 사용 위치
    private Integer salary;

    @Id
    @GeneratedValue
    public Long getId() {
        return id;
    }
```

Key Concepts of JPA

1. Entities
2. Entity manager
3. Persistence Context
4. Mapping
5. Query
6. Relationship
7. Inheritance

Key Concepts of JPA

1. Entity

- An entity is an object that represents a table stored in a database in JPA.

```java
@Entity
@Table(name="users")
public class User {
    2개 사용 위치
    private Long id;
    2개 사용 위치
    @Column(name="name")
    private String name;
    2개 사용 위치
    private String teamName;
    2개 사용 위치
    private Integer salary;

    @Id
    @GeneratedValue
    public Long getId() {
        return id;
    }
```

Key Concepts of JPA

2. EntityManager

- An EntityManager is an object that manages an entity. The entity manager allows you to read, create, update, and delete Entities.

```
EntityManager em;

public void join(String name, int age) {
    Member member = new Member();
    member.setName(name);
    member.setAge(age);


    EntityTransaction tx = em.getTransaction();

    try {
        tx.begin();

        em.persist(member);

        tx.commit();
    } catch(Exception e) {
        tx.rollback();
    } finally {
        em.close();
    }
}
```

Key Concepts of JPA

3. Persistence Context

- Persistence Context is an object that Entity Manager uses to manage Entities.
- Persistence Context manages the life cycle of Entity and ensures the work of Entity and Database.

Key Concepts of JPA

4. Mapping

- Mapping : Defines a mapping rule between Entity and Database.
- JPA allows you to define mapping rules between Objects and Tables as annotations or XML files.

```java
@Entity
@Table(name="users")
public class User {
    2개 사용 위치
    private Long id;
    2개 사용 위치
    @Column(name="name")
    private String name;
    2개 사용 위치
    private String teamName;
    2개 사용 위치
    private Integer salary;

    @Id
    @GeneratedValue
    public Long getId() {
        return id;
    }
```

Key Concepts of JPA

5. Query

- Query : JPA provides an object-oriented query language called Java Persistence Query Language (JPQL).
- JPQL can create queries for Entity objects and uses SQL-like grammar.

```
SELECT m FROM Member m JOIN m.team t WHERE t.name = '팀A'
```

6. Relationship

- Defines the relationship between Entity Objects. JPA supports a variety of relationships, including Many-To-One, One-To-One, One-To-Many, and Many-To-Many.

```
@OneToMany(mappedBy = "role")
private List<User> users = new ArrayList<~>();
```

Key Concepts of JPA

7. Inheritance

- JPA allows you to map inheritance relationships between objects. This feature enhances reusability and maintainability of entities.

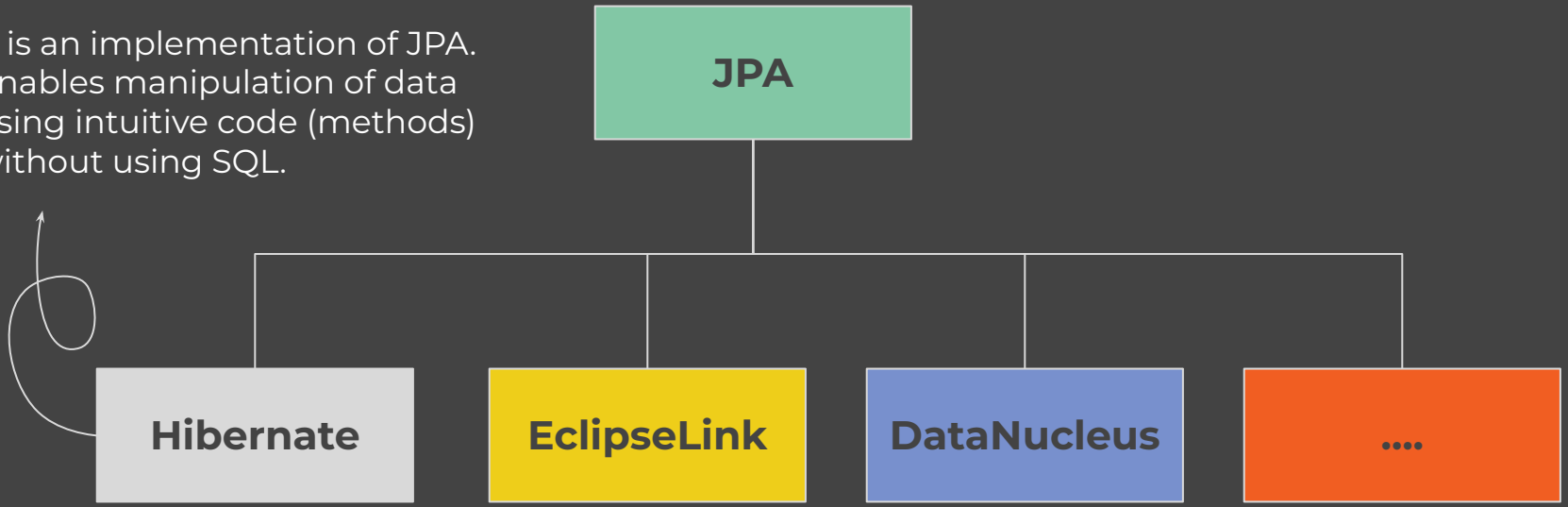
```java
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
@DiscriminatorColumn
public abstract class Food {
    @Id @GeneratedValue
    @Column(name = "FOOD_ID")
    private Long id;

