SMT

Spring Boot 관련 세미나

(with. H.T 기술 공유)

지선학

목차

1.1 Spring Framework 정의

The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform.

A key element of Spring is infrastructural support at the application level: Spring focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments.

Support Policy and Migration

For information about minimum requirements, guidance on upgrading from earlier versions and support policies, please check out the official Spring Framework wiki page

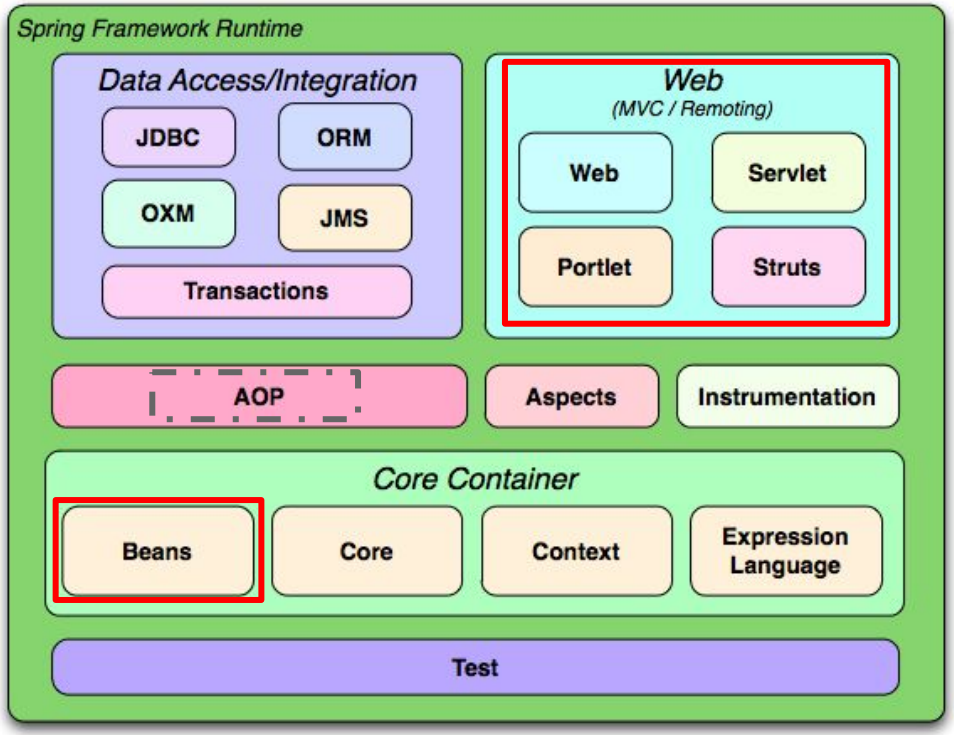
Features

- Core technologies: dependency injection, events, resources, i18n, validation, data binding, type conversion, SpEL, AOP.
- Testing: mock objects, TestContext framework, Spring MVC Test, `WebTestClient`.
- Data Access: transactions, DAO support, JDBC, ORM, Marshalling XML.
- Spring MVC and Spring WebFlux web frameworks.
- Integration: remoting, JMS, JCA, JMX, email, tasks, scheduling, cache.
- Languages: Kotlin, Groovy, dynamic languages.

1) 자바 플랫폼을 위한 오픈소스 어플리케이션 프레임워크

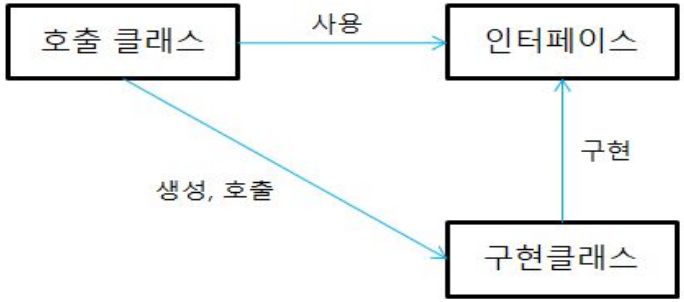
2) 객체의 생성 및 소멸 그리고 라이프 사이클을 관리하며 언제든 Spring 컨테이너로 부터 필요한 객체를 가져와 사용

1.1 Spring Framework 정의



1.2 IoC(Inversion of Control) 개념

- 메소드나 객체의 호출작업을 개발자가 결정하는 것 아니라, 외부에서 결정되는 것을 의미
- 개발자가 Framework에 필요한 부품을 개발하고 조립하는 방식



