

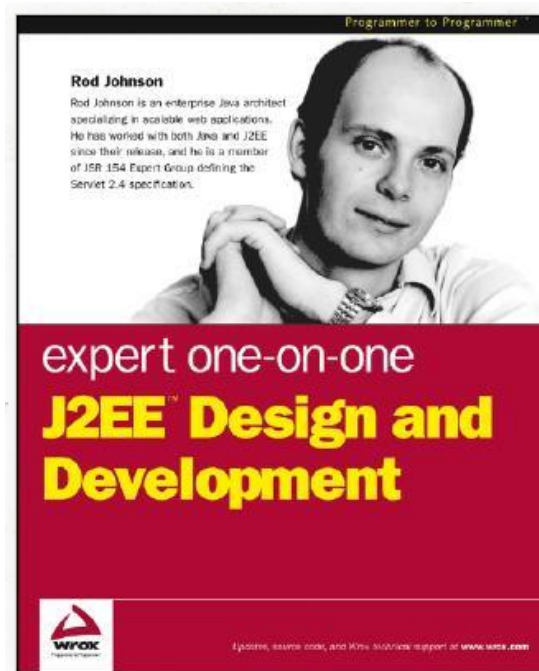
Spring Day 1

▼ Spring

▼ About SPRING Framework

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/20073f9f-2020-45c7-9b1f-9c478041dc37/Expert_One-on-One_J2EE_Development_without_EJB_\(1\).pdf](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/20073f9f-2020-45c7-9b1f-9c478041dc37/Expert_One-on-One_J2EE_Development_without_EJB_(1).pdf)

2002년 로드 존슨



“전통적인 EJB 의 거울을 넘어 새로운 시작이다” 라는 의미로 스프링이라고 명칭을 짓게 되었다.

“ 스프링이라는 프레임워크를 개발했어요.
이제 프레임워크를 사용하면
비즈니스 로직 구현에 집중할 수 있어요. ”

2002년 어느 날, 로드 존슨



프레임워크 —————> 라이브러리

프레임워크와 라이브러리의 가장 큰 차이는 바로 앱의 흐름을 누가 가지고 있느냐의 차이이다.

프레임 워크는 전체적인 흐름을 스스로 쥐고 있다.