    private String name;
    private int price;
}
```

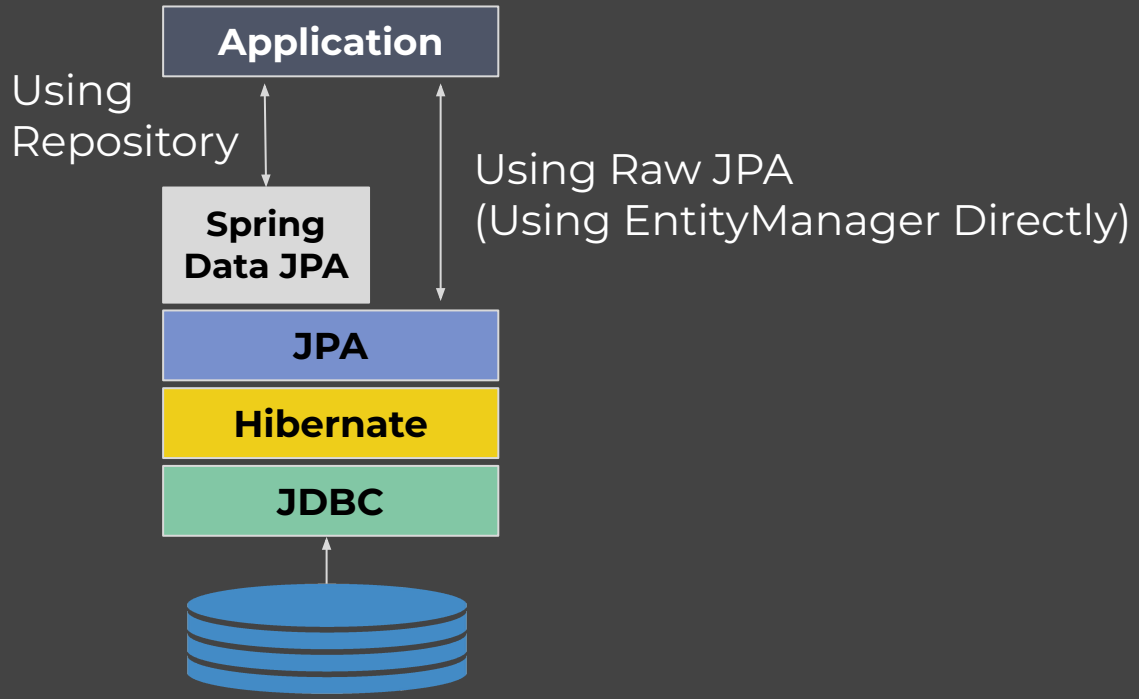
```java
@Entity
@DiscriminatorValue("P")
public class Pasta extends Food {
    private String restaurant;
}
```

What is Hibernate?

It is an implementation of JPA. Enables manipulation of data using intuitive code (methods) without using SQL.



What is Spring Data JPA



Repository with Entity Manager

```java
@Repository
@RequiredArgsConstructor
public class OrderRepository {

    private final EntityManager em;

    public void save(Order order){
        em.persist(order);
    }

    public Order findOne(Long id){
        return em.find(Order.class, id);
    }

    public List<Order> findAll() {
        return em.createQuery("select o from Order o", Order.class)
                .getResultList();
    }

    public List<Order> findAllWithItem() {
        return em.createQuery(
                "select distinct o from Order o" +
                        " join fetch o.member m" +
                        " join fetch o.delivery d" +
                        " join fetch o.orderItems oi" +
                        " join fetch oi.item i", Order.class)
                .getResultList();
    }
}
```

Repository with Spring Data JPA

CrudRepository + PagingAndSortingRepository

```java
public interface AccountRepository extends JpaRepository<Account, Long> {

    boolean existsByUsername(String username);

    Optional<Account> findById(Long id);

    Optional<Account> findByUsername(String username);

    Optional<Account> findByUsernameAndPassword(String username, String password);
}
```

Query Methods

JPA Sample code (1 of 4)

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Spring Data JPA abstracts library for Spring Boots.

It automatically manages versions of JPA-related libraries according to the spring boot version.

It is an in-memory relational database.

It is widely used in testing as a characteristic that runs in memory and initializes when the application is restarted.

It can be managed only by project dependence without separate installation.

JPA Sample code (2 of 4)

```java
@Entity
@Table(name="users")
public class User {
    2개 사용 위치
    private Long id;
    2개 사용 위치
    @Column(name="name")
    private String name;
    2개 사용 위치
    private String teamName;
    2개 사용 위치
    private Integer salary;