<인터페이스 호출 방식>

-> 구현클래스 교체가 용이하지만 구현클래스 교체 시 호출클래스의 수정도 필요함

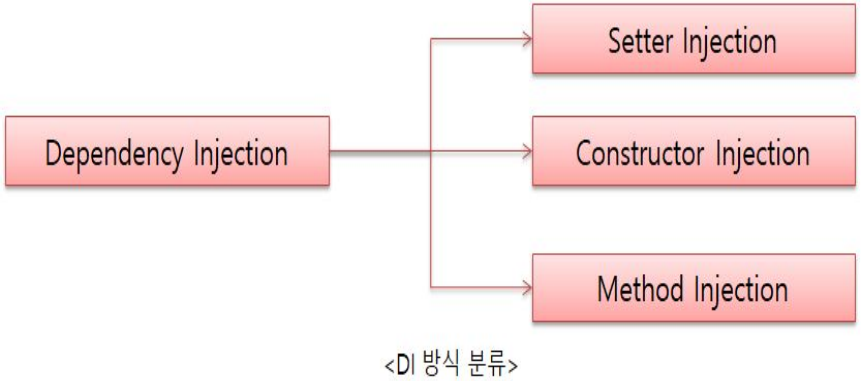
<IoC 호출 방식>

-> 조립기가 구현클래스를 생성하므로 호출 클래스에는 영향을 미치지 않음

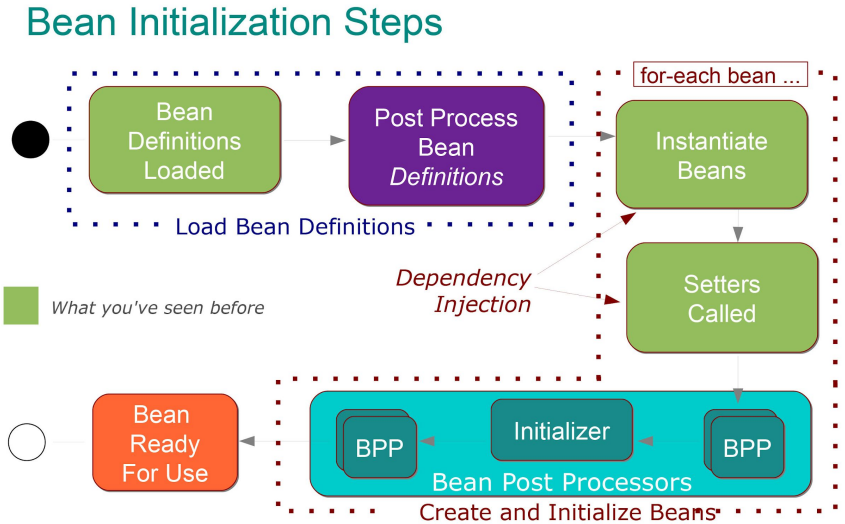
-> 즉, 어떠한 것에도 의존하지 않은 상태로 클래스 간의 관계 형성

1.3 DI(Dependency Injection)정의 및 방식

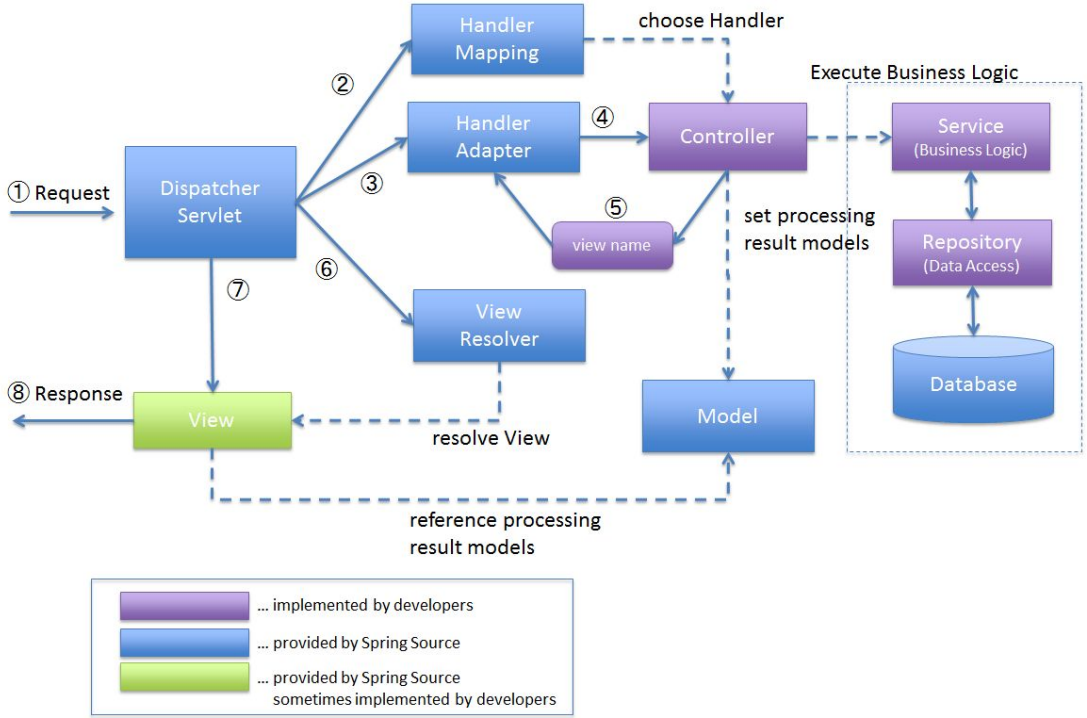
- 각 클래스간의 의존 관계를 빈 설정 정보를 바탕으로 컨테이너(spring)가 자동으로 연결
- Spring DI 컨테이너가 관리하는 객체를 Bean이라고 함
- DI의 가장 큰 장점은 컴포넌트 간의 결합도가 제거 된다는 점임



