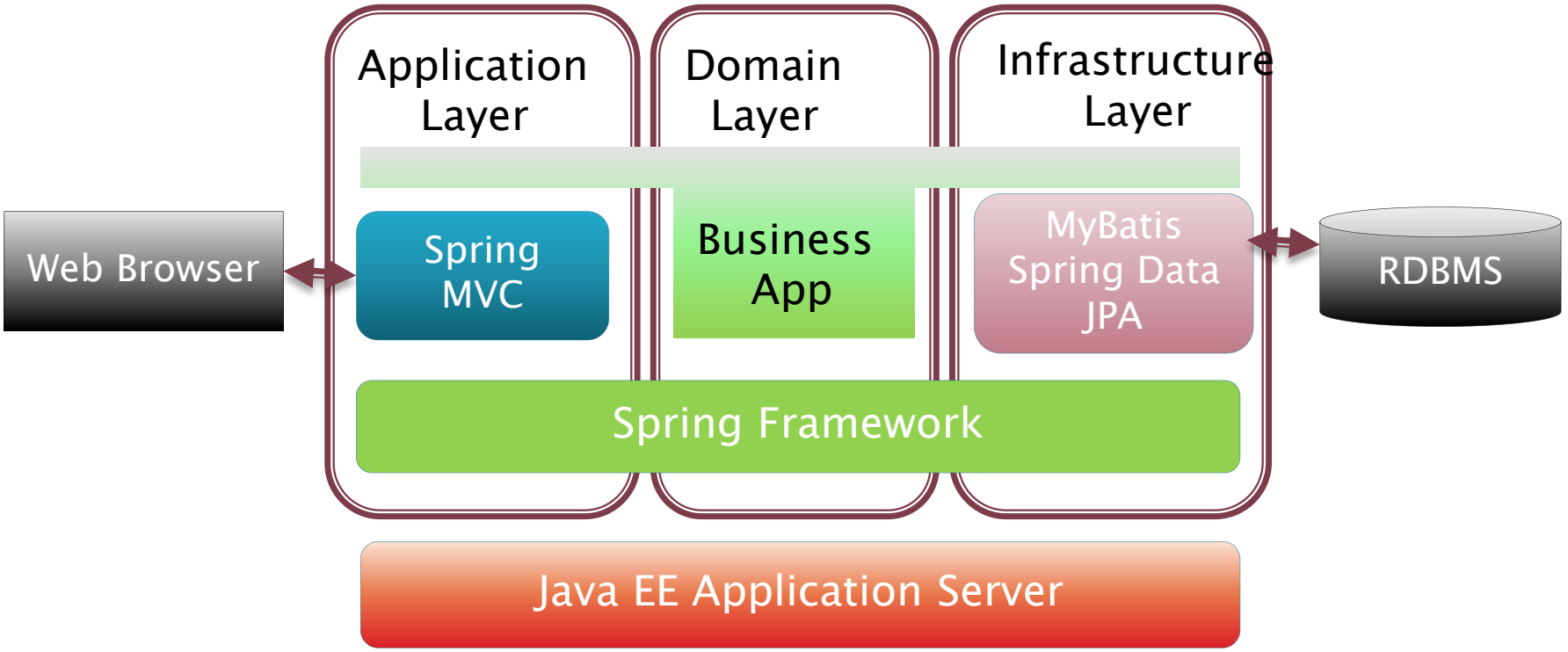
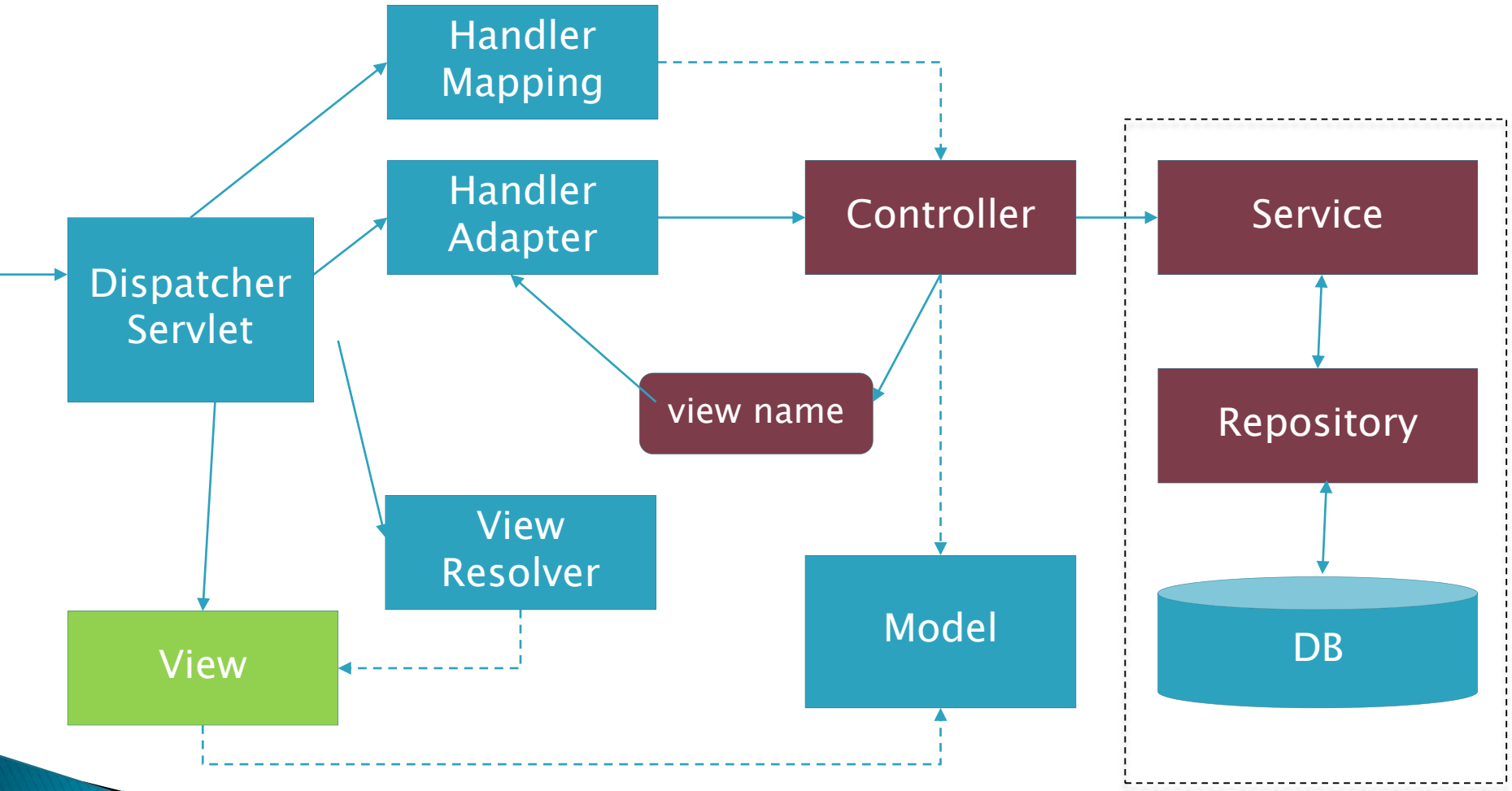
Spring Boot & JPA

oraclejava

Software Framework(Server)



Spring MVC


Handler Mapping
Handler Adapter
Controller
Service
Dispatcher Servlet
view name
Repository
View Resolver
Model
DB
View

개발환경 설정

종류	제품명
OS	Windows 7
JVM	Java 1.8
IDE	Spring Tool Suite 3.6.4.RELEASE
Build Tool	Apache Maven 3.3.9
Application Server	Pivotal tc Server Developer Edition v3.1
Web Browser	Google Chrome

Spring MVC & JPA2 & Hibernate

JPA 환경설정

- LocalContainerEntityManagerFactoryBean
- DataSource 설정
- JpaTransactionManager
- Tx:annotation-driven 설정

LocalContainerEntityManagerFactoryBean

```
<bean id="myEMF"
class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
<property name="dataSource" ref="dataSource" />
<property name="packagesToScan" value="com.oraclejava.todo.domain.model"/>
<property name="jpaVendorAdapter">
<bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"
/>
</property>
<property name="jpaProperties">
<props>
<!-- 개발시: update, 운용시: none, 개발환경 새로 만들때 : create-drop -->
<prop key="hibernate.hbm2ddl.auto">create-drop</prop>
<prop key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</prop>
<prop key="hibernate.show_sql">true</prop>
</props>
</property>
</bean>
```

