**MongoDb, Spring 연동**

# 목차

[1. 배경지식 2](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936755)

[1) MongoDb설정 2](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936756)

[2) user 설정 2](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936757)

[2. MongoTest프로젝트 4](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936760)

[1) 프로젝트 생성 4](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936761)

[2) pom.xml 6](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936762)

[3) application.properties 7](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936763)

[3. Java 클래스 구현 8](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936764)

[1) domain 클래스 8](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936765)

[2) Repository 클래스 10](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936766)

[3) Service 클래스 10](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936766)

[4) 컨트롤러 구현 11](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936767)

[5) MongoConfig 10](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936766)

[6) Example01Application 11](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936767)

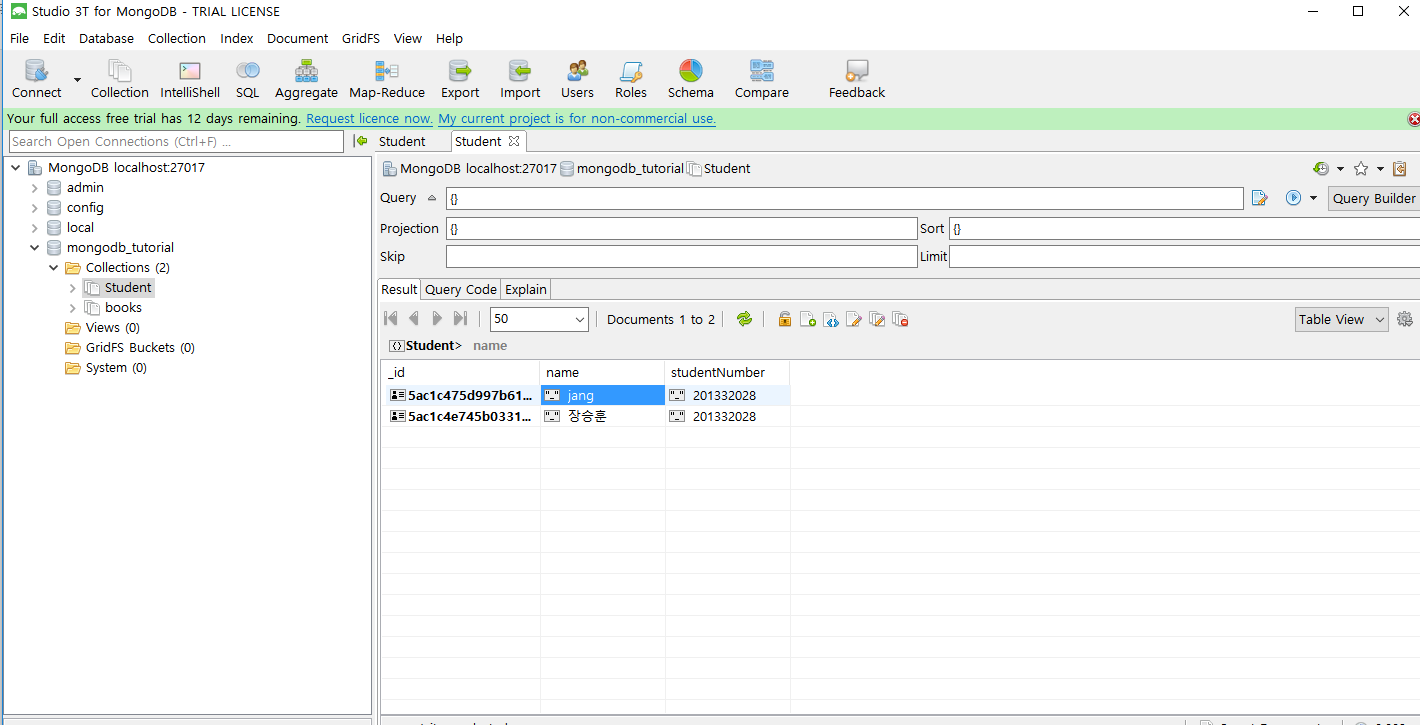
[4. 뷰 구현 12](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936768)