스프링의 특징

JSP 방식

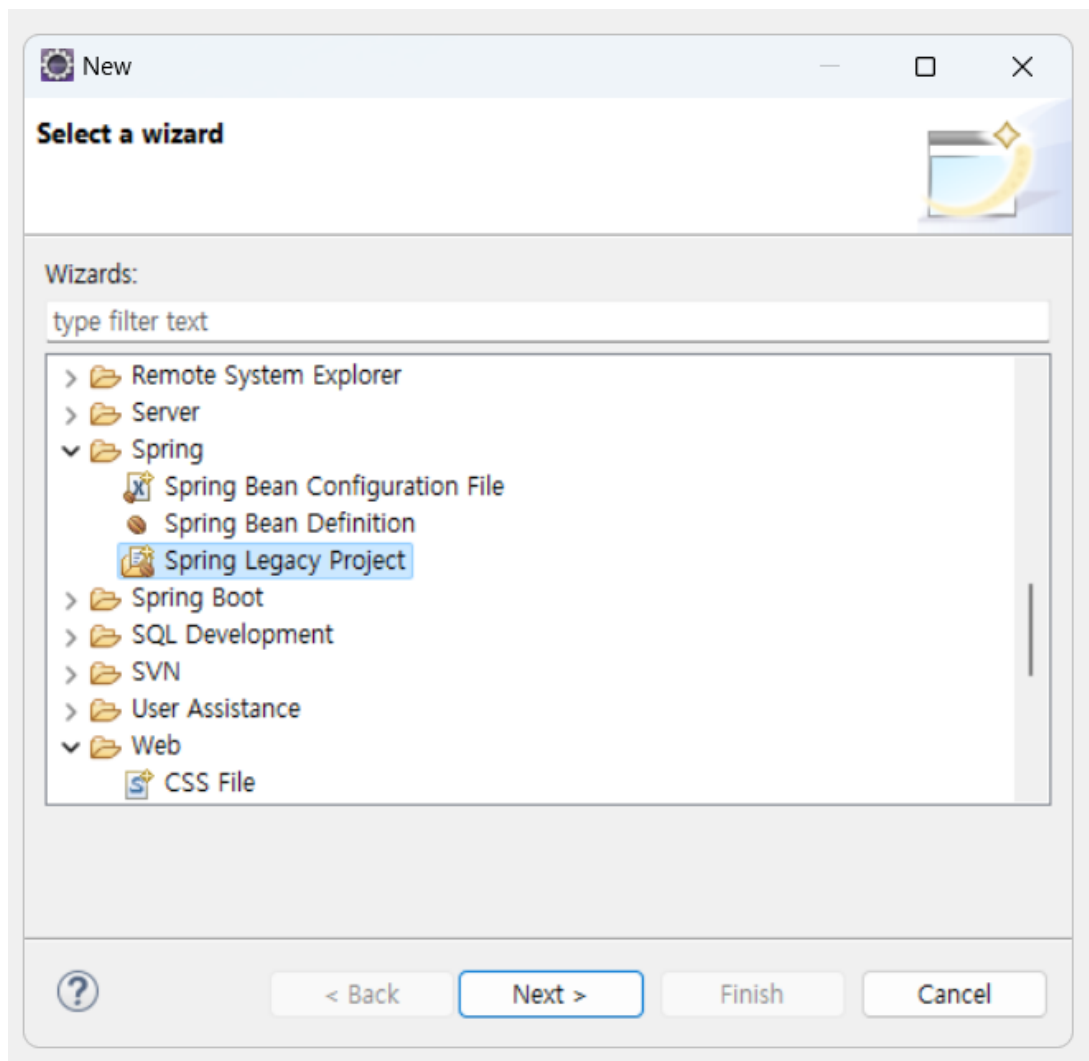
request 요청 —————> 서블릿 —————> JSP (view) —————> 응답 response

스프링 방식

request 요청 —————> **Controller (Model)** —————> JSP (view) —————> 응답 response

브라우저의 호출은 반드시 **Controller** 를 통해서 호출 된다.

▼ Spring Project 만들기.



New Spring Legacy Project

Spring Legacy Project

Click 'Next' to load the template contents.

Project name:


☒ Use default location

Location:

Select Spring version:

Templates:

- Simple Projects
 - Simple Java
 - Simple Spring Maven
 - Simple Spring Web Maven
- Batch
- GemFire
- Integration
- Persistence
- Simple Spring Utility Project
- Spring MVC Project**

 requires downloading [Configure templates...](#)

Description:
A new Spring MVC web application development project

URL: <https://dist.springsource.com/release/STS/help/org.springframework.templates.mvc-3.2.2.zip>

Working sets

☐ Add project to working sets

Working sets:

New Spring Legacy Project

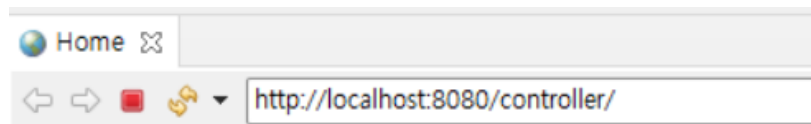
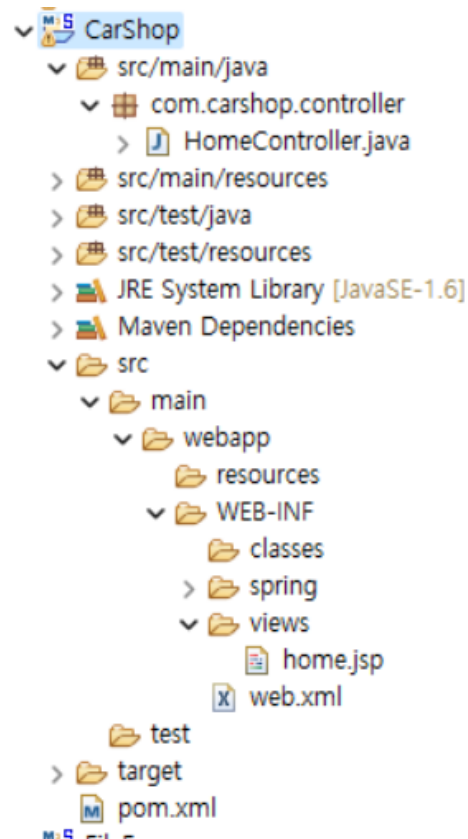
Project Settings - Spring MVC Project

Define project specific settings. Required settings are denoted by "*".

