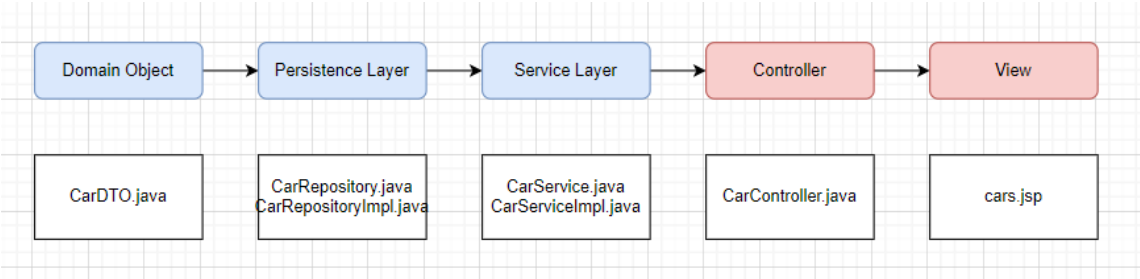
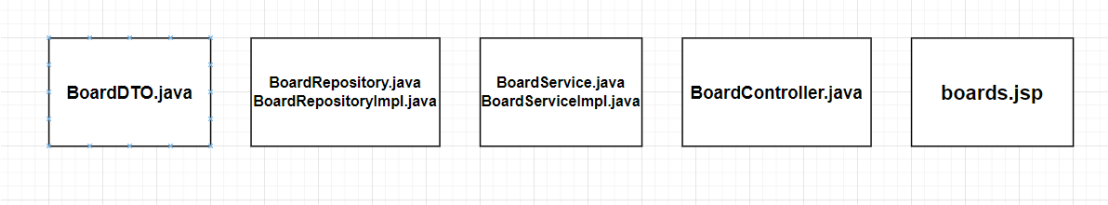
Spring Day 2

▼ Spring

▼ Homework Review



▼ 실습(Board)



BoardDTO.java

```
package com.carshop.controller;

public class BoardDTO {

  private String btitle, bcontent, bauthor, date;

  public String getBtitle() {
    return btitle;
  }

  public void setBtitle(String btitle) {
    this.btitle = btitle;
  }

  public String getBcontent() {
    return bcontent;
  }

  public void setBcontent(String bcontent) {
    this.bcontent = bcontent;
  }

  public String getBauthor() {
    return bauthor;
  }

  public void setBauthor(String bauthor) {
```

```
    this.bauthor = bauthor;
  }

  public String getDate() {
    return date;
  }

  public void setDate(String date) {
    this.date = date;
  }

  public BoardDTO(String btitle, String bcontent, String bauthor, String date) {
    this.btitle = btitle;
    this.bcontent = bcontent;
    this.bauthor = bauthor;
    this.date = date;
  }

  public BoardDTO() {
    super();
  }




}
```

board.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<%@ page session="false" %>
<html>
<head>
  <title>board</title>
  <%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
      <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KFdBI
</head>
<body>
<nav class="navbar navbar-expand navbar-dark bg-dark" aria-label="Second navbar example">
    <div class="container-fluid">
      <a class="navbar-brand" href="#">CarShop</a>
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarsExample02" aria-contro
        <span class="navbar-toggler-icon"></span>
      </button>

      <div class="collapse navbar-collapse" id="navbarsExample02">
        <ul class="navbar-nav me-auto">
          <li class="nav-item">
            <a class="nav-link active" aria-current="page" href="/">홈</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="/cars">차량보기</a>
          </li>
                    <li class="nav-item">
            <a class="nav-link" href="/board">게시판</a>
          </li>
        </ul>
        <form role="search">
          <input class="form-control" type="search" placeholder="Search" aria-label="Search">
        </form>
      </div>
    </div>
  </nav>
<div class="alert alert-dark" role="alert">
<div class="container"><h1>게시판(김도영)</h1></div>
</div>


  <div class="container">
    <div class="row" align="center">

      <c:forEach items="${boardList}" var="board">
        <div class="col-md-4">
        <h3>${board.btitle}</h3>
        <p>${board.bcontent}
```