<DI 방식 분류>



1.4 Spring MVC(Model-View-Control)



2.1 Spring Boot 정의

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need minimal Spring configuration.

If you're looking for information about a specific version, or instructions about how to upgrade from an earlier release, check out the project release notes section on our wiki.

Features

- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- Provide opinionated 'starter' dependencies to simplify your build configuration
- Automatically configure Spring and 3rd party libraries whenever possible
- Provide production-ready features such as metrics, health checks, and externalized configuration
- Absolutely no code generation and no requirement for XML configuration

1) 스프링을 쉽게 사용할 수 있도록 필요한 설정을 대부분 미리 세팅 해놓은 프레임워크

2) 프로젝트 생성시 원하는 third-party 라이브러리를 선택하여 최소한의 설정으로 프로젝트를 생성 가능

2.2 Spring Boot vs Spring Framework

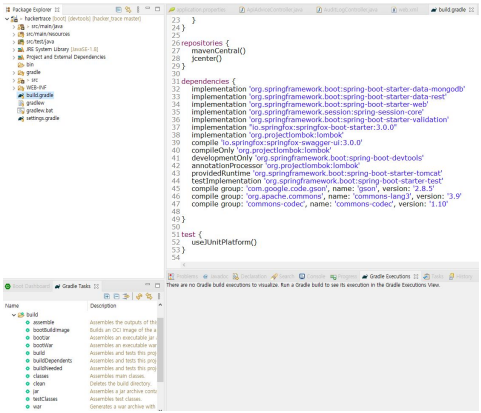
1) Spring Framework 보다 간단해진 설정
2) 내장 된 WAS(tomcat)
3) 기본 설정된 옵션들(.properties)
4) 간단한 배포와 실행(.jar)

3.1 Gradle

1) Gradle은 그루비를 기반으로 한 빌드 도구임

2) Ant와 Maven과 같은 이전 세대 빌드 도구의 단점(xml설정, 속도)을 보완하고 장점을 취합하여 만든 빌드도구

<pom.xml>



<build.gradle>

3.2 Lombok

Project Lombok

- Java 라이브러리에서 반복되는 getter, setter, toString 등의 메서드 작성 코드를 줄여주는 코드 다이어트 라이브러리

```
package com.jsh.portfolio.dto;

import java.sql.Date;

public class Board {
	private long bId;
	private String bUserName;
	private String bTitle;
	private String bContent;
	private int bHit;
	private Date bRegDate;


	public Date getbRegDate() {
		return bRegDate;
	}

	public void setbRegDate(Date bRegDate) {
		this.bRegDate = bRegDate;
	}

	public int getbHit() {
		return bHit;
	}

	public void setbHit(int bHit) {
		this.bHit = bHit;
	}

	public long getbId() {
		return bId;
	}
}
```