[1) student/list.jsp 12](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936769)

[2) index.jsp 13](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936770)

[3) 실행 13](file:///C:\Users\wkdtn\Downloads\01%20Spring%20boot%20and%20JPA%20(1).docx#_Toc483936771)

1. MongoDb 설정

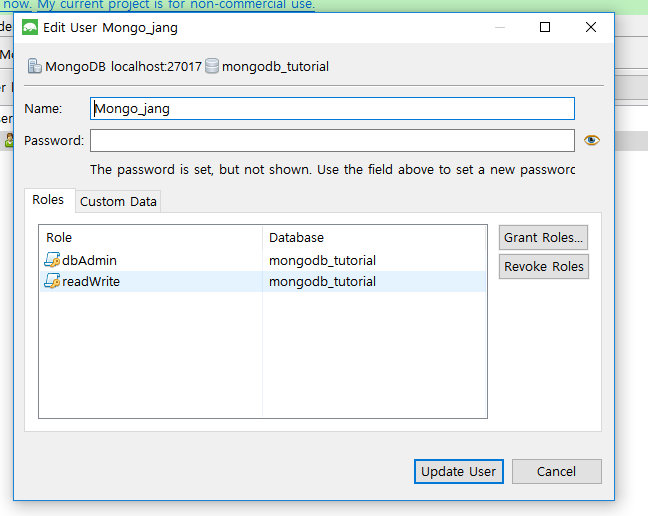


지금 이 프로젝트에서 사용한 디비 설정이다.

Mongodb\_tutorial을 db로 사용하고 있고 student collection을 뷰에서 보여주는 프로젝트를 만들 것이다.

Document는 {“name”:“jang”,”studentNumber”:”201332028”},{“name”:”장승훈”:”studentNumber”:”201332028” } 이 저장되어있다.

1. User 설정



Mongodb\_tutorial에 Mongo\_jang이라는 user를 생성해 놓았다 . 밑에 Role은 dbAdmin ,readWrite가 가능하도록 설정해놨다. 만약에 밑에 디비에 admin계정으로 MongoDb 연동을 진행하면

서버 실행 중 Caused by : 권한 설정 에러가 발생한다. Admin 계정으로 하고 싶다면 poperties에

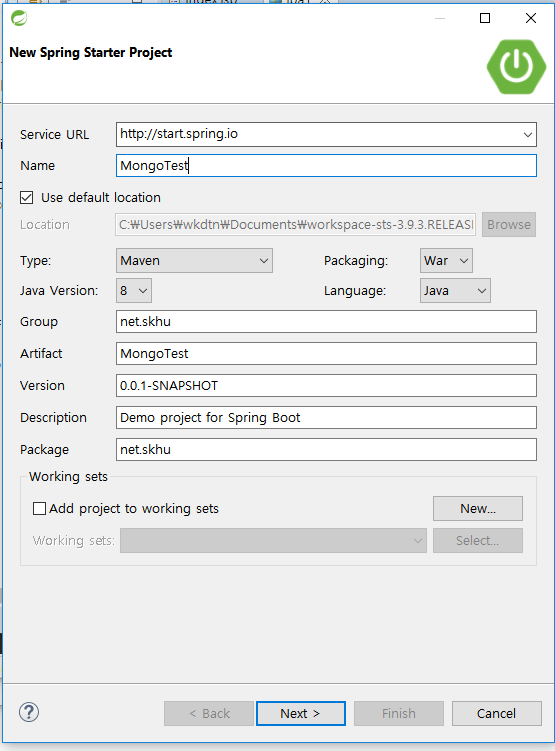
spring.data.mongodb.authentication-database=admin이란 문장을 첨가하면 된다.

User 까지 설정이 완료 되었으면 프로젝트를 생성 해보겠습니다.