Please specify the top-level package e.g. com.mycompany.myapp*

com.carshop.controller

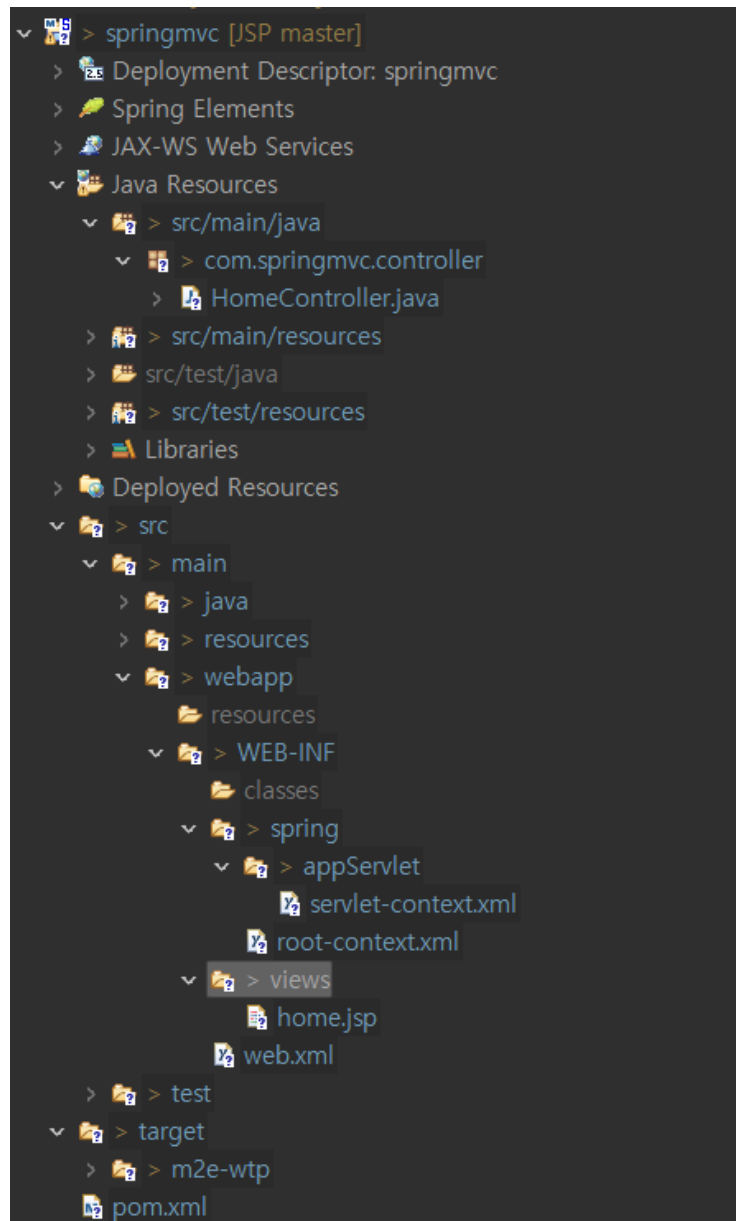
? < Back Next > Finish Cancel



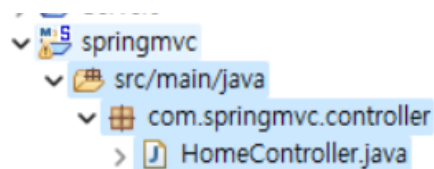
Hello world!

The time on the server is 2023? 2? 27? (?) ?? 9? 25? 12?.

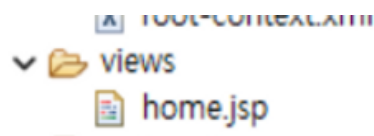
Spring Project 구조.