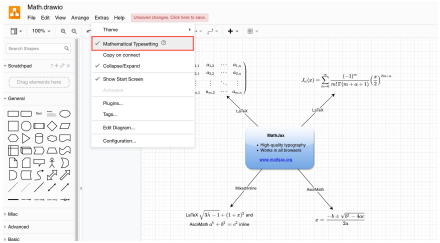
<lombok 미적용 vo class>

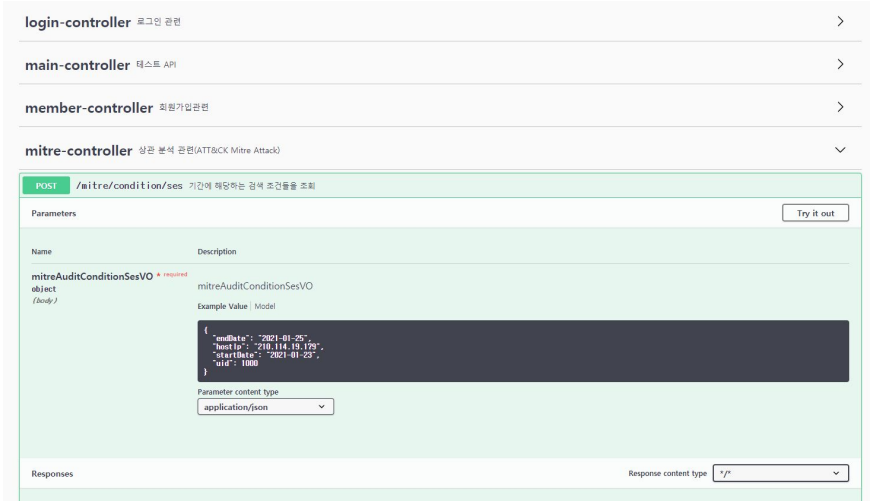
```
@Getter
@Setter
public class MemberVO  {

    @NotNull
    @NotBlank
    @ApiModelProperty(value="사용자 아이디", example = "test1234")
    private String userId;

    @NotNull
    @NotBlank
    @Pattern(
      regexp="^(?=.*[A-Za-z])(?=.*\\d)(?=.*[$@$!%*#?&])[A-Za-z\\d$@$!%*#?&]{8,}$",
      message = "비밀번호는 최소 8자리 이상, 숫자, 문자, 특수문자 각각 1개 이상 포함해야합니다.")
    @ApiModelProperty(value="사용자 패스워드", example = "test1234")
    private String password;

}
```

<lombok 적용 vo class>

3.3 Swagger

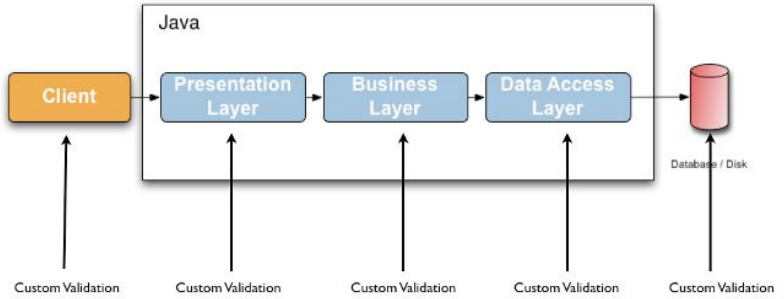
- 개발자가 REST API 서비스를 설계, 빌드, 문서화, 테스팅 할 수 있도록 하는 프로젝트
- API를 통해 Parameter, 응답 정보, 예제 등 Spec 정보 전달이 용이



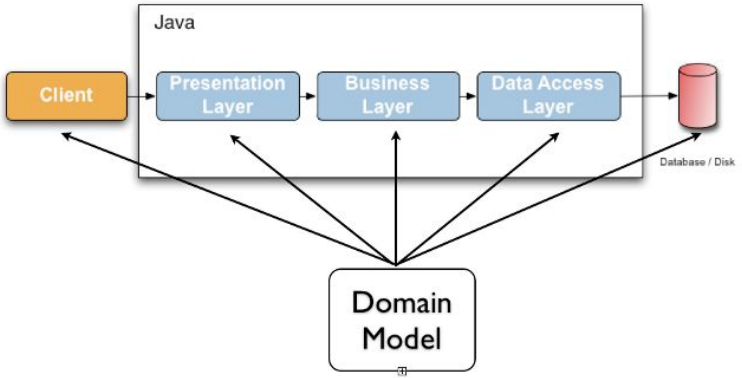


3.4 javax.validation

```
@Getter
@Setter
public class MemberVO  {

    @NotNull
    @NotBlank
    @ApiModelProperty(value="사용자 아이디", example = "test1234")
    private String userId;

    @NotNull
    @NotBlank
    @Pattern(
      regexp="^(?=.*[A-Za-z])(?=.*\\d)(?=.*[$@$!%*#?&])[A-Za-z\\d$@$!%*#?&]{8,}$",
      message = "비밀번호는 최소 8자리 이상, 숫자, 문자, 특수문자 각각 1개 이상 포함해야합니다.")
    @ApiModelProperty(value="사용자 패스워드", example = "test1234")
    private String password;

}
```



출처 Hibernate Validator 6.0.11.Final — JSR 380 Reference Implementation: Reference Guide



출처 Hibernate Validator 6.0.11.Final — JSR 380 Reference Implementation: Reference Guide