DataSource

- <!-- DataSource -->
- <bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource">
- <property name="driverClassName" value="com.mysql.jdbc.Driver" />
- <property name="url" value="jdbc:mysql://localhost/todo" />
- <property name="username" value="root" />
- <property name="password" value="123456" />
- </bean>

JpaTransactionManager

- <bean id="transactionManager"
- class="org.springframework.orm.jpa.JpaTransactionManager">
- <property name="entityManagerFactory" ref="myEMF"></property>
- </bean>

tx:annotation-driven 설정

- Servlet-context.xml에 tx:annotation-driven 설정
- <tx:annotation-driven transaction-manager="*transactionManager*"/>

AbstractDao 작성

- EntityManager 작성
- @PersistenceContext 로 의존성 주입

- **public class AbstractDao<PK extends Serializable, T> {**
- **private final Class<T> persitentClass;**

- @SuppressWarnings("unchecked")
- **public AbstractDao() {**
- //제네릭스로 부모클래스가 운용하는 파라메터의 클래스 취득
- **this.persitentClass = (Class<T>)((ParameterizedType)this.getClass().**
- getGenericSuperclass()).getActualTypeArguments()[1];
- }

- @PersistenceContext
- EntityManager entityManger;

- **protected EntityManager getEntityManager() {**
- **return this.entityManger;**
- **}**

- **protected T getByKey(PK key) {**
- **return (T)entityManger.find(persitentClass, key);**
- **}**

```
protected void persist(T entity) {
entityManger.persist(entity);
}

protected void upd(T entity) {
entityManger.merge(entity);
}

protected void del(T entity) {
entityManger.remove(entity);
}
}
```

Dao작성

- EntityManager 사용
- JPQL 사용

```
@Repository
public class TodoRepositoryImpl2 extends AbstractDao<Integer, Todo>
implements TodoRepsitory {

@Override
public Todo findOne(int todoId) {
return getByKey(todoId);
}

@Override
public Collection<Todo> findAll() {
List<Todo> todos =
getEntityManager()
.createQuery("SELECT t FROM Todo t")
.getResultList();
return todos;
}
```

```
@Override
public void create(Todo todo) {
persist(todo);
}

@Override
public boolean update(Todo todo) {
upd(todo);
return true;
}

@Override
public void delete(Todo todo) {
del(todo);
}
```

```
@Override
public long countByFinished(boolean finished) {
long cnt = (Long)getEntityManager().
createNativeQuery("SELECT count(*) FROM Todo where finished = :finished")
.setParameter("finished", finished)
.getSingleResult();
return cnt;
}

}
```

Spring Boot 설치와 사용 (Bootstrapping)

- 1. 설치
- JDK 8 설치 http://iclass.tistory.com/entry/1-Java-SE-Development-Kit-8-설치-v180
- Maven 설치 http://freestrokes.tistory.com/entry/MAVEN-MAVEN-설치-및-Eclipse-연동하기
- STS 설치 http://jwgye.tistory.com/12

STS설치시 주의사항

- 알집(Alzip) 설치
  - Window기본 압축 프로그램이 너무 느려서 알집등으로 압축 해제
- 압축파일 이름 짧게 변경
  - spring-tool-suite-3.8.4.RELEASE-e4.6.3-win32-x86_64.zip 등으로 파일이름이 너무 길어서 이대로 압축 해제시 window 자체오류 발생함
  - Sts.zip 등으로 짧게 변경하고 압축해제

2. 프로젝트 작성

- 명령 프롬프트에서 사용할 워크 스페이스 로 이동
- Ex)cd d:\Dev\workspace
- D:\Dev\workspace2>mvn archetype:generate -DarchetypeArtifactId=maven-archetype-quickstart -DgroupId=org.example -DartifactId=skeleton-web-services -DinteractiveMode=false

- maven-archetype-quickstart : 프로젝트타입(간단한? Java Project 작성)
- org.example: 기본 패키지로 사용할 이름
- skeleton-web-services: 프로젝트명
- -DinteractiveMode=false : 인터렉티브 모드를 사용하지 않음(y/n등을 물어보지 않음)

3. Pom.xml 수정

- a. 필요없는 element 삭제
  - Packaging, name, url (패키징은 기본 jar로)
  - JUnit 삭제(추후 테스팅이 필요시 추가가능)
- b. 추가
  - Parent 엘리먼트

- <parent>
- <groupId>org.springframework.boot</groupId>
- <artifactId>spring-boot-starter-parent</artifactId>
- <version>1.2.1.RELEASE</version>
- </parent>

Dependency 추가

- <dependencies>
- <dependency>
- <groupId>org.springframework.boot</groupId>
- <artifactId>spring-boot-starter-web</artifactId>
- </dependency>
- </dependencies>

build추가

- <build>
- <plugins>
- <plugin>
- <groupId>org.springframework.boot</groupId>
- <artifactId>spring-boot-maven-plugin</artifactId>
- </plugin>
- </plugins>
- </build>

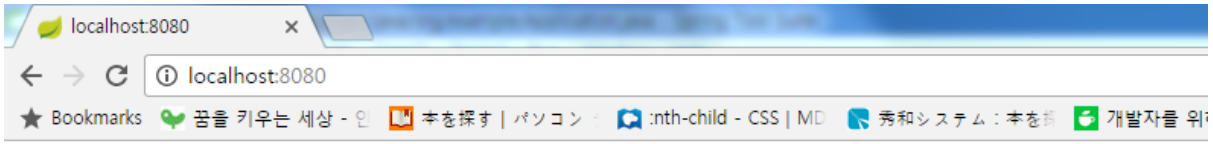
4. Application.java 수정

- package org.example;
- import org.springframework.boot.SpringApplication;
- import org.springframework.boot.autoconfigure.SpringBootApplication;
- @SpringBootApplication
- public class Application
- {
- public static void main(String[] args) throws Exception
- {
- SpringApplication.run(Application.class, args);
- }
- }

- 이후, 프로젝트 우클릭 후, Maven -> Update Project… 실행

5. 프로젝트 실행 및 jar 실행

- 도스 프롬프트(현재 프로젝트의 루트에 있음)에서
- mvn spring-boot:run 엔터
- 8080으로 포트가 잡힌 것을 확인한 후, 브라우저에서 http://localhost:8080 하면



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Wed Dec 14 20:27:04 KST 2016
There was an unexpected error (type=Not Found, status=404).
No message available

포트 변경

- 스프링부트의 기본포트 8080
- 다른 포트를 사용하려면 application.properties 수정
- server.port = 포트번호
- 적용방법
- STS package explorer
- 프로젝트명(boot) - src/main/resources - application.properties 작성

Jar를 실행

- Ctrl + C로 해서 서버를 중단
- 중단된 프롬프트 화면에서
- mvn clean package 엔터
- (현제 프로젝트의 패키지를 다시 클린하게 하는 역할)
- D:\Dev\workspace2\skeleton-web-services>java -jar target\skeleton-web-services-1.0-SNAPSHOT.jar

RESTful Web Service 작성

iPhone 7 128GB

구매하러가기

LG V20

월 27,140 원~

(데이터 스페셜 D 요금제 기준)

P9

월 16,830 원~

(데이터 스페셜 A 요금제 기준)

LG U

월 8,210 원~

(데이터 일반 요금제 기준)

화웨이 H폰

월 2,600 원~

(데이터 일반 요금제 기준)

```
package org.example.ws.model;

public class Phone {
    private int id;    //폰id
    private String name;   //폰이름
    private int price;    //월 얼마?
//이하 getter/setter
```