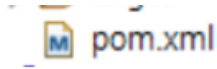
자바 파일들의 위치.



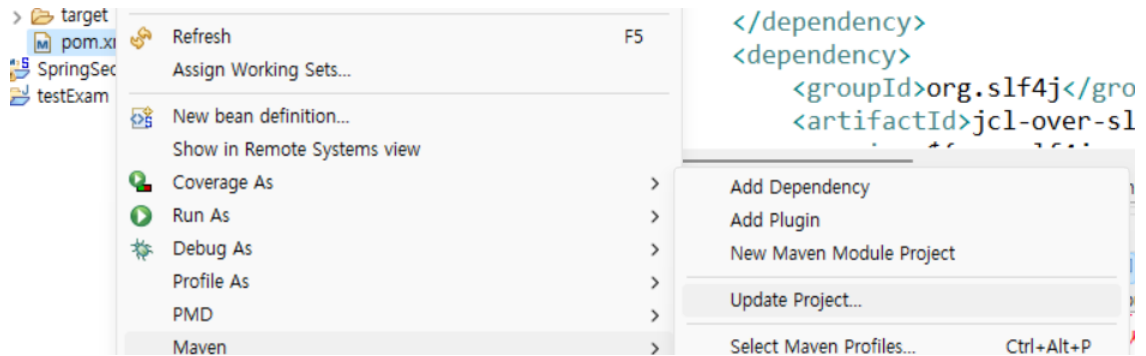
View(jsp) 위치.



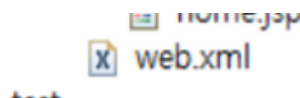
기본 환경 설정 파일(Maven 설정)



pom.xml 을 수정한 뒤에는 반드시 **Update Project(Alt + F5)**



web.xml 프로젝트 설정 파일.



▼ Spring Basic

호출 흐름

request → web.xml → servlet-context.xml → Controller 호출 → servlet-context.xml
→ JSP view 호출 → response



web.xml 중요!(수정/설정할 부분이 아주 많은 파일)

요청이 들어오면 가장 먼저 web.xml 파일을 읽어서 차례대로 해석하게 된다.

root-context.xml

공유하는 자원들을 선언하는 용도.

View와 관련되지 않은 Service, Repository DB 객체들을 정의하게 된다.

servlet-context.xml

웹 요청을 직접 처리할 Controller의 매핑을 설정하거나 View를 어떻게 처리 할지 설정하게 된다.

Controller

```
package com.springmvc.controller;

import java.text.DateFormat;
import java.util.Date;
import java.util.Locale;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

/**
 * Handles requests for the application home page.
 */
컨트롤러 @Controller 자바 클래스 가 컨트롤러 역할을 하는 자바 클래스임을 알려준다.
@Controller
public class HomeController {

    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);

    /**
     * Simply selects the home view to render by returning its name.
     */

    요청(request)이 있을때 호출되는 매서드를 지정하는 역할
    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
        logger.info("Welcome home! The client locale is {}. ", locale);

        Date date = new Date();
        DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG, DateFormat.LONG, locale);

        String formattedDate = dateFormat.format(date);

        model.addAttribute("serverTime", formattedDate );
    }
}
```

```
수행 결과를 어디로 전송할지를 설정한다. home은 home.jsp 를 의미하게 된다.  
    return "home";  
}  
  
}
```