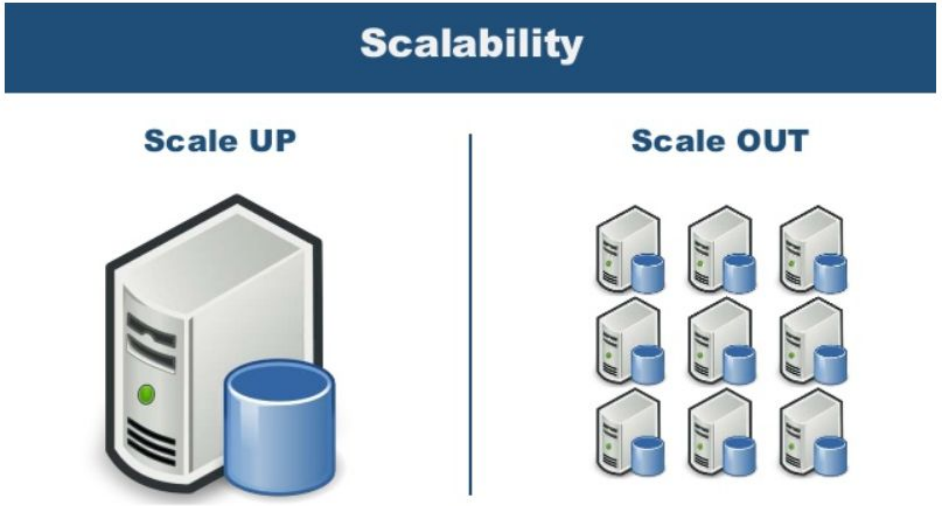
4.1 MongoDB란?