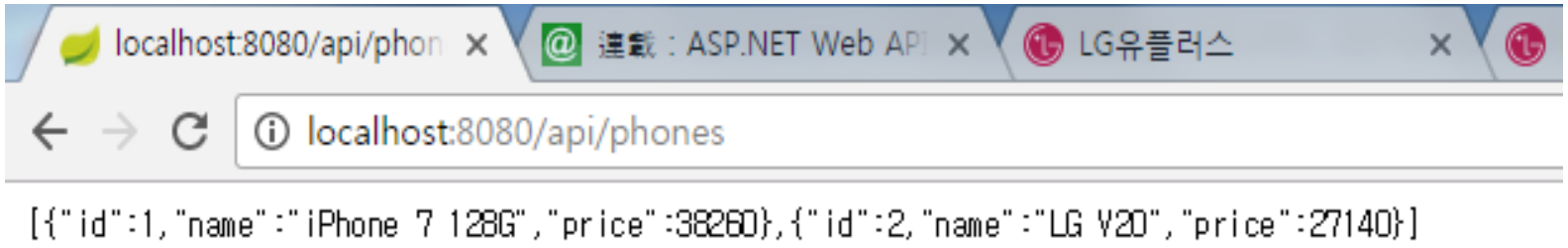
컨트롤러 작성

- @RestController
- **public class** PhoneController {
-
- @RequestMapping(value="api/phones",
-
method=RequestMethod.*GET*,
-
produces=MediaType.*APPLICATION_JSON_VALUE*)
- **public** ResponseEntity<Collection<Phone>> getPhones() {
- ArrayList<Phone> phones = **new** ArrayList<Phone>();

```
Phone p1 = new Phone();
            p1.setId(1);
            p1.setName("iPhone 7 128G");
            p1.setPrice(38260);

            phones.add(p1);

            Phone p2 = new Phone();
            p2.setId(2);
            p2.setName("LG V20");
            p2.setPrice(27140);

            phones.add(p2);
```

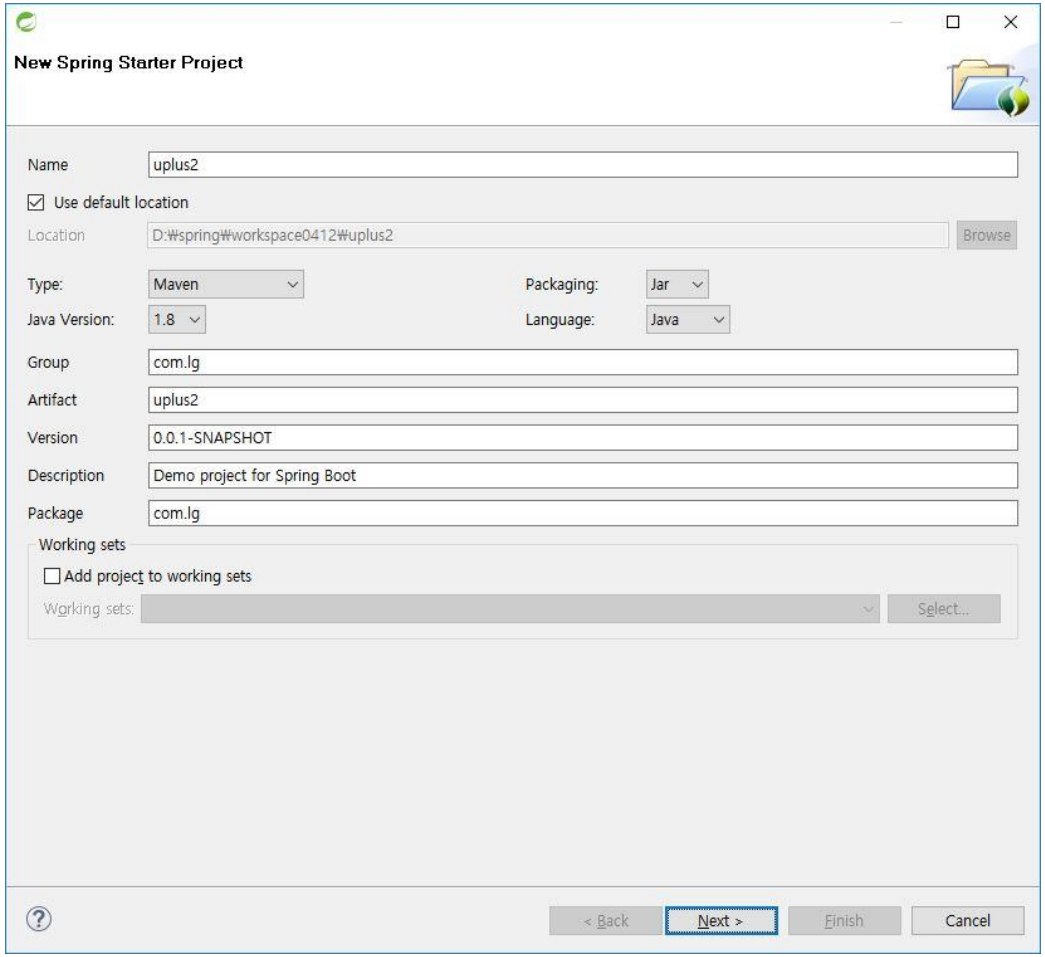
- // 이하 폰 추가
- **return new** ResponseEntity<Collection<Phone>>(phones, HttpStatus.*OK*);
- }
- }
-

브라우저 확인(Json 데이터 표시)