계층적 구조

계층적 구조 없이 한곳에서 모든 작업을 한번에 처리하면

- 코드의 복잡성 증가
- 유지 보수의 어려움
- 유연성 부족
- 중복 코드 증가
- 낮은 확장성

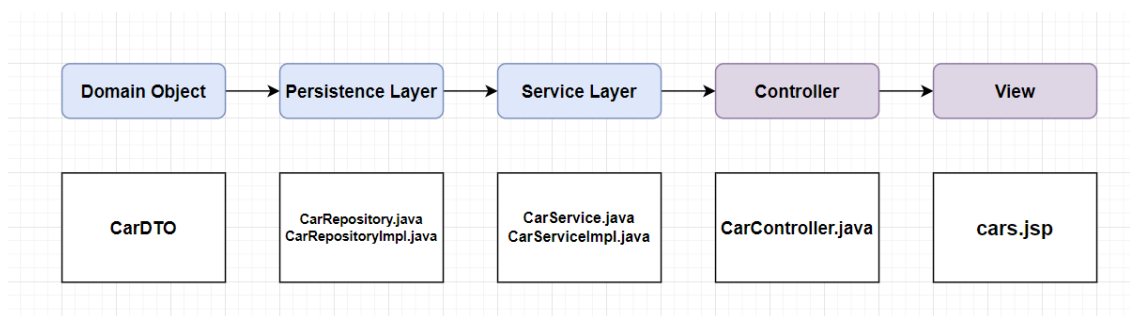
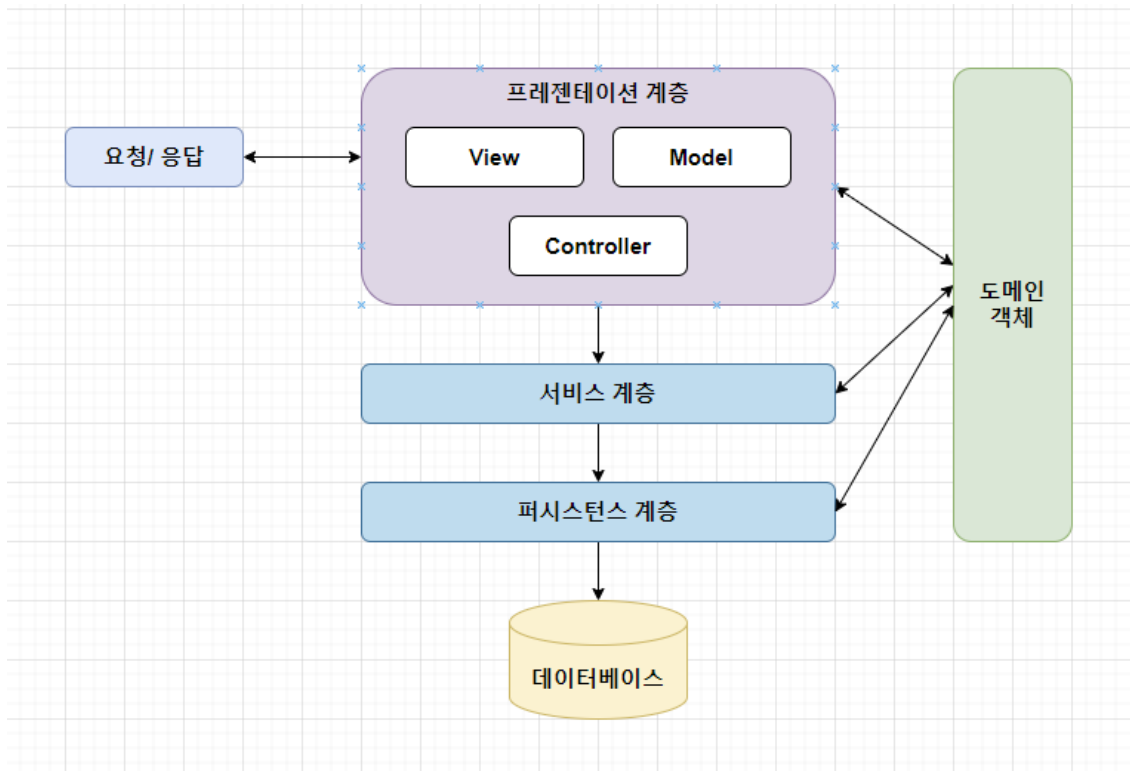
등의 문제가 발생할 수 있다.

계층적 구조는

- 퍼시스턴스(Persistence) 계층
- 서비스(Service) 계층
- 프레젠테이션(Presentation) 계층

으로 분리하여 사이트를 구성한다. MVC는 이것들 중에서 프레젠테이션 계층에 속하게 된다.

Spring 구조.



도메인 객체(Domain Object) : 데이터 모델, 객체 정보를 저장한다.
(jsp에서의 DTO와 같다.)