NoSQL은 RDBMS와는 달리 데이터 간의 관계를 정의하지 않음

RDBMS에 비해 훨씬 더 대용량의 데이터를 저장

분산형 구조

고정되지 않은 테이블 스키마



4.1 MongoDB란?

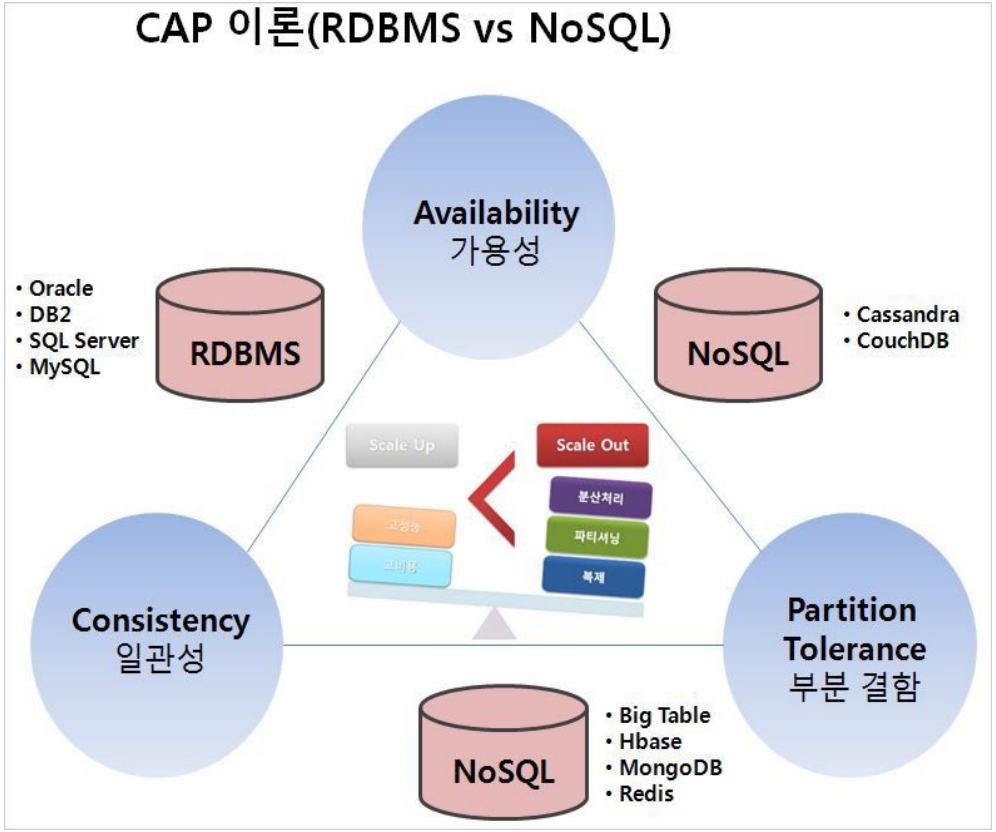
CAP 이론

Consistency
모든 노드가 동일한 데이터를 가짐

Availability
노드가 멈춰도 사용할 수 있음

Partition Tolerance
물리적 분산 환경에서 동작 가능

*모든 DBMS는 두 가지 특성만을 가진다.
하지만, 클러스터링 전략 별로 다양하게 변화함



4.1 MongoDB란?

공식 홈페이지 : www.mongodb.org

역사 : 10gen(현 MongoDB사)에서 개발

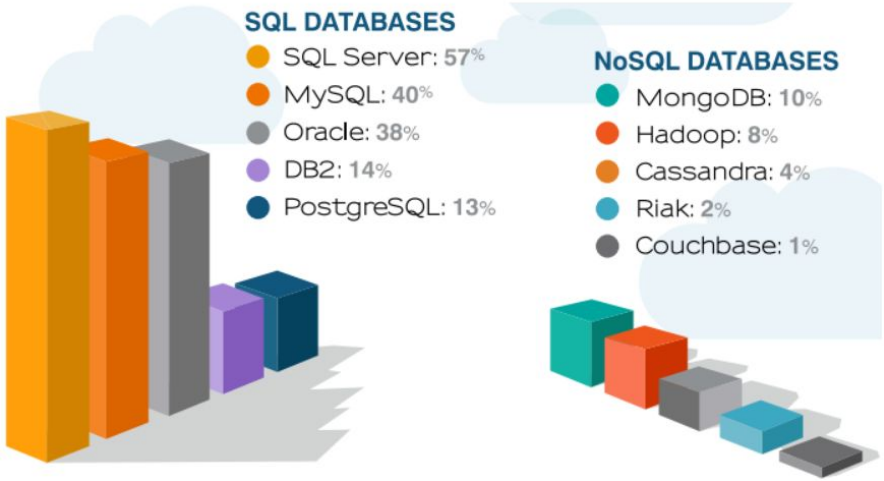
기술 및 언어 : C++로 구현

접근 방식 : 자바스크립트 명령행 인터페이스, C, C#, C++, 얼랭, 하스켈, 자바 등 여러 언어용 드라이버가 존재

쿼리 언어 : SQL과 유사한 쿼리 언어

오픈소스 라이선스 : SSPL(Server Side Public License)

사용하는 곳 : 포스퀘어, 셔터플라이, 인튜이트, 깃허브 등



4.1 MongoDB란?

1) Document-Oriented Storage: 모든 데이터가 JSON 형태로 저장되며 스키마가 없음

2) Full Index Support: RDBMS에 뒤지지 않는 다양한 인덱싱 제공

3) Replication & High Availability: 데이터 복제를 통한 가용성 향상

4) Auto-Sharing: Primary key를 기반으로 여러 서버에 데이터를 나누는 scale-out이 가능

5) Querying: Key 기반의 get, put뿐만 아니라 다양한 종류의 쿼리 제공

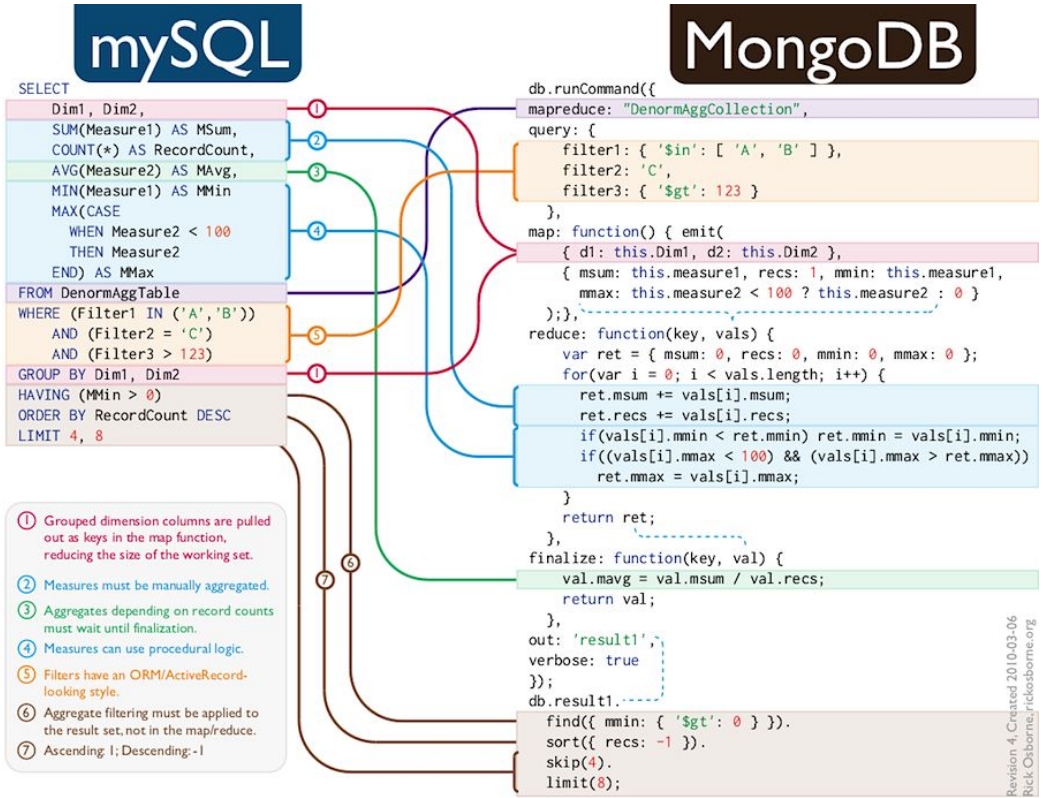
6) Fast In-Place Updates: 고성능 atomic operation 지원