```
        <p>${board.bauthor}
        <p>${board.date}


        </div>


      </c:forEach>

    </div>
  </div>



</body>
</html>
```

BoardController.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 BoardController {

  @Autowired    // DI
  private BoardService boardService;

  @RequestMapping("/board")
  public String BoardList(Model model) {
    List<BoardDTO> list = boardService.getAllBoardList();
    model.addAttribute("boardList", list);
    return "board";
  }

}
```

BoardService.java

```
package com.carshop.controller;

import java.util.*;

public interface BoardService {

  List<BoardDTO> getAllBoardList();
}
```

BoardServiceImpl.java

```
package com.carshop.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class BoardServiceImpl implements BoardService{

  @Autowired    // DI 의존성 주입 IoC 제어의 역전
  private BoardRepository boardRepository;
```

```
  @Override
  public List<BoardDTO> getAllBoardList() {
    return boardRepository.getAllBoardList();
  }




}
```

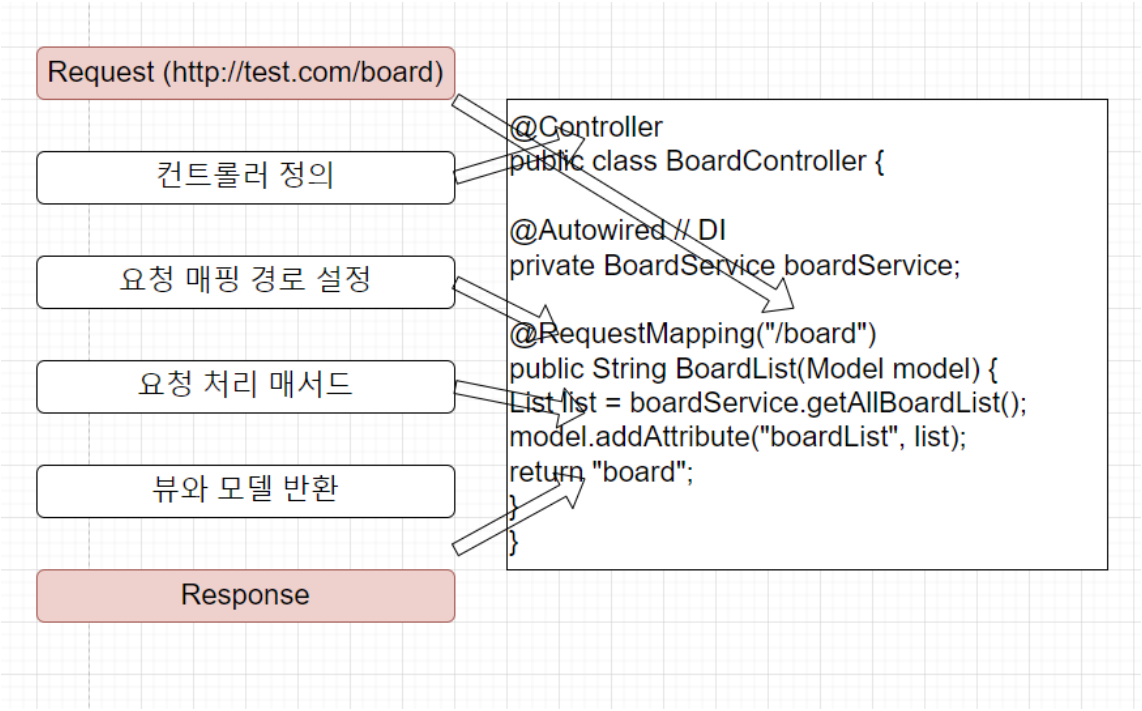
BoardRepository.java

```
package com.carshop.controller;

import java.util.List;

public interface BoardRepository {

  List<BoardDTO> getAllBoardList();

}
```

BoardRepositoryImpl.java

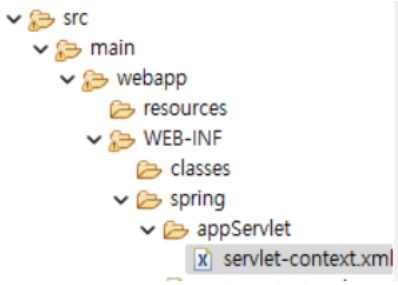
```
package com.carshop.controller;

import java.util.*;

import org.springframework.stereotype.Repository;

@Repository
public class BoardRepositoryImpl implements BoardRepository {

  private List<BoardDTO> listOfBoards = new ArrayList<BoardDTO>();

  public BoardRepositoryImpl() {
    BoardDTO board1 = new BoardDTO("가입인사", "가입했어요.", "하나", "2023/02/27");
    BoardDTO board2 = new BoardDTO("등록", "등록했어요.", "두리", "2023/02/22");
    BoardDTO board3 = new BoardDTO("등업", "등업부탁해요.", "석삼", "2023/02/21");

    listOfBoards.add(board1);
    listOfBoards.add(board2);
    listOfBoards.add(board3);

  }

  @Override
  public List<BoardDTO> getAllBoardList() {

    return listOfBoards;
  }

}
```