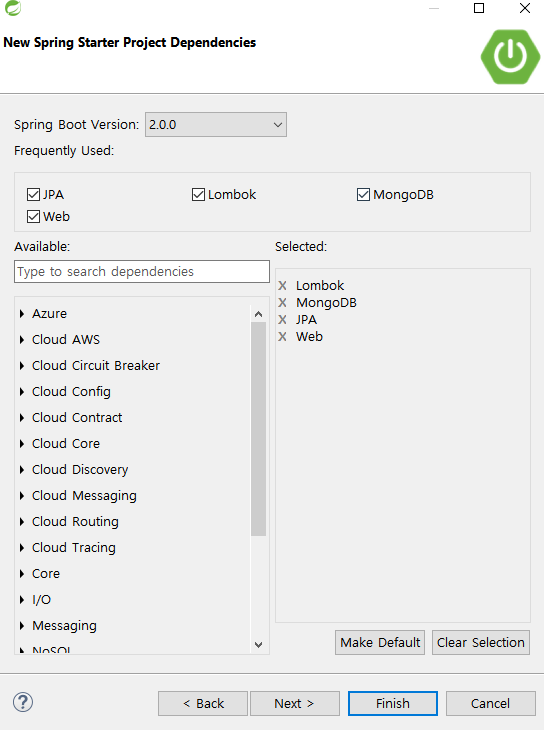
# 2. Mongo 프로젝트

## 프로젝트 생성

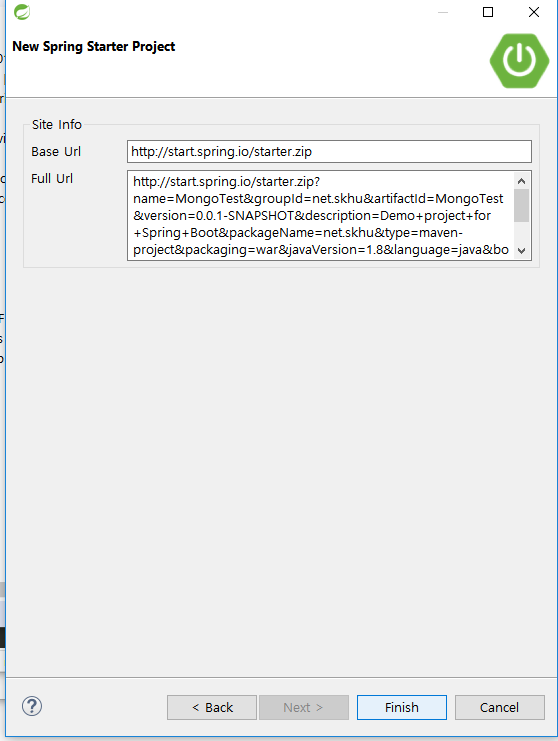
메뉴: File - New - Spring Starter Project



|  |  |
| --- | --- |
| Name | MongoTest |
| Packaing | War |
| Group | net.skhu |
| Artifact | MongoTest |
| Package | net.skhu |



Main Package : net.kang / Dependency : Lombok, MongoDB, Web, JPA로 설정하도록 한다.



Finish 클릭

## 2) pom.xml

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67 | <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>net.skhu</groupId>  <artifactId>MongoTest\_1</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>war</packaging>  <name>MongoTest\_1</name>  <description>Demo project for Spring Boot</description>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.0.RELEASE</version>  <relativePath/> <!-- lookup parent from repository -->  </parent>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  <java.version>1.8</java.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-jpa</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-mongodb</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.projectlombok</groupId>  <artifactId>lombok</artifactId>  <optional>true</optional>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-tomcat</artifactId>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.mongodb</groupId>  <artifactId>mongo-java-driver</artifactId>  <version>3.6.3</version>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |
|  |  |

노란색 부분 jstl이 추가되었다. MongDb를 선택해서 org.mongodb Maven dependency가 추가 된 것을 알 수 있다. 그리고 lombok을 쓰므로 Lombok 설치가 안되있으면 설치 하길 바란다.