7) MapReduce: 맵리듀스 지원

4.1 MongoDB란?

RDBMS	몽고디비
Database	Database
Table	Collection
Tuple/Row	Document
Column	Key/Field
Table Join	Embedded Documents
Primary Key	Primary Key(_id)
mysqld	mongod
mysql	mongo

4.2 기본쿼리


mySQL
SELECT
Dim1, Dim2,
SUM(Measure1) AS MSum,
COUNT(*) AS RecordCount,
AVG(Measure2) AS MAvg,
MIN(Measure1) AS MMin
MAX(CASE
WHEN Measure2 < 100
THEN Measure2
END) AS MMax
FROM DenormAggTable
WHERE (Filter1 IN ('A','B'))
AND (Filter2 = 'C')
AND (Filter3 > 123)
GROUP BY Dim1, Dim2
HAVING (MMin > 0)
ORDER BY RecordCount DESC
LIMIT 4, 8
MongoDB
db.runCommand({
mapreduce: "DenormAggCollection",
query: {
filter1: { '$in': ['A', 'B'] },
filter2: 'C',
filter3: { '$gt': 123 }
},
map: function() { emit(
{ d1: this.Dim1, d2: this.Dim2 },
{ msum: this.measure1, recs: 1, mmin: this.measure1,
mmax: this.measure2 < 100 ? this.measure2 : 0 }
);},
reduce: function(key, vals) {
var ret = { msum: 0, recs: 0, mmin: 0, mmax: 0 };
for(var i = 0; i < vals.length; i++) {
ret.msum += vals[i].msum;
ret.recs += vals[i].recs;
if(vals[i].mmin < ret.mmin) ret.mmin = vals[i].mmin;
if((vals[i].mmax < 100) && (vals[i].mmax > ret.mmax))
ret.mmax = vals[i].mmax;
}
return ret;
},
finalize: function(key, val) {
val.mavg = val.msum / val.recs;
return val;
},
out: 'result1',
verbose: true
});
db.result1.
find({ mmin: { '$gt': 0 } }).
sort({ recs: -1 }).
skip(4).
limit(8);
1 Grouped dimension columns are pulled out as keys in the map function, reducing the size of the working set.
2 Measures must be manually aggregated.
3 Aggregates depending on record counts must wait until finalization.
4 Measures can use procedural logic.
5 Filters have an ORM/ActiveRecord-looking style.
6 Aggregate filtering must be applied to the result set, not in the map/reduce.
7 Ascending: 1; Descending: -1
Revision 4, Created 2010-03-06
Rick Osborne, rickosborne.org

4.3 활용 in H.T(HackerTrace)

Problem : 특정 host에서 발생하는 대량 로그(대략 1억건 Document(ROW))에 대한 조회 및 통계 정보(5초이내)