퍼시스턴스 계층(Persistence Layer) : 데이터 액세스 계층, DB에 접근하여 데이터를 처리한다. (jsp에서의 DAO와 같다.)

서비스 계층(Service Layer) : 애플리케이션에서 제공하는 포괄적인 서비스를 표현한다.
(비즈니스 계층이라고도 한다.)

DTO

CarDTO.java

```
package com.carshop.controller;

public class CarDTO {

    private String cid, cname, cprice, ccate, cdesc;

    public String getCid() {
        return cid;
    }

    public void setCid(String cid) {
        this.cid = cid;
    }

    public String getName() {
        return cname;
    }

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

    public String getCprice() {
        return cprice;
    }

    public void setCprice(String cprice) {
        this.cprice = cprice;
    }

    public String getCcate() {
        return ccate;
    }

    public void setCcate(String ccate) {
        this.ccate = ccate;
    }

    public String getCdesc() {
        return cdesc;
    }

    public void setCdesc(String cdesc) {
        this.cdesc = cdesc;
    }

    public CarDTO(String cid, String cname, String cprice, String ccate, String cdesc) {
        this.cid = cid;
        this.cname = cname;
        this.cprice = cprice;
        this.ccate = ccate;
        this.cdesc = cdesc;
    }

    public CarDTO() {
        super();
    }

}
```

Interface

1. 인터페이스는 일종의 명령서이다.
2. 인터페이스는 다형성을 표현/제어한다.

CarRepository.java 인터페이스

```
package com.carshop.controller;

import java.util.List;

public interface CarRepository {

    List<CardTO> getAllCarList();

}
```

CarRepositoryImpl.java 구현 클래스

```
package com.carshop.controller;

import java.util.*;

import org.springframework.stereotype.Repository;

@Repository
public class CarRepositoryImpl implements CarRepository {

    private List<CardTO> listOfCars = new ArrayList<CardTO>();

    public CarRepositoryImpl() {
        CardTO car1 = new CardTO("c0001", "람보르기니", "20000", "스포츠카", "신규");
        CardTO car2 = new CardTO("c0002", "그랜저", "3500", "승용차", "신규");
        CardTO car3 = new CardTO("c0003", "아반테", "2000", "승용차", "신규");

        listOfCars.add(car1);
        listOfCars.add(car2);
        listOfCars.add(car3);
    }

    @Override
    public List<CardTO> getAllCarList() {

        return listOfCars;
    }

}
```

CarService.java 인터페이스

```
package com.carshop.controller;

import java.util.*;

public interface CarService {
```

```
List<CarDTO> getAllCarList();  
}
```

CarServiceImpl.java 구현 클래스

```
package com.carshop.controller;  
  
import java.util.List;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
@Service  
public class CarServiceImpl implements CarService{  
  
    @Autowired  
    private CarRepository carRepository;  
  
    @Override  
    public List<CarDTO> getAllCarList() {  
        return carRepository.getAllCarList();  
    }  
  
}
```

CarController.java

```
package com.carshop.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;  
  
@Controller  
public class CarController {  
  
    @Autowired  
    private CarService carService;  
  
    @RequestMapping("/cars")  
    public String CarList(Model model) {  
        List<CarDTO> list = carService.getAllCarList();  
        model.addAttribute("carList", list);  
        return "cars";  
    }  
  
}
```