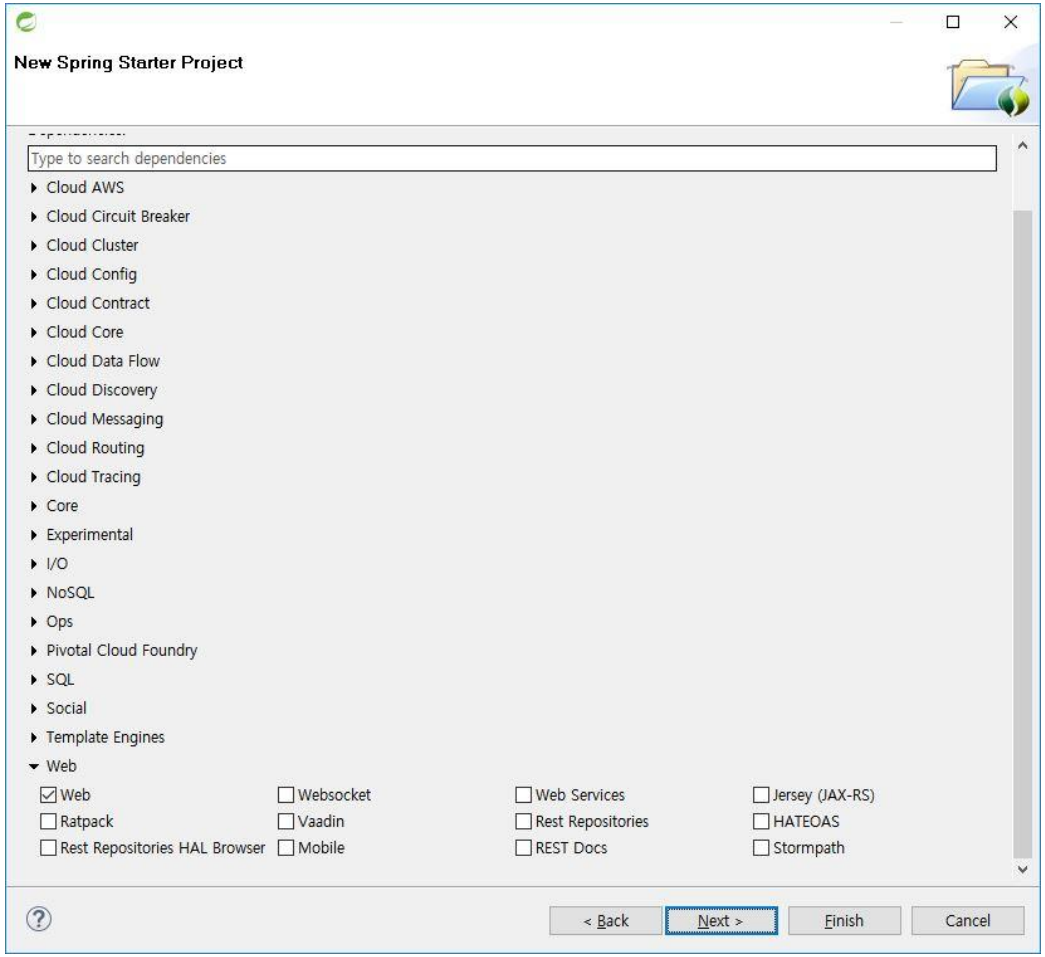


Spring Starter Project 작성

- STS File -> New -> Spring Starter Project

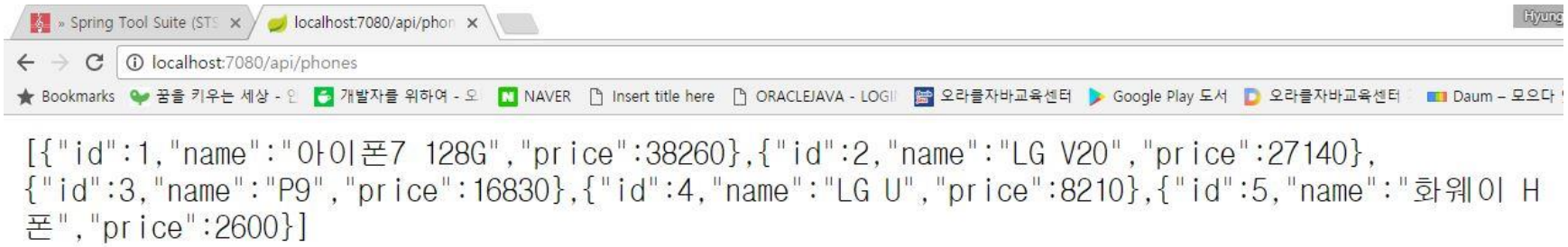

New Spring Starter Project
Name
uplus2
Use default location
Location
D:₩spring₩workspace0412₩uplus2
Browse
Type:
Maven
Packaging:
Jar
Java Version:
1.8
Language:
Java
Group
com.lg
Artifact
uplus2
Version
0.0.1-SNAPSHOT
Description
Demo project for Spring Boot
Package
com.lg
Working sets
Add project to working sets
Working sets:
Select...
< Back
Next >
Finish
Cancel

Next


New Spring Starter Project
Type to search dependencies
Cloud AWS
Cloud Circuit Breaker
Cloud Cluster
Cloud Config
Cloud Contract
Cloud Core
Cloud Data Flow
Cloud Discovery
Cloud Messaging
Cloud Routing
Cloud Tracing
Core
Experimental
I/O
NoSQL
Ops
Pivotal Cloud Foundry
SQL
Social
Template Engines
Web
Web
Websocket
Web Services
Jersey (JAX-RS)
Ratpack
Vaadin
Rest Repositories
HATEOAS
Rest Repositories HAL Browser
Mobile
REST Docs
Stormpath
< Back
Next >
Finish
Cancel

- Web 에 체크 후 Next

화면 캡쳐


» Spring Tool Suite (STS
localhost:7080/api/phon
localhost:7080/api/phones
Bookmarks
꿈을 키우는 세상 - 인
개발자를 위하여 - 오
NAVER
Insert title here
ORACLEJAVA - LOGI
오라클자바교육센터
Google Play 도서
오라클자바교육센터
Daum – 모으다
[{"id":1,"name":"아이폰7 128G","price":38260},{"id":2,"name":"LG V20","price":27140},
{"id":3,"name":"P9","price":16830},{"id":4,"name":"LG U","price":8210},{"id":5,"name":"화웨이 H
폰","price":2600}]

MVC에 근거한 컨트롤러

```
@Controller
public class HelloController {

    @RequestMapping(value="/", method=RequestMethod.GET)
    public ModelAndView index() {
        ModelAndView mav = new ModelAndView();
        mav.setViewName("index");
        mav.addObject("msg", "안녕하세요");
        return mav;
    }
}
```

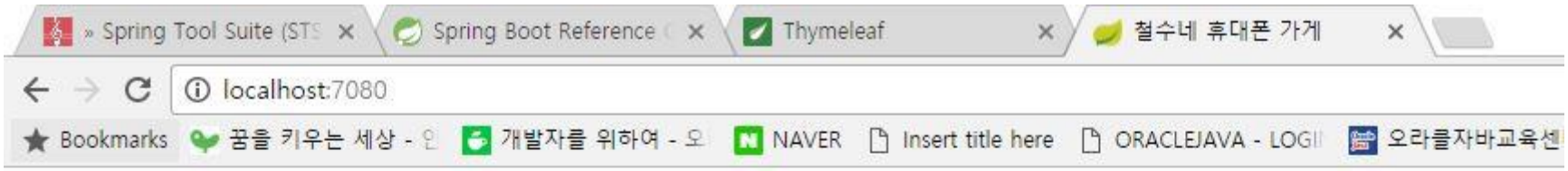
, Thymeleaf 라는 뷰 템플릿엔진을 이용

- *JSPs should be avoided if possible, there are several known limitations when using them with embedded servlet containers.*
-
- http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#boot-features-spring-mvc-template-engines

- 템플릿 위치: src/main/resources templates 폴더

- 소스 코드
- <!DOCTYPE html>
- <html>
- <head>
- <meta charset="UTF-8"/>
- <title>철수네 휴대폰 가게</title>
- </head>
- <body>
- 잠시만 기다려 주세요...
- </body>
- </html>

화면 캡쳐


» Spring Tool Suite (STS
Spring Boot Reference
Thymeleaf
철수네 휴대폰 가게
localhost:7080
Bookmarks
꿈을 키우는 세상 - 인
개발자를 위하여 - 오
NAVER
Insert title here
ORACLEJAVA - LOGI
오라클자바교육센

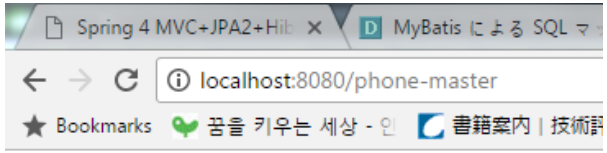
안녕하세요

AngularJS와 연동

- AngularJS란?
  - 구글에서 개발한 JavaScript MVC Framework
- 모듈
  - AngularJS에서 다루는 하나의 단위(App)
  - phoneApp, chatApp, VideoApp
- 컨트롤러
  - 모델(데이터)와 뷰(화면)을 연결하는 하나의 클래스 역할