My Solve : 1) 검색 조건의 특정 필드에 대한 Indexing, 중복 데이터 필드 Indexing

2) Spring Scheduler를 사용한 주기적인 통계 Collection(table) 정보 갱신

5.1 Controller

```
@ResponseBody
@RequestMapping(value = "/reqLogin", method = RequestMethod.POST)
public Object reqLogin(@RequestParam("APP_KEY") String appKey,
                @RequestParam("EMAIL") String email) {

        restClient = new RestClient(url);

        JSONObject sendData = new JSONObject();
        sendData.put("APP_KEY", appKey);
        sendData.put("EMAIL", email);

        String urlAdd = "/reqLogin";
        System.out.println("INPUT("+urlAdd+")==="+sendData.toJSONString());
        String jsonResult = restClient.post(urlAdd, sendData.toJSONString());
        System.out.println("OUTPUT ==="+jsonResult);

        return jsonResult;

}
```

```
@ApiOperation(value = "사용자 로그인 인증")
@CrossOrigin(origins = "*", allowedHeaders = "*")
@RequestMapping(value="/user", method = RequestMethod.POST)
public ResultVO reqUserLogin(@Valid @RequestBody LoginVO loginVO) {

    int checkUserInfo =service.checkUserInfo(loginVO);

    if( checkUserInfo == 0)
        return APIUtil.resResult(checkUserInfo, "사용자 인증이 완료되었습니다.", null);
    else if(checkUserInfo == -5)
        return APIUtil.resResult(checkUserInfo, "알 수 없는 사용자입니다.", null);
    else
        return APIUtil.resResult(checkUserInfo, "사용자 인증에 실패되었습니다.", null);

}
```

```
@RestControllerAdvice
public class ApiAdviceController {

    @ExceptionHandler(MethodArgumentNotValidException.class)
    public ResultVO handleValidationExceptions(MethodArgumentNotValidException ex) {
        Map<String, String> errors = new HashMap<>();
        ex.getBindingResult().getAllErrors().forEach(c -> errors.put(((FieldError) c).getField(), c.getDefaultMessage()));

        ResultVO vo = new ResultVO();
        vo.setReturn_code(-1);
        vo.setMsg(String.valueOf(errors));
        vo.setData("");

        return vo;
    }
```

5.2 VO 중심 개발

(1) (2)

```
@Getter
@Setter
@ToString
public class AuditLogListVO extends SearchListVO {

    @NotNull
    @Pattern(
(3)         regexp = "^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])$",
            message = "올바른 IP형식이 아닙니다.")
    @ApiModelProperty(value="호스트 아이피", example = "127.0.0.1")
    private String hostIp;

(4) @ApiModelProperty(value="공격 단계", example = "initial-access")
    private String phases;

}
```

1) lombok를 이용한 @Getter, @Setter, @ToString 선언
2) Paging, SearchList VO 클래스 상속을 공통 필드 처리
3) javax.validation을 이용한 필드에 대한 validation 처리
4) Swagger를 통한 파라미터 설명 및 예제 처리

5.3 @RestControllerAdvice

- 전역에서 발생할 수 있는 예외를 잡아 처리해주는 annotation

```
@RestControllerAdvice
public class ApiAdviceController {

(1) @ExceptionHandler(MethodArgumentNotValidException.class)
    public ResultVO handleValidationExceptions(MethodArgumentNotValidException ex) {
        Map<String, String> errors = new HashMap<>();
        ex.getBindingResult().getAllErrors().forEach(c -> errors.put(((FieldError) c).getField(), c.getDefaultMessage()));

        ResultVO vo = new ResultVO();
(2)     vo.setReturn_code(-1);
        vo.setMsg(String.valueOf(errors));
        vo.setData("");

        return vo;
    }
```

1) javax.validation에 의해 발생한 parameter 검증 에러에 대한 처리
2) validation 에러가 발생할 경우 -1과 validation에서 제공하는 error 메시지 return

5.4 No Query DAO

```xml
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.jsh.portfolio.mapper.AdminMainMapper">

	<resultMap id="accessIdResult" type="com.jsh.portfolio.dto.Log">
		<result property="logUserName" column="LogUserName" />
	</resultMap>

	<select id="accessIdSearch" resultMap="accessIdResult">
		SELECT DISTINCT LogUserName
		FROM log order by LogRegDate ASC LIMIT 5
	</select>

	<resultMap id="latestBoardResult" type="com.jsh.portfolio.dto.Board">
		<result property="bId" column="bId" />
		<result property="bTitle" column="bTitle" />
	</resultMap>

	<select id="latestBoard" resultMap="latestBoardResult">
		SELECT
			bId
		       ,bTitle
		FROM userBoard
		ORDER BY bRegDate ASC LIMIT 5
	</select>
```

<mybatis 사용예>

```java
@Repository
public class LoginDAO {

    @Autowired
    private MongoTemplate mongoTemplate;

    public Document findUserInfoByUserId(LoginVO vo) {

        MongoCollection<Document> memCol = mongoTemplate.getCollection("MEMBER");

        BasicDBObject findQuery = new BasicDBObject("userId", vo.getUserId());

        System.out.println(findQuery.toJson());
        List<Document> docList = memCol.find(findQuery).into(new ArrayList<>());

        return docList.size()<1?null:docList.get(0);
    }

}
```

(1) (2) (3) (4)

1) DB 접속 초기화
2) Collection 정보 초기화
3) Where 쿼리 초기화
4) select와 동시에 arrayList로 변환

6. 향후 계획

1. 로그 생성 모듈 필요(log4j 등)
2. CommonDAO 생성을 통한 자주 사용하는 쿼리 로직 모듈화
3. DB 접속 정보, 스케줄링 주기 등 외부 설정 파일로 빼기

확장 가능한 공통 모듈 개발