## 3) application.properties

**src/main/resources/application.properties**

|  |
| --- |
| spring.mvc.view.prefix=/WEB-INF/views/  spring.mvc.view.suffix=.jsp  spring.data.mongodb.host=127.0.0.1  spring.data.mongodb.port=27017  spring.data.mongodb.database=mongodb\_tutorial  spring.data.mongodb.username=Mongo\_jang  spring.data.mongodb.password=123 |

위의 두줄은 뷰 파일의 위치를 나타내고 밑에 5줄은 몽고디비 연결 설정이다 데이터베이스 설정과 거의 비슷하다. host, port는 MongoDB에서 현재 실행되고 있는 서버 번호를 입력하면 되는데 대부분 Host는 127.0.0.1로 작동을 하는 경우도 있고, localhost로 작성을 해도 무관하다. 그리고 Port는 27017로 쓰면 된다. MySQL는 3306이었는데 MongoDB에서 클러스터링 작업을 한다면 27017, 27018, … 순으로 클러스터링 Port가 지정이 되기 때문에 주로 쓰는 27017로 설정하면 된다. 현재 Mongo\_jang이라는 User는 한 데이터베이스에 대한 권한을 가진 일반 User로서 admin 데이터베이스를 통해 모든 Database를 총괄하는 관리자 User를 이용해서 접속을 하려면 spring.data.mongodb.authentication-database=admin이란 문장을 첨가하면 된다.

# 3. Java 클래스 구현

# Student 도메인 클래스

**src/main/java/net/skhu/domain/Student.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | package net.skhu.domain;  import org.springframework.data.annotation.Id;  import org.springframework.data.mongodb.core.mapping.Document;  import lombok.Data;  @Data  @Document(collection="Student")  public class Student {  @Id  String id;  String name;  String studentNumber;  } |

여기서 새로 추가된 개념은 @Document 어노테이션이다. 이에 대해서 어떻게 구성되는지에 대해서 알아보도록 하겠다.

@Document 어노테이션 : Document들은 Collection 내부에 있다. 그래서 Collection의 이름을 작성해서 Domain 클래스를 연동하여 MongoDB와의 DTO(Data Transfer Object)의 역할을 해줄 수 있도록 하기 위해 이 어노테이션을 작성한다. Spring Data MySQL에서는 @Entity(table=””)와 같은 개념으로 생각하면 되겠다.

# Repository 클래스

**src/main/java/net/skhu/repository/StudentRepository.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | **package** net.skhu.repository;  **import** org.springframework.data.mongodb.repository.MongoRepository;  **import** net.skhu.domain.Student;  **public** **interface** StudentRepository **extends** MongoRepository<Student, String>{  } |

기본 조회만 실행 할 것이기 떄문에 repository에 따로 추가하지는 않았다. 여기서 다른 점은

Extends 로 MongoRepository를 한다는 점인데

우리가 기존에 쓴 jpaRepository는 RDBMS에 대해 JPA를 이용하기 위해 제공을 하고

MongoRepository Spring Data MongoDB에서 제공을 한다. 물론 RDBMS이든 NoSQL이든 CrudRepository, PagingAndSortingRepository를 사용해도 상관은 없다. 또한 Redis에서는 데이터들에 대해서 정렬을 하는 경우는 크게 없기 때문에 CrudRepository를 대표적으로 활용한다고 한다.

# Service 구현

**src/main/java/net/skhu/service/StudentService.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | **package** net.skhu.service;  **import** java.util.List;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Service;  **import** net.skhu.domain.Student;  **import** net.skhu.repository.StudentRepository;  @Service  **public** **class** StudentService {  @Autowired StudentRepository studentRepository;  **public** List<Student> findAll(){  **return** studentRepository.findAll();  }  } |

조회기능을 사용할 것이기 떄문에 findAll메소드에 studentRepository.findAll() 를 리턴해주도록 설정하자 다른 거는 기존의 jpa와 동일하다

# 컨트롤러 구현

**src/main/java/net/skhu/controller/StudentController.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | **package** net.skhu.controller;  **import** java.util.List;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.ui.Model;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** net.skhu.domain.Student;  **import** net.skhu.service.StudentService;  @Controller  @RequestMapping("student")  **public** **class** StudentController {  @Autowired StudentService studentService;  @RequestMapping("list")  **public** String list(Model model) {  List<Student> list = studentService.findAll();  model.addAttribute("list", list);  **return** "student/list";  }  }} |

헷갈렸던 점은 mongoDb예제를 보면 @RestController를 쓰는 예제들이 많은데 RestController를 쓰면 return에서 뷰를 반환하지 않고 문자열을 반환한다. 위의 컨트롤러에서는 student/list를 반환한다. 이 경우 @ResponseBody가 필요하지 않는다.