Autowired

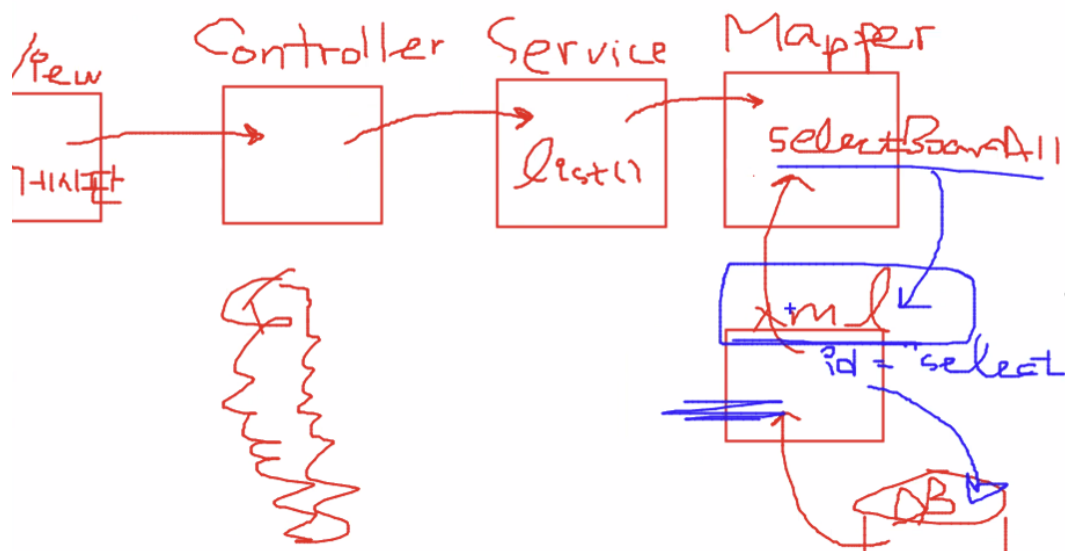
필요한 의존 객체의 "타입"에 해당하는 빈을 찾아 주입한다.

- 생성자
- setter
- 필드

위의 3가지의 경우에 Autowired를 사용할 수 있다.

(Autowired는 기본값이 true이기 때문에 의존성 주입을 할 대상을 찾지 못한다면 애플리케이션 구동에 실패한다.)

▼ etc.



▼ Python

Crawling Basic

본인이 사용하는 크롬 버전을 확인할것.

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/f67170d5-6c77-4714-b53b-529691532c27/chromedriver_win32.zip

```
# 최초 한번만 설치 BeautifulSoup
# !pip install bs4
# 웹 페이지에서 긁어온 txt를 분석하기 쉽게 가공해주는 역할
```

```
import requests # 웹 페이지 읽어오는 모듈
from bs4 import BeautifulSoup as bs # 읽어온 웹페이지 정제
```

```
url = "https://www.naver.com/"
txt = requests.get(url) # 읽어온 페이지가 의미가 없는 그냥 글자일 뿐이다.
# print(txt.text)
```

```
url = "https://www.naver.com/"
txt = requests.get(url) # 웹페이지 그냥 글자로 읽어오기
html = bs(txt.text) # 읽어온 그냥 글자를 의미가 있는 html 로 변환 bs
# print(html)
```

Bugs

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/f98c339c-fd8c-4189-a683-2d9a37a3101b/bugs.xlsx>

```
#####
# Bugs 차트 읽어오기 #
#####

url = "https://music.bugs.co.kr/chart"
txt = requests.get(url) # 웹페이지 그냥 글자로 읽어오기
html = bs(txt.text) # 읽어온 그냥 글자를 의미가 있는 html 로 변환 bs
# print(html)
```



```

len(html.select('tr')) # select 를 통해 특정 요소를 전부 추출한다.

len(html.select('table > tbody > tr')) # 꺾쇠는 상하 관계를 표현한다. 범위를 점차 확대해가면서 줄인다.

len(html.select('table.byChart > tbody > tr')) # .점은 클래스를 의미한다.

# 최종 원하는 100곡을 찾을 수 있었다.

# 전체 소스에서 위에서 찾아낸 100개의 tr들만을 추출하자.

songs = html.select('table.byChart > tbody > tr')
print(songs)

song = songs[0]
print(song)

len(song.select('a'))

len(song.select('p > a'))

len(song.select('p.title > a'))

print(song.select('p.title > a'))

print(song.select('p.title > a')[0].text)

# 1등 노래제목 추출 완료

# 2등 노래제목 추출 하세요

print(songs[1].select('p.title > a')[0].text)

# 1등의 가수명을 추출하세요.

print(song.select('p.artist > a')[0].text)

# 전체 100곡의 노래제목과 가수명 추출

song_data = []
rank = 1

for song in songs:

    title = song.select('p.title > a')[0].text
    singer = song.select('p.title > a')[0].text

    song_data.append(['Bugs', rank, title, singer])
    rank += 1

import pandas as pd

df = pd.DataFrame(song_data, columns=['서비스', '순위', '타이틀', '가수'])
df

# df 을 excel 파일로 변환

df.to_excel('bugs.xlsx', index=False)