    @Id
    @GeneratedValue
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
```

Create a User Entity that is a domain object.

Define and use columns in the table.

If the variable of the user object
and the column name of the table are different,
create the column name of the table with column annotation.

@ GeneratedValue strategy

- IDENTITY, SEQUENCE, TABLE

JPA Sample code (3 of 4)

Defining the JpaRepository Interface

```
@Component
public interface UserJpaRepository extends JpaRepository<User, Long> {
    2개 사용 위치
    User findByName(String name);
}
```

select * from users where name = ‘name’;’

JPA Sample code (4 of 4)

```java
@RestController
@RequestMapping("/users")
public class UsersController {

    4개 사용 위치
    @Autowired
    private UserJpaRepository userJpaRepository;

    /**
     * Used to fetch all the users from DB
     *
     * @return list of {@link User}
     */
    @GetMapping(value = "/all")
    public List<User> findAll() {
        return userJpaRepository.findAll();
    }

    /**
     * Used to find and return a user by name
     *
     * @param name refers to the name of the user
     * @return {@link User} object
     */
    @GetMapping(value = "/{name}")
    public User findByName(@PathVariable final String name){
        return userJpaRepository.findByName(name);
    }

    /**
     * Used to create a User in the DB
     *
     * @param users refers to the User needs to be saved
     * @return the {@link User} created
     */
    @PostMapping(value = "/load")
    public User load(@RequestBody final User users) {
        userJpaRepository.save(users);
        return userJpaRepository.findByName(users.getName());
```

Store and query simple data
by using Repository

Query the entire data in the User table

Query data by name in the User table

Saving User Information in the User table

Effects of JPA

1. Reduces the use of repeated simple queries.

- Just variable values can be entered and stored in entity since CRUD is defined by method.

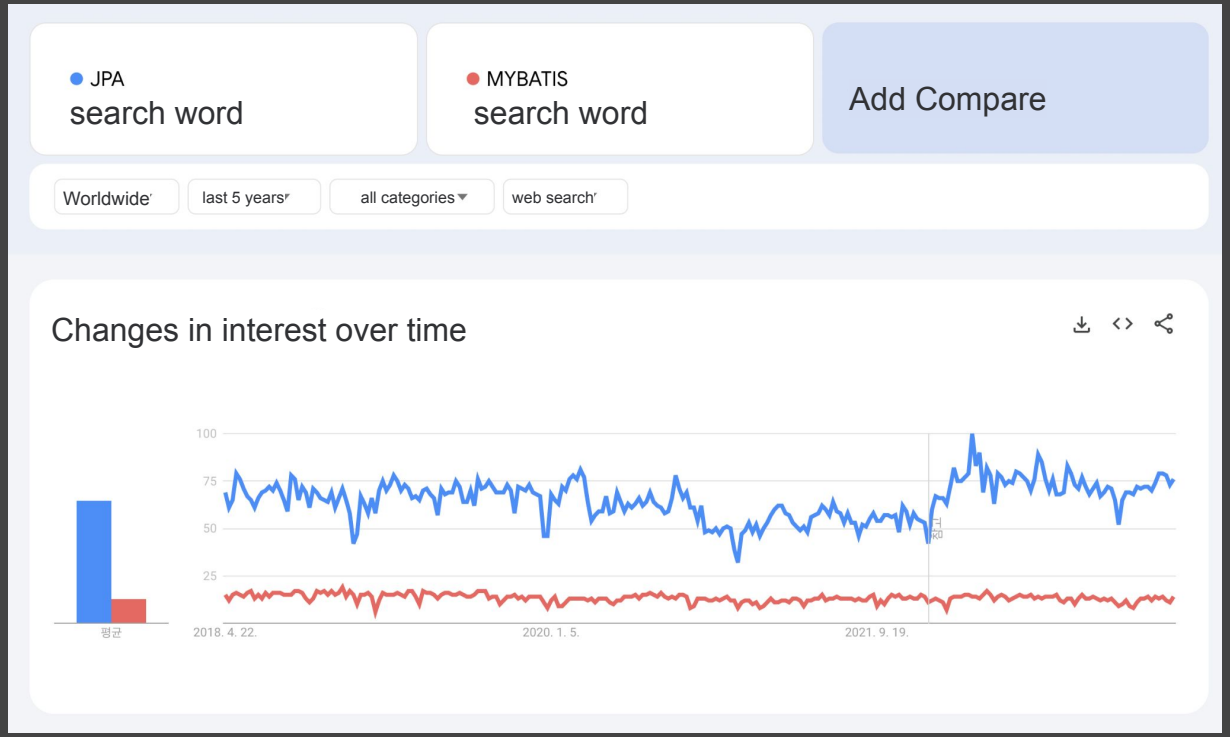
2. Focus on development of business logic

- Enables to focus on business logic as it doesn't use simple and repetitive queries.

3. Knowledge sharing at the same level

- By using Query method, it enables uniform quality code creation without individual differences.

Global Trends



THANK YOU

감사합니다