그리고 @Controller를 사용하면 mvc 아키텍쳐의 뷰를 반환 할 수 있다.

그 외에는 studentService에서 findAll 메소드를 이용해서 document들을 list에 담아 건내주는 코드들이다.

4). MongoConfig 클래스 생성

**src/main/java/net/skhu/config/MongoConfig.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | **package** net.skhu.config;  **import** java.util.Arrays;  **import** org.springframework.beans.factory.annotation.Value;  **import** org.springframework.context.annotation.Configuration;  **import** org.springframework.data.mongodb.MongoDbFactory;  **import** org.springframework.data.mongodb.config.AbstractMongoConfiguration;  **import** org.springframework.data.mongodb.core.MongoTemplate;  **import** org.springframework.data.mongodb.core.SimpleMongoDbFactory;  **import** org.springframework.data.mongodb.core.convert.DefaultMongoTypeMapper;  **import** org.springframework.data.mongodb.core.convert.MappingMongoConverter;  **import** org.springframework.data.mongodb.core.mapping.MongoMappingContext;  **import** com.mongodb.MongoClient;  **import** com.mongodb.MongoCredential;  **import** com.mongodb.ServerAddress;  @Configuration  **public** **class** MongoConfig **extends** AbstractMongoConfiguration{  @Value("${spring.data.mongodb.host}")  **private** String mongoHost;  @Value("${spring.data.mongodb.port}")  **private** Integer mongoPort;  @Value("${spring.data.mongodb.database}")  **private** String mongoDatabase;  @Value("${spring.data.mongodb.username}")  **private** String userName;  @Value("${spring.data.mongodb.password}")  **private** String password;  @Override  **protected** String getDatabaseName() {  **return** mongoDatabase;  }  @Override  **public** MongoClient mongoClient(){  MongoCredential credential = MongoCredential.*createCredential*(userName, mongoDatabase, password.toCharArray());  **return** **new** ~~MongoClient~~(**new** ServerAddress(mongoHost, mongoPort), Arrays.*asList*(credential));  }  @Override  **public** MongoDbFactory mongoDbFactory() {  **return** **new** SimpleMongoDbFactory(mongoClient(), getDatabaseName());  }  @Override  **public** MongoMappingContext mongoMappingContext() **throws** ClassNotFoundException {  **return** **super**.mongoMappingContext();  }  @Override  **public** MongoTemplate mongoTemplate() **throws** Exception{  MappingMongoConverter converter=**new** ~~MappingMongoConverter~~(mongoDbFactory(), mongoMappingContext());  converter.setTypeMapper(**new** DefaultMongoTypeMapper(**null**));  MongoTemplate mongoTemplate=**new** MongoTemplate(mongoDbFactory(), converter);  **return** mongoTemplate;  }  } |

AbstractMongoConfiguration 정의

AbstractMongoConfiguration은 MongoDB와 Spring을 연동하기 위한 Configuration을 접근하기 쉽도록 하기 위한 기본적인 설정을 파악할 수 있도록 유도하는 Configuration으로 추상 메소드로 인식할 수 있다

@Value 어노테이션 : application.properties에 현존하는 요소들에 대해서 Config 멤버 변수에 기재를 하도록 도와주는 어노테이션으로 볼 수 있다.

MappingMongoConverterconverter: converter.setTypeMapper(newDefaultMongoTypeMapper(null)); 문장을 쓴 이유가 이를 빼놓고 REST API에 Insert, Update 작업을 한다면 MongoDB 서버 자체에서 \_class Field를 통해서 우리가 작성한 Domain 클래스의 패키지 주소를 저장을 하기 때문에 이를 방지하고자 추가를 하였다.