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8"/>
<title>Insert title here</title>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
<script th:inline="javascript">
//모듈 작성
var app = angular.module('phoneApp', []);
var contextRoot = /*[[@{/}]]*/'';
//컨트롤러 작성
app.controller('phoneCtrl', function($scope, $http){
$http.get(contextRoot + "api/phones")
.success(function(data) {
$scope.phones = data;
});
});
</script>
```

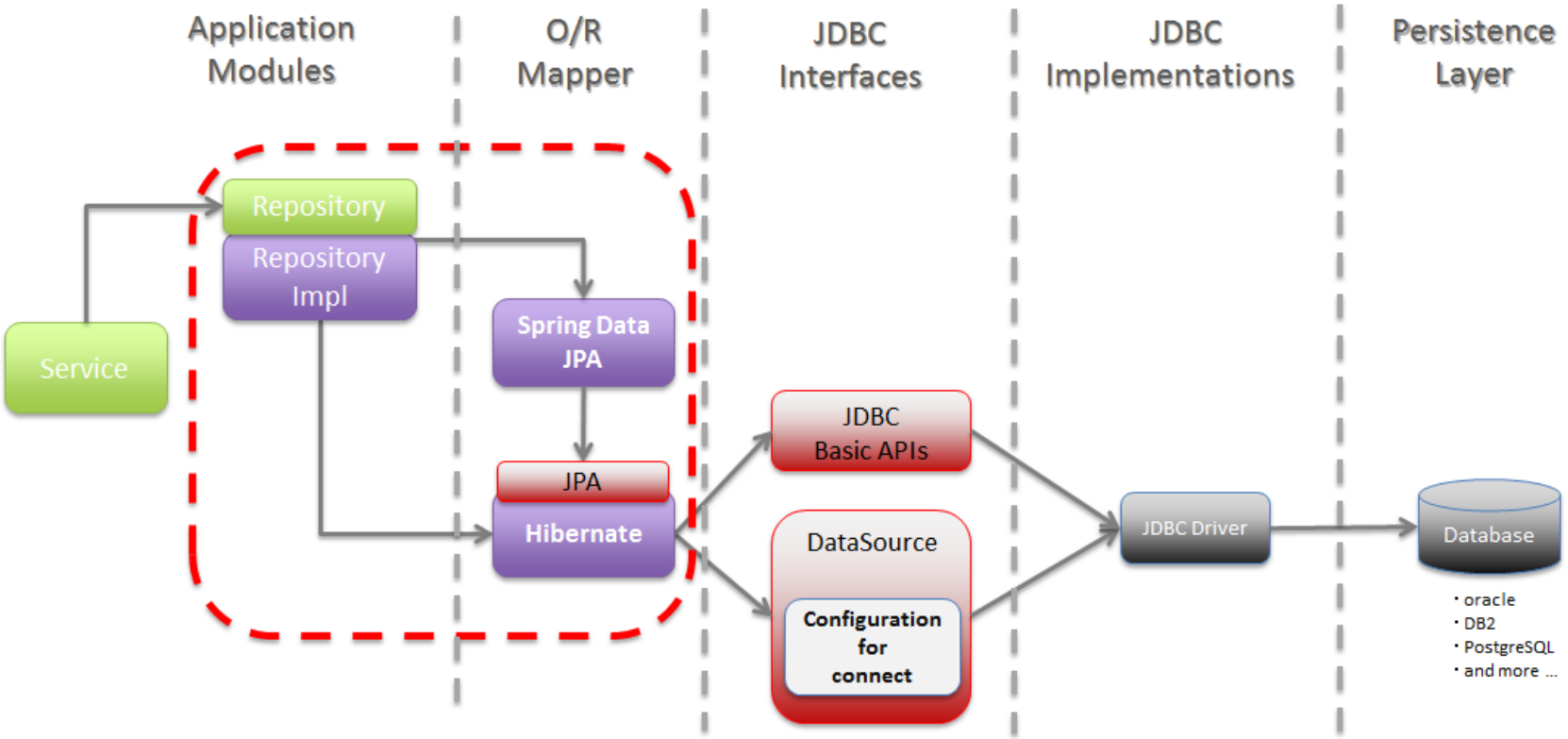
```
</head>
<body>
<div ng-app="phoneApp" ng-controller="phoneCtrl">
<table border="1">
<tr>
<th>id</th><th>name</th><th>price</th>
</tr>
<tr ng-repeat="p in phones">
<td>{{p.id}}</td>
<td>{{p.name}}</td>
<td>{{p.price}}</td>
</tr>
</table>
</div>
</body>
</html>
```



id	name	price
1	갤럭시 S6	3
2	갤럭시 S7	30
3	갤럭시 S8	50
4	LG G3	1
5	LG G4	3
6	LG G5	5
7	하웨이 HPhone	4
8	iPhone SE	20
9	iPhone 6S	30
10	iPhone 7 Plus	90
11	iPhone 5	20
12	iPhone 5S	30
13	iPhone 6	40
14	iPhone 6+	50

Spring Data JPA

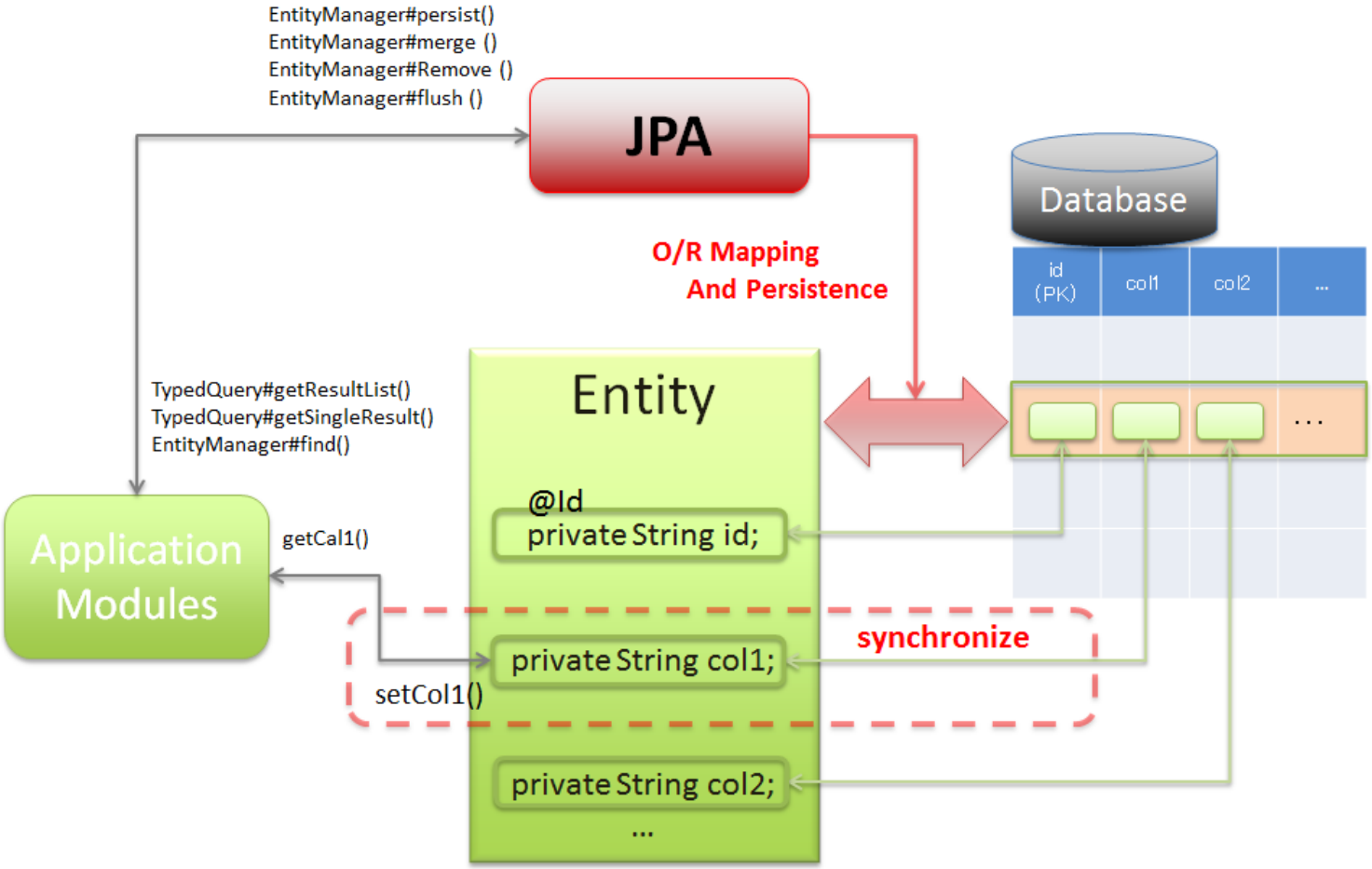
Spring Data JPA의 전체 구조



JPA란?

- Java Persistence API의 약자. RDBMS(관계형 데이터베이스)에서 관리되는 레코드를 Java 객체로 맵핑하거나, 맵핑된 객체에 대해 조작한 내용을 DB에 반영하는 방법을 Java API의 사양으로 정의해 놓은 것.
- 사양만 정의해 놓을 것을 뿐 실제 구현은 없음. JPA의 구현은 각 O/R Mapper 프로바이더가 담당. 대표적인 프로바이더로 Hibernate가 있음.

JPA의 O/R Mapping

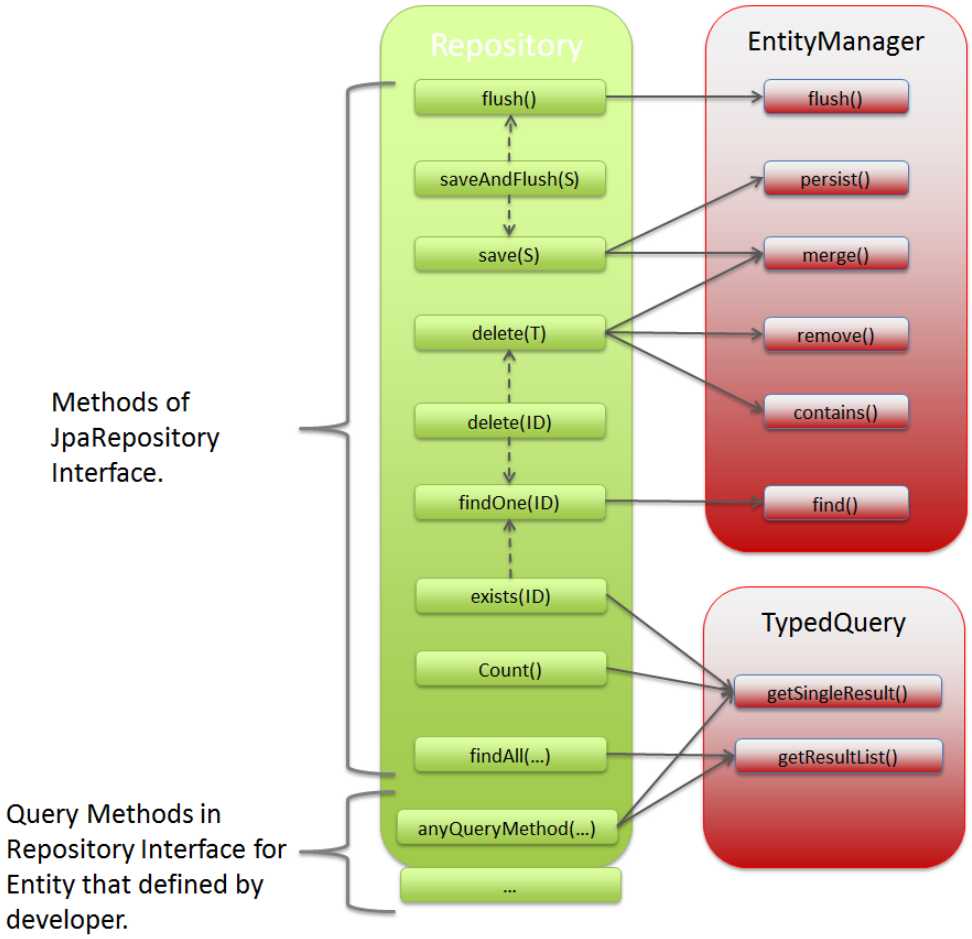


Entity 클래스

- RDB에서 관리되는 레코드를 표현하는 클래스. @javax.persistence.Entity어노테이션을 붙이면 Entity 클래스가 됨

Spring Data JPA

- Spring Data JPA란? JPA를 사용하여 Repository(DAO클래스 같은 것-인터페이스로 작성)를 작성하기 위한 라이브러리
- Spring Data JPA를 이용하면 Query메서드를 부르는 메서드를 Repository인터페이스에 정의하는 것만으로 조건에 일치하는 Entity를 취득 가능.


Repository
flush()
saveAndFlush(S)
save(S)
delete(T)
delete(ID)
findOne(ID)
exists(ID)
Count()
findAll(...)
anyQueryMethod(...)
...
EntityManager
flush()
persist()
merge()
remove()
contains()
find()
TypedQuery
getSingleResult()
getResultList()
Methods of JpaRepository Interface.
Query Methods in Repository Interface for Entity that defined by developer.

Spring Data JPA를 코딩하기 위한 pom.xml에 추가