▼ Controller



컨트롤러는 매서드를 포함하고 있는 일반적인 자바 클래스가 아니라 브라우저에서 들어온 요청 request 를 처리하는 매서드를 포함하고 있는 특정 자바 클래스이다. 반드시 @Controller 를 선언하여 이 클래스가 컨트롤러의 역할을 수행하는 클래스라는 것을 알려야 한다.

사이트 작성시 규모가 커지게 되면 여러개의 컨트롤러를 사용하게 된다. 따라서 컨트롤러 클래스들을 모두 각각 등록해주어야 하는데 편의상 *를 사용하여 한번에 모두 등록할 수 있다.



이렇게 하나하나 각각 등록해야하는 번거로움을 해결할 수 있다.

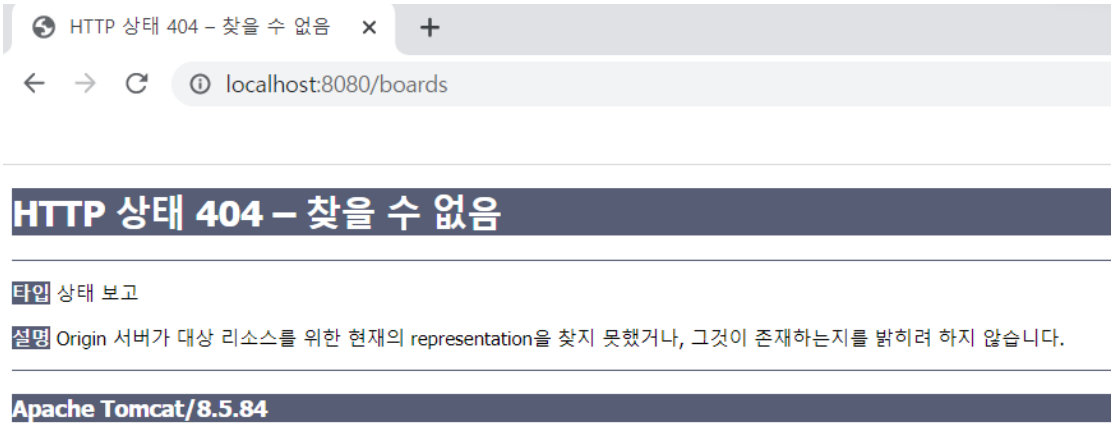
```
<context:component-scan base-package="com.carshop.controller" />
```