5). Example01Application 클래스 생성

**src/main/java/net/skhu/Example01Application.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | **package** net.skhu;  **import** javax.sql.DataSource;  **import** org.springframework.boot.SpringApplication;  **import** org.springframework.boot.autoconfigure.EnableAutoConfiguration;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  **import** org.springframework.boot.autoconfigure.data.jpa.JpaRepositoriesAutoConfiguration;  **import** org.springframework.boot.autoconfigure.domain.EntityScan;  **import** org.springframework.boot.context.properties.ConfigurationProperties;  **import** org.springframework.boot.jdbc.DataSourceBuilder;  **import** org.springframework.context.annotation.Bean;  **import** org.springframework.context.annotation.ComponentScan;  **import** org.springframework.data.mongodb.repository.config.EnableMongoRepositories;  @EnableAutoConfiguration(exclude = {JpaRepositoriesAutoConfiguration.**class**})  @EnableMongoRepositories(basePackages = "net.skhu.repository")  @EntityScan(basePackages = "net.skhu.domain")  @ComponentScan(basePackages = "net.skhu")  @SpringBootApplication  **public** **class** Example01Application {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(Example01Application.**class**, args);  }  @Bean  @ConfigurationProperties(prefix = "spring.data.db-main")  **public** DataSource mainDataSource() {  **return** DataSourceBuilder.*create*().build();  }  @Bean  @ConfigurationProperties(prefix = "spring.data.db-log")  **public** DataSource contractDataSource() {  **return** DataSourceBuilder.*create*().build();  }  } |

MongoConfig를 통해 설정을 완료했다면 이를 기반으로 작동을 하는데 Example01Application에서 해결을 해야 하는 문제가 바로 AutoConfiguration 문제이다. 우리가 작성한 properties는 spring.datasource를 이용한 것이 아니라 spring.data.mongo를 이용해서 작성을 하였기 때문에 DataSource가 없다는 에러가 뜨기 때문에 이를 해결하기 위해 @EnableAutoConfiguration에서 JpaRepositoriesAutoConfiguration을 빼고(이는 MongoRepository에 문제가 없도록 하기 위해서이다.) 추가로 작성을 하였고 @Bean을 통해 추가한 DataSource를 이용해서 이에 대한 문제를 해결 하도록 추가로 작성을 하였다. 그리고 @EnableMongoRepositories, @EntityScan, @ComponentScan 어노테이션은 각각 Repository, Entity(Document Domain 클래스), Component들에 대해서 패키지 주소를 입력해서 지정해두기 위해 추가한 문장이다.

# 4.뷰 구현

# 1)student/list.jsp

**src/main/webapp/WEB-INF/views/student/list.jsp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"* pageEncoding=*"UTF-8"*%>  <%@ taglib uri=*"http://java.sun.com/jsp/jstl/core"* prefix=*"c"*%>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=utf-8"*>  <meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*>  <link rel=*"stylesheet"* href=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"*>  <script src=*"https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"*></script>  <script src=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"*></script>  </head>  <body>  <div class=*"container"*>  <h1>student</h1>  <table class=*"table table-bordered"*>  <thead>  <tr>  <th>id</th>  <th>이름</th>  <th>학번</th>    </tr>  </thead>  <tbody>  <c:forEach var=*"student"* items=*"*${ list }*"*>  <tr>  <td>${ student.id }</td>  <td>${ student.name }</td>  <td>${ student.studentNumber }</td>    </tr>  </c:forEach>  </tbody>  </table>  </div>  </body>  </html> |

# 2)index.jsp

src/main/webapp/index.jsp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"* pageEncoding=*"UTF-8"*%>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>  <title>mongo</title>  </head>  <body>  <h1>mongo</h1>  <a href=*"student/list"*>student/list</a> <br />  </body>  </html> |

5. 실행 화면