```
<!-- Spring data jpa -->
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<!-- MySqlConnector -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
</dependency>
<!-- Spring dev tools on -->
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <optional>true</optional>
</dependency>
```

uplus라는 db

	id	name	price
▶	1	갤럭시S8	26300
	2	갤럭시S8+	28600
	3	LG G6	29200
	4	LG X400	1450
	5	갤럭시 A5	12560
	6	IPHONE 7 128GB	38260
	7	LG X300	2490
	8	IPHONE 6S 128GB	21340
	9	갤럭시 노트5 64gb	14590
	10	LG V20	21820
	11	IPHONE 7 Plus 1...	44630
	12	갤럭시 On7	7820
	13	갤럭시 폴더	1590
	14	갤럭시 A7	6180
	15	Galaxy S7 32G	21620
*	NULL	NULL	NULL

기존의 Phone클래스를 Entity화

```
@Entity
public class Phone {
    @Id
    private int id;
```

application.properties에 spring.datasource설정

- #mysql settings
- spring.datasource.url = jdbc:mysql://localhost:3306/uplus
- spring.datasource.username = uplus
- spring.datasource.password = uplususer
- spring.datasource.driver-class=com.mysql.jdbc.Driver

Dao까지 만듦

- **package** com.lg.dao;
-
- **import** org.springframework.data.repository.CrudRepository;
-
- **import** com.lg.ws.model.Phone;
-
- **public interface** PhoneRepository **extends** CrudRepository<Phone, Integer> {
-
- }

컨트롤러에 추가

```
@Autowired
    private PhoneRepository phoneRepository;

    @RequestMapping(value="/", method=RequestMethod.GET)
    public ModelAndView index() {

            ModelAndView mav = new ModelAndView();
            mav.setViewName("index");
            mav.addObject("msg", "안녕하세요");
            mav.addObject("phoneList", phoneRepository.findAll());
            return mav;
    }
```

html 수정