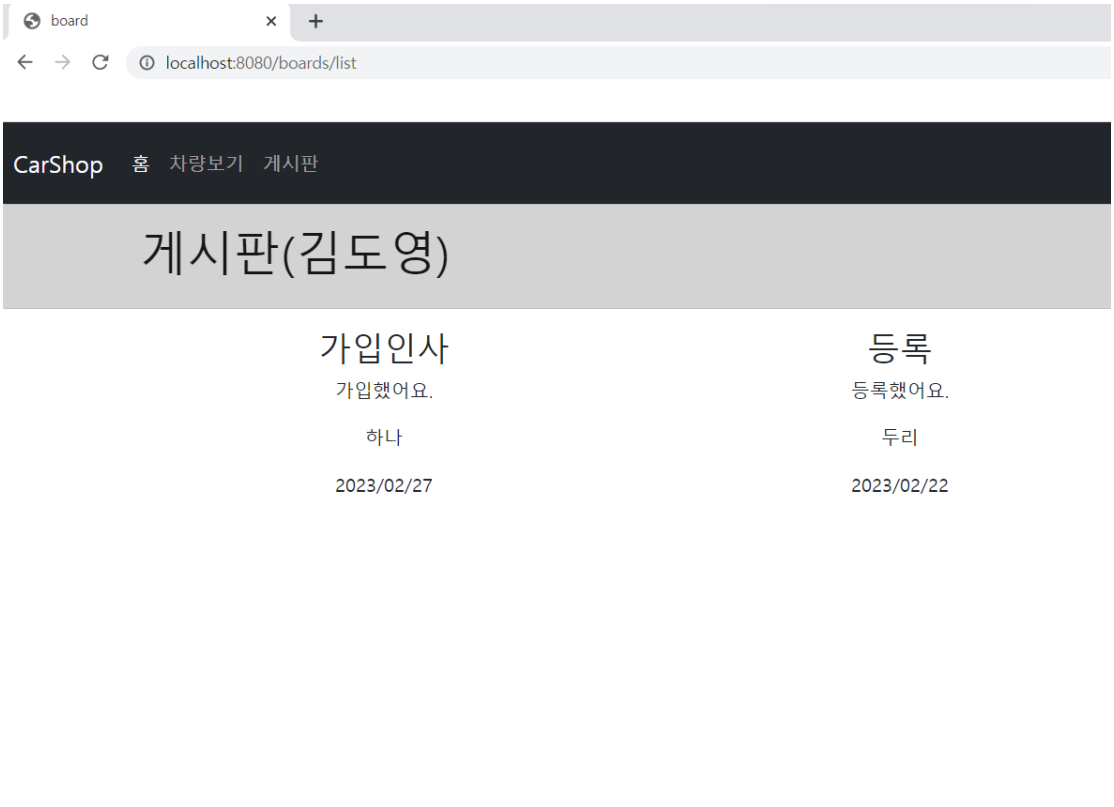
```
<context:component-scan base-package="com.carshop.*" />
```

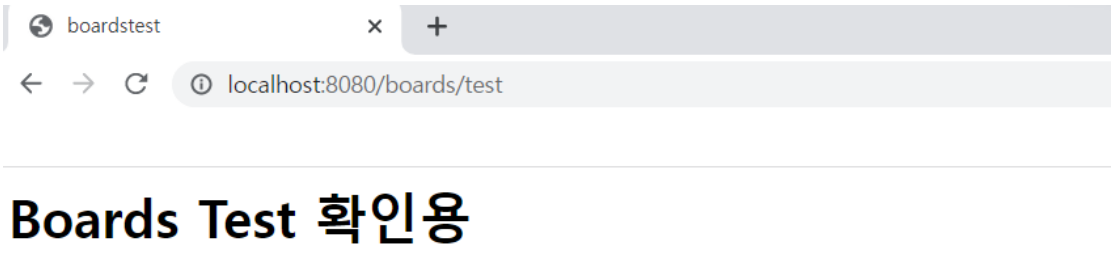
▼ **@RequestMapping on Class**

```java
package com.carshop.controller;

import java.util.List;

@Controller
@RequestMapping(value = "/boards", method = RequestMethod.GET)
public class BoardController {

    @Autowired      // DI
    private BoardService boardService;

    @RequestMapping("/list")
    public String BoardList(Model model) {
        List<BoardDTO> list = boardService.getAllBoardList();
        model.addAttribute("boardList", list);
        return "board";
    }

    @RequestMapping("/test")
    public String BoardTest() {
        return "boardstest";
    }

}
```

boards 처리 매서드는 존재하지 않아서 예외 발생.