```

Melon

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/98f56f7a-43b8-438f-9965-9ded97c907bf/Melon\(%EA%B9%80%EB%8F%84%EC%98%81\).xlsx](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/98f56f7a-43b8-438f-9965-9ded97c907bf/Melon(%EA%B9%80%EB%8F%84%EC%98%81).xlsx)

```
#####
#                               Melon 차트 읽어오기                               #
#####

url = "https://www.melon.com/chart/"
txt = requests.get(url) # 웹페이지 그냥 글자로 읽어오기
html = bs(txt.text)     # 읽어온 그냥 글자를 의미가 있는 html 로 변환 bs

print(html)

# 멜론도 위에서의 벅스처럼 크롤링을 시도하였으나 사이트에서 크롤링을 막고 있다.
# 따라서 다른 방식으로 크롤링을 시도해야 한다.

!pip install selenium

import selenium

from selenium import webdriver
# 위에서의 request와는 다른 방식의 크롤링 모듈

driver = webdriver.Chrome('chromedriver.exe')
driver.get("https://www.melon.com/chart/index.htm")

txt = driver.page_source
html = bs(txt) # 벅스 문법과 약간 다름
print(html)

songs = html.select('tbody > tr')
song_data = []
rank = 1

for song in songs:

    title = song.select('div.rank01 > span > a')[0].text
    singer = song.select('div.rank02 > a')[0].text

    song_data.append(['Melon', rank, title, singer])
    rank += 1

df = pd.DataFrame(song_data, columns=['서비스', '순위', '타이틀', '가수'])
df

df.to_excel('Melon.xlsx', index=False)
```

Genie

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4822f979-2716-46ac-aa75-1af24b4447ba/genie\(%EA%B9%80%EB%8F%84%EC%98%81\).xlsx](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4822f979-2716-46ac-aa75-1af24b4447ba/genie(%EA%B9%80%EB%8F%84%EC%98%81).xlsx)

```

driver = webdriver.Chrome('chromedriver.exe')
driver.get("https://www.genie.co.kr/chart/top200?ditc=D&ymd=20230227&hh=16&rtm=Y&pg=1")

txt = driver.page_source
html = bs(txt) # 벅스 문법과 약간 다름

songs = html.select('tbody > tr')
song_data = []
rank = 1

for song in songs:

    title = song.select('td > a.title')[0].text.strip()
    singer = song.select('td > a.artist')[0].text.strip()

    song_data.append(['Genie', rank, title, singer])
    rank += 1

driver = webdriver.Chrome('chromedriver.exe')
driver.get("https://www.genie.co.kr/chart/top200?ditc=D&ymd=20230227&hh=16&rtm=Y&pg=2")

txt = driver.page_source
html = bs(txt) # 벅스 문법과 약간 다름

songs = html.select('tbody > tr')

for song in songs:

    title = song.select('td > a.title')[0].text.strip()
    singer = song.select('td > a.artist')[0].text.strip()

    song_data.append(['Genie', rank, title, singer])
    rank += 1

df = pd.DataFrame(song_data, columns=['서비스', '순위', '타이틀', '가수'])
df

df.to_excel('genie(김도영).xlsx', index=False)

```

https://www.starbucks.co.kr/store/store_map.do(서울에 있는 스타벅스 전체 지점 정보)

▼ .select() vs .find()

두가지 모두 태그를 찾아주는 메서드이다. 약간 다른점이 있다.

.select() : 괄호 안의 조건에 해당하는 태그를 모두 추출

.find()_all : 괄호 안의 조건에 해당하는 태그를 모두 추출

.select_one() : 괄호 안의 조건에 해당하는 태그를 하나만 추출

.find() : 괄호 안의 조건에 해당하는 태그를 하나만 추출

가장 큰 차이점은 `.find()`는 참/거짓 조건을 넣어서 필터링 할 수 있다는 것이다.