```
<table class="table table-hover">
        <thead>
            <tr>
                <td>ID</td>
                <td>이름</td>
                <td>가격</td>
            </tr>
        </thead>
        <tbody>
            <tr th:each="p : ${phoneList}">
                <td th:text="${p.id}">1</td>
                <td th:text="${p.name}">베가 시리우스</td>
                <td th:text="${p.price}">897,000</td>
            </tr>
        </tbody>
    </table>
```

안녕하세요

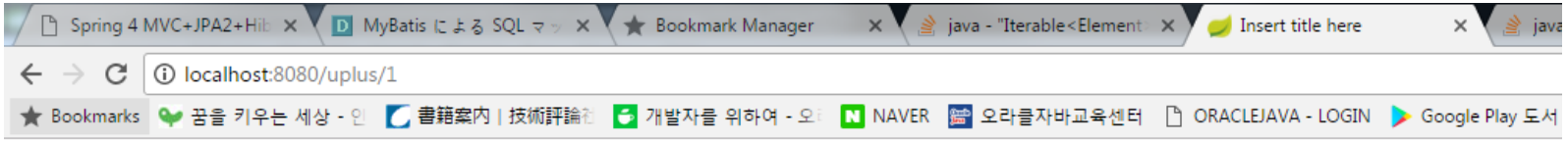
ID	이름	가격
1	갤럭시S8	26300
2	갤럭시S8+	28600
3	LG G6	29200
4	LG X400	1450
5	갤럭시 A5	12560
6	IPHONE 7 128GB	38260
7	LG X300	2490
8	IPHONE 6S 128GB	21340
9	갤럭시 노트5 64gb	14590
10	LG V20	21820

페이징 처리 (컨트롤러)

- Page<Player> players = playerRepository.findAll(
- **new PageRequest(pageNumber-1, *PAGE_SIZE, Sort.Direction.ASC, "name"));**
- **int current = players.getNumber() + 1;**
- **int begin = 1*;***
- **int end = *players.getTotalPages();***

- model.put("playerList", players);
- model.put("beginIndex", begin);
- model.put("endIndex", end);
- model.put("currentIndex", current);

-
- <ul class="pagination">
- <li th:each="i : ${#numbers.sequence(beginIndex, endIndex)}" class="page-item" th:classappend="${currentIndex == i} ? active : ''">
- <a th:href="'/Player/' + ${i}" ><span th:text="${i}">1
-
-
-



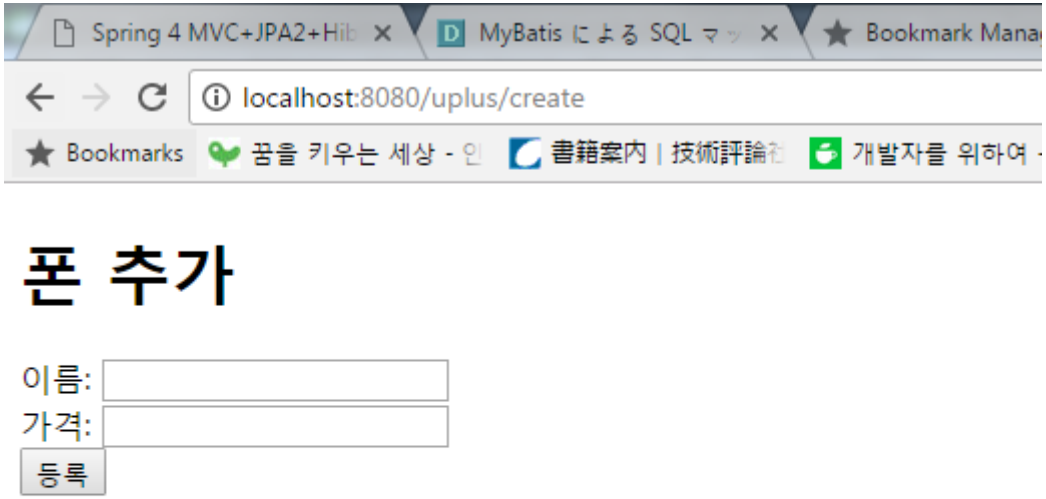
폰 추가

ID	이름	가격(단위: 만원)
11	iPhone 5	20
12	iPhone 5S	30
13	iPhone 6	40
14	iPhone 6+	50
9	iPhone 6S	30
10	iPhone 7 Plus	90
8	iPhone SE	20
4	LG G3	1
5	LG G4	3
6	LG G5	5

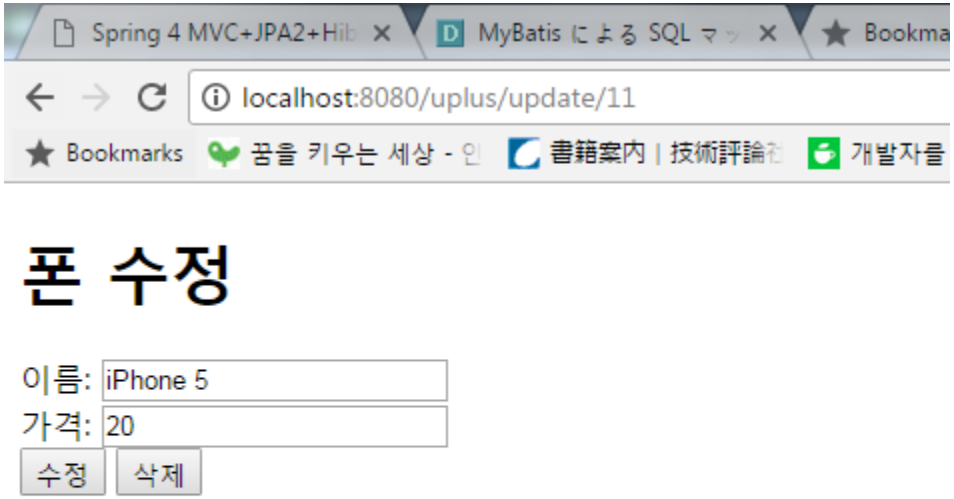
1 2

CRUD 작성

- 추가화면
- 수정화면
- Controller수정


Spring 4 MVC+JPA2+Hib
MyBatis による SQL マッ
Bookmark Mana
localhost:8080/uplus/create
Bookmarks
꿈을 키우는 세상 - 인
書籍案内 | 技術評論社
개발자를 위하여
폰 추가
이름:
가격:
등록

- <h1>폰 추가</h1>
- <form method="post" action="">
- <label for="name">이름:</label>
- <input type="text" name="name" id="name" />
- <label for="name">가격:</label>
- <input type="text" name="price" id="price" />
- <input type="submit" value="등록" />
- </form>


Spring 4 MVC+JPA2+Hib
MyBatis による SQL マッ
Bookma
localhost:8080/uplus/update/11
Bookmarks
꿈을 키우는 세상 - 인
書籍案内 | 技術評論社
개발자를
폰 수정
이름: iPhone 5
가격: 20
수정
삭제

- <h1>폰 수정</h1>
- <form method="post" action="" th:object="${phone}">
- <input type="hidden" name="id" th:field="*{id}"/>
- <label for="name">이름:</label>
- <input type="text" name="name" id="name" th:field="*{name}"/>
- <label for="name">가격:</label>
- <input type="text" name="price" id="price" th:field="*{price}"/>
- <input type="submit" value="수정" name="update" />
- <input type="submit" value="삭제" name="delete" />
- </form>

- @RequestMapping(value="/uplus/create", method=RequestMethod.*GET)*
- **public String create(Model model) {**
- **return "phoneCreate";**
- }
- @RequestMapping(value="/uplus/create", method=RequestMethod.*POST)*
- **public String create(Phone phone, Model model) {**
- phoneRepository.save(phone);
- **return "redirect:/uplus/1";**
- }

- @RequestMapping(value="/uplus/update/{id}", method=RequestMethod.*GET)*
- **public String update(@PathVariable Integer id, Model model) {**
- Phone phone = phoneRepository.findOne(id);
- model.addAttribute("phone", phone);
- **return "phoneUpdate";**
- }

- @RequestMapping(params="update", value="/uplus/update/{id}", method=RequestMethod.*POST)*
- **public String update(Phone phone, Model model) {**
- Phone sphone = phoneRepository.findOne(phone.getId());
- sphone.setName(phone.getName());
- sphone.setPrice(phone.getPrice());
- phoneRepository.save(sphone);
- **return "redirect:/uplus/1";**
- }

- @RequestMapping(params="delete", value="/uplus/update/{id}", method=RequestMethod.*POST)*
- **public String delete(@PathVariable Integer id, Model model) {**
- Phone sphone = phoneRepository.findOne(id);
- phoneRepository.delete(sphone);
- **return "redirect:/uplus/1";**
- }

JPQL 과 Spring Data JPA Query

JPQL(Java Persistence Query Language)

- JPA에서 Query작성방법
  - JPQL
  - Navite Query
  - Criteria API → QueryDSL

JPQL 문법

- Select p from Player p where p.age > 40
- 엔티티 명과 엔티티 별명을 적어야 함
- 속성을 참조할때도 p.age와 같이 별명을 기술필요
- Select 구에 엔티티 별명을 기술함으로 select * 과 유사한 쿼리 결과 가져옴

- 각각의 프로퍼피나 집계함수도 사용가능
- Select max(p.age), p.name from Player p
- 조인 지원
- Select p from Player p join Team t on m.teamId = t.id
  where p.age > 40 orderby t.name
- ※ JPQL에서는 내부결합과 좌외부결합만 지원

- 갱신, 삭제 쿼리 지원
  - Entity Property 갱신이나 remov는 1건씩 SQL이 실행됨
  - 복수개의 레코드 조작 가능하게

```
Int deleted = em.createQuery("delete from player p
Where p.age > :age").setParameter("age", "40")
.executeUpdate();
```

- JPQL에서 사용가능한 함수
- https://docs.oracle.com/javaee/7/tutorial/persistence-querylanguage005.htm#BNBVP
- 사용자 정의 함수 실행
  - Select function('myFunc', p.name, 100) from Player p
    - 2개의 인수를 받는 myFunc 이라는 함수가 있는 경우

Spring Data JPA Query method

- Query 자동작성기능
- @Query 사용
- Sort 사용
- Named parameter 사용
- Stored procedure 실행

Query 자동작성기능

- findByName
  - … where p.name = ?1
  - https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods

@Query 사용

- @Query("SELECT p FROM p WHERE :ageFrom <= p.age AND p.age < :ageTo ORDER BY p.age DESC")
  List<Player> findByAge(
  @Param("ageFrom") int ageFrom,
  @Param("ageTo") int ageTo);

Sort 사용

- @Query(“select p from Player p where p.name like ?1%”)
  List<Player> findByNameAndSort(String name, Sort sort);

- Repo.findNameAndSort(“오”, new Sort(“name”));

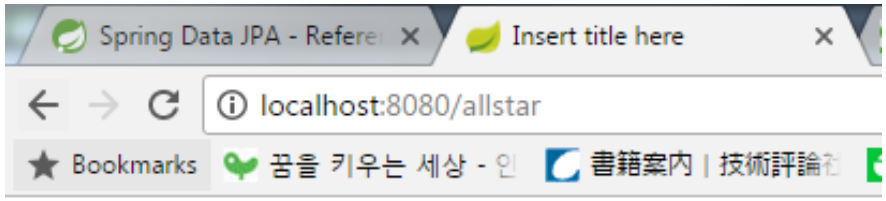
Stored procedure 실행

- Stored procedure 작성
- create procedure plus1inout(in arg int, out res int)set res = arg + 1;

- 확인
- call plus1inout(10, @res);
- select @res;

- @Procedure("plus1inout")
- Integer explicitlyNamedPlus1inout(Integer arg);

- 컨트롤러 호출
- model.addAttribute("number", playerRepository.explicitlyNamedPlus1inout(100));



스토어드 프로시저 호출:101

ID	name	age	comment	덧글쓰기
1	임요환	38	못생겼다 지금 뭘 하나... 잘생겼다 그럭저럭...	덧글쓰기
7	임창민	32		덧글쓰기
8	임창용	42		덧글쓰기
6	임형구	45		덧글쓰기

Query 자동작성기능

JPA를 이용한 One-to-Many RelationShip


Player
Player_id
name
age
fk_comment_playerid
Comment
id
comment
player_id

JPA Entity 정의

- @Entity
- **public class Player {**

- @Id
- @GeneratedValue(strategy=GenerationType.*IDENTITY)*
- **private int id;**
- **private String name;**
- **private int age;**

- @OneToMany(mappedBy="player", cascade = CascadeType.*ALL)*
- **private Set<Comment> comments;**
- **이후 Getter, setter…**