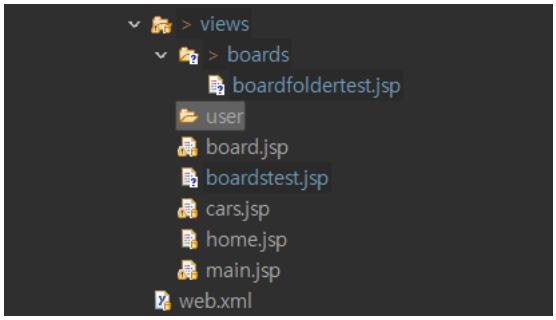




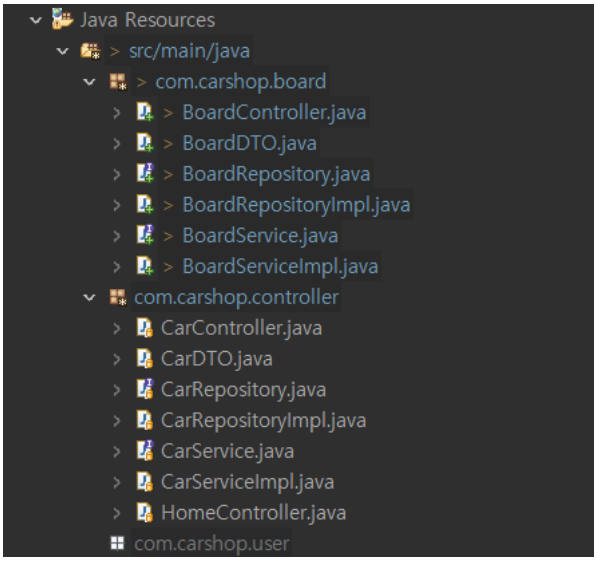


Boards Test 확인용

클래스 단위의 매핑과 매서드 단위의 매핑을 섞어서 사용할 경우
뷰도 폴더를 사용하여 분류하게 되면 관리가 더 편하다(가독성이 좋다.)

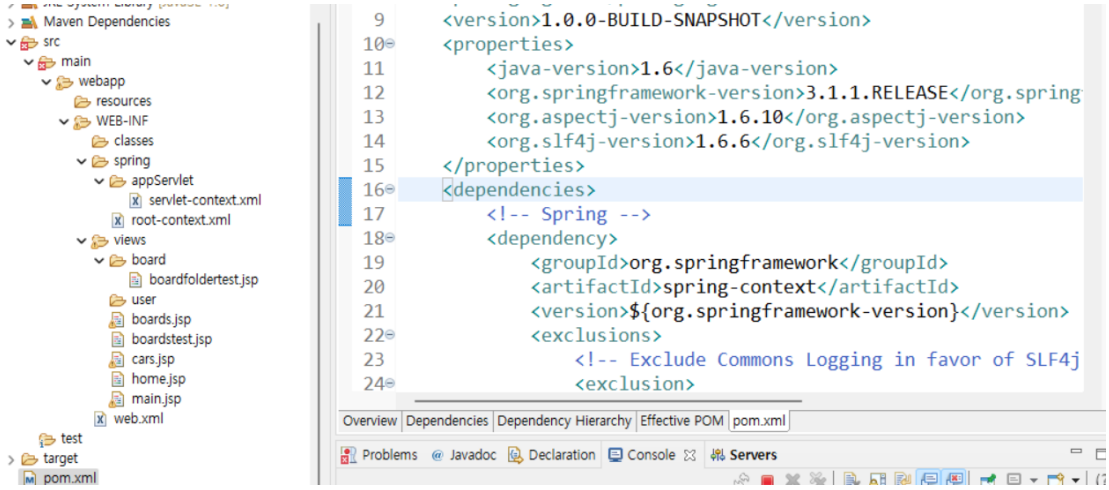


마찬가지로 자바 클래스들도 종류별로 패키지를 분리하여 관리하는 것이 바람직하다.



▼ @GetMapping on Method

> **GetMapping 은 Spring Version이 중요하다.**




```
30
31    @GetMapping("/boardfoldertest")
32    public String boardfoldertest() {
33        return "boards/boardfoldertest";
34    }
```

기본 설정된 스프링 버전 3.1.1에서는 getMapping 을 지원하지 않고 앞으로도 많은 부분에서 3버전대에서는 지원하는 않는 기능이 많으므로 다양한 요소에서 예외를 발생시키게 된다.

따라서 스프링의 기본 버전을 5.x.x 대 이상으로 올려줘야 한다. 최신 버전은 6.x.x 이지만 최신 버전은 다른 버전들과 호환성 면에서 떨어지는 부분이 오히려 더 많기 때문에 조금 낮은 버전을 사용하는 것이 바람직하다.

```xml
<?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 https://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.carshop</groupId>
    <artifactId>controller</artifactId>
    <name>CarShop</name>
    <packaging>war</packaging>
    <version>1.0.0-BUILD-SNAPSHOT</version>
    <properties>
        <java-version>1.6</java-version>
        <org.springframework-version>5.3.19</org.springframework-version>
        <org.aspectj-version>1.6.10</org.aspectj-version>
        <org.slf4j-version>1.6.6</org.slf4j-version>
    </properties>
    <dependencies>
```

💡 **pom.xml 을 수정한 후에는 반드시 Update Project 를 해주어야 한다.**

```java

@Controller
@RequestMapping(value = "/boards", method = RequestMethod.GET)
public class BoardController {

    @Autowired      // DI
    private BoardService boardService;

    @GetMapping("/list")
    public String BoardList(Model model) {
        List<BoardDTO> list = boardService.getAllBoardList();
        model.addAttribute("boardList", list);
        return "board";
    }

    @GetMapping("/test")
    public String BoardTest() {
        return "boardstest";
    }

    @GetMapping("/boardfoldertest")
    public String BoardForderTest() {
        return "boards/boardfoldertest";
    }

}

```

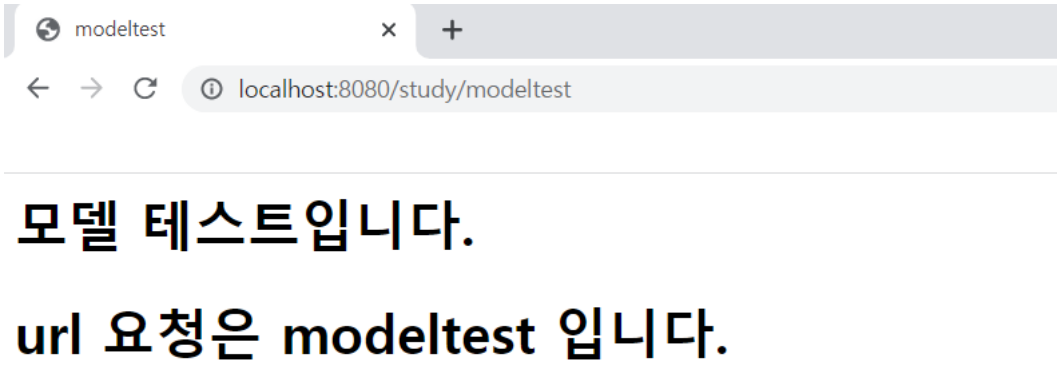
▼ Model[데이터(또는 객체) 정보를 저장]

모델은 사용자의 요청을 처리한 결과 데이터를 관리하고 전달한다.

뷰는 모델을 받아서 브라우저에 출력하는 역할을 한다.

```java
package com.carshop.study;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/study")
public class ModelExam {


    @GetMapping("/modeltest")
    public String modeltest(Model model) {
        model.addAttribute("data1", "모델 테스트입니다.");
        model.addAttribute("data2", "url 요청은 modeltest 입니다.");
        return "study/modeltest";
    }




}

```

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<meta charset="utf-8"><%@ taglib uri="http://java.sun.com/jsp/jstl/core"
    prefix="c"%>
<%@ page session="false"%>
<html>
<head>
<title>modeltest</title>
</head>
<body>
    <h1>${data1 }</h1>
    <h1>${data2 }</h1>
</body>
</html>

```



▼ **ModelMap[데이터(또는 객체) 정보를 저장]**

ModelMap 은 Model 과 완전히 동일하다.

따라서 Model 을 사용하자.

```java
package com.carshop.study;

import org.springframework.stereotype.Controller;

@Controller
@RequestMapping("/study")
public class ModelMapExam {


    @GetMapping("/modelmaptest")
    public String modelmap(ModelMap modelMap) {
        modelMap.addAttribute("data1", "모델 맵 테스트입니다.");
        modelMap.addAttribute("data2", "모델과 완전히 동일합니다.");
        return "study/modelmaptest";
    }



}
```

▼ **ModelAndView[모델 정보 및 뷰 정보를 저장]**

위의 두가지 방법 과는 달리 ModelAndView 는 객체 형태로 리턴한다.

따라서 뷰도 속성 형태로 설정하여 함께 리턴하게 된다.

```java
package com.carshop.study;


import java.util.List;

@Controller
@RequestMapping("/study")
public class ModelAndViewExam {

    private BoardService boardService;

    @GetMapping("/modelandview")
    public ModelAndView modelandview() {
        ModelAndView modelAndView = new ModelAndView();
        List<BoardDTO> list = boardService.getAllBoardList();
        modelAndView.addObject("test", list);
        modelAndView.setViewName("study/modelandview");
        return modelAndView;
    }

}

```

▼ 경로 변수와 @PathVariable

웹 요청 url 에 포함된 파라미터 값을 전달 받는 경로 변수와 이를 처리하는 @PathVariable 어노테이션에 대해 알아보자.