```
@Entity
public class Comment {

public Comment() {
}

public Comment(String comment) {
this.comment = comment;
}

public Comment(String comment, Player player) {
this.comment = comment;
this.player = player;
}
```

- @Id
- @GeneratedValue(strategy=GenerationType.*IDENTITY)*
- **private int id;**

- **private String comment;**

- @ManyToOne
- @JoinColumn(name="player_id")
- **private Player player;**
- **이후 getter, setter**

Comment 작성 컨트롤러

- @RequestMapping(value="/allstar/comment/{id}", method=RequestMethod.*GET)*
- **public String comment(@PathVariable Integer id, Model model){**
- model.addAttribute("playerId",id);
- **return "comment";**
- }

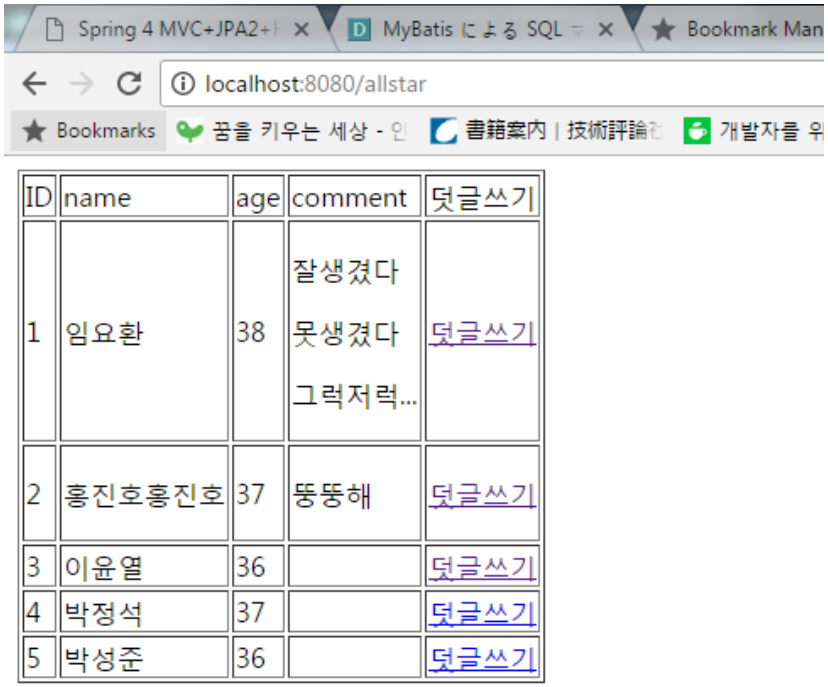
- @RequestMapping(value="/allstar/comment/{id}", method=RequestMethod.*POST)*
- **public String comment(@RequestParam Integer playerId,**
- @RequestParam String comment, Model model){
- Player p = playerRepository.findOne(playerId);
- p.getComments().add(**new Comment(comment, p));**
- playerRepository.save(p);
- **return "redirect:/allstar";**
- }

Thymeleaf 수정(index.html)

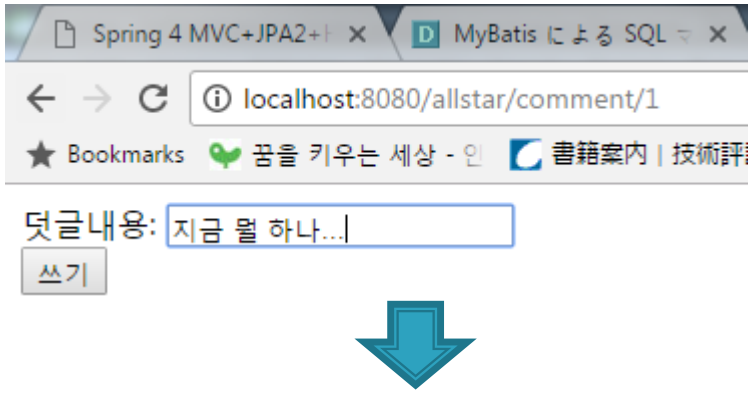
- <tr th:each="p : ${playerList}">
- <td th:text="${p.id}"></td>
- <td th:text="${p.name}"></td>
- <td th:text="${p.age}"></td>
- <td>
-
- 잘생겼다
-
- </td>
- <td>
- <a th:href="'/allstar/comment/' + ${p.id}">덧글쓰기
- </td>
- </tr>

Thymeheaf 추가(comment.html)

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8"/>
<title>Insert title here</title>
</head>
<body>
<form action="" method="post">
<input type="hidden" name="playerId" th:value="${playerId}" />
<label for="name">덧글내용:</label>
<input type="text" name="comment" id="comment" /> <br/>
<input type="submit" value="쓰기" />
</form>
</body>
</html>
```



ID	name	age	comment	덧글쓰기
1	임요환	38	잘생겼다 못생겼다 그럭저럭...	덧글쓰기
2	홍진호홍진호	37	뚱뚱해	덧글쓰기
3	이윤열	36		덧글쓰기
4	박정석	37		덧글쓰기
5	박성준	36		덧글쓰기



ID	name	age	comment	덧글쓰기
1	임요환	38	잘생겼다 못생겼다 그럭저럭... 지금 뭘 하나...	덧글쓰기
2	홍진호홍진호	37	뚱뚱해	덧글쓰기
3	이윤열	36		덧글쓰기
4	박정석	37		덧글쓰기
5	박성준	36		덧글쓰기

검색을 위한 QueryDsl

QueryDSL

- jpa에서 query작성시 타입체크가 안되는 문제(사실 이것때문에 mybatis와 차이점이 많이 없어지게 되지 않을까?)
- querydsl로 해결 가능
- 라이브러리 개발 회사
  - Mysema → http://www.mysema.com/

Pom.xml 추가

- <!-- QueryDSL 관련 라이브러리 -->

- <dependency>
- <groupId>com.querydsl</groupId>
- <artifactId>querydsl-apt</artifactId>
- <scope>provided</scope>
- </dependency>

- <dependency>
- <groupId>com.querydsl</groupId>
- <artifactId>querydsl-jpa</artifactId>
- </dependency>

```
<!-- QueryDSL관련 플러그인 -->
<plugin>
<groupId>com.mysema.maven</groupId>
<artifactId>apt-maven-plugin</artifactId>
<version>1.1.3</version>
<executions>
<execution>
<goals>
<goal>process</goal>
</goals>
<configuration>
<outputDirectory>target/generated-sources/java</outputDirectory>
<processor>com.querydsl.apt.jpa.JPAAnnotationProcessor</processor>
</configuration>
</execution>
</executions>
</plugin>
```

PlayerPredicate.java 작성

- **public class PlayerPredicate {**

- **public static Predicate search(String name) {**
- QPlayer player = QPlayer.***player;***

- BooleanBuilder builder = **new BooleanBuilder();**
- **if (name != null) {**
- builder.and(player.name.like("%" + name + "%"));
- }
- **return builder;**
- }
- }

Repository에 추가

- **public interface PlayerRepository**
- **extends PagingAndSortingRepository<Player, Integer>,**
- QueryDslPredicateExecutor<Player>{
- }

컨트롤러에 적용

- Page<Player> players = playerRepository.findAll(
- PlayerPredicate.*search("나")*,
- **new PageRequest(pageNumber-1, *PAGE_SIZE, Sort.Direction.ASC, "name"));***

PlayerList

Kbo players

Name	Age	Salary
나주환	36	3400
나지환	34	10000

1