```java
package com.carshop.controller;

import java.util.List;

@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";
    }

    @GetMapping("/cars/{ccate}")
    public String requestCarsByCategory(@PathVariable("ccate") String carCategory, Model model) {
        List<CarDTO> carsByCategory = carService.getCarListByCategory(carCategory);
        model.addAttribute("carList", carsByCategory);
        return "cars";
    }

}

```

▼ Spring 한글 인코딩 설정.

web.xml 하단에 추가.

CarController.java web.xml

```
            <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>appServlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

    <filter>
        <filter-name>encodingFilter</filter-name>
        <filter-class>org.springframework.web.filter.CharacterEncodingFilter
        </filter-class>
        <init-param>
            <param-name>encoding</param-name>
            <param-value>UTF-8</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>encodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

</web-app>
```

```
<filter>
    <filter-name>encodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter
    </filter-class>
    <init-param>
      <param-name>encoding</param-name>
      <param-value>UTF-8</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>encodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
```

▼ Python

▼ Homework Review

```
from selenium import webdriver      # 웹 브라우저 컨트롤(크롬)
from bs4 import BeautifulSoup as bs # 데이터 분석을 용이하게 정제
import pandas as pd

driver = webdriver.Chrome('chromedriver.exe')   # 버전 주의
url = 'https://www.genie.co.kr/chart/top200'
driver.get(url)

txt = driver.page_source    # 이 때 읽어온 데이터는 그냥 글자
html = bs(txt)

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

song_data = []
rank = 1

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

for song in songs:
```

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

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

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

txt = driver.page_source    # 이 때 읽어온 데이터는 그냥 글자
html = bs(txt)

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

for song in songs:
    title = song.select('a.title')[0].text.strip()
    singer = song.select('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)
```

▼ Naver Webtoon Title

https://comic.naver.com/webtoon/weekday

```
##########################################
#           웹툰 타이틀 읽어오기           #
##########################################


from selenium import webdriver      # 웹 브라우저 컨트롤(크롬)
from bs4 import BeautifulSoup as bs # 데이터 분석을 용이하게 정제
import pandas as pd

driver = webdriver.Chrome('chromedriver.exe')   # 버전 주의
url = 'https://comic.naver.com/webtoon/weekday'
driver.get(url)

txt = driver.page_source    # 이 때 읽어온 데이터는 그냥 글자
html = bs(txt)

titles = html.select('a.title')
title_data = []

for title in titles:
    subject = title.text

    title_data.append(['웹툰', subject])

pprint(title_data)

df = pd.DataFrame(title_data, columns= ['웹툰', '제목'])
df

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

▼ Naver Webtoon Thumbnail

https://comic.naver.com/webtoon/weekday

```
##########################################
#       썸 네일 모두 자동으로 다운받기       #
```

```
##########################################


import requests
from bs4 import BeautifulSoup as bs # 데이터 분석을 용이하게 정제
import pandas as pd
from pprint import pprint

url = 'https://comic.naver.com/webtoon/weekday'

txt = requests.get(url)    # 이 때 읽어온 데이터는 그냥 글자
html = bs(txt.text, 'html.parser')

tlist = html.findAll('div', {'class':'col_inner'})

pprint(tlist)
```

▼ from pprint import pprint

```
from pprint import pprint

titles = html.findAll('a', {'class' : 'title'})
titles = [title.text for title in titles]
pprint(titles)
```

▼ 네이버 날씨

```
from selenium import webdriver      # 웹 브라우저 컨트롤(크롬)
from bs4 import BeautifulSoup as bs # 데이터 분석을 용이하게 정제
import pandas as pd

driver = webdriver.Chrome('chromedriver.exe')   # 버전 주의
url = 'https://weather.naver.com/'
driver.get(url)

txt = driver.page_source    # 이 때 읽어온 데이터는 그냥 글자
html = bs(txt)

temp = html.select('strong.current')[0].text.strip()
pprint(temp[5:])
```

▼ Bitcoin Price

```
from selenium import webdriver      # 웹 브라우저 컨트롤(크롬)
from bs4 import BeautifulSoup as bs # 데이터 분석을 용이하게 정제
import pandas as pd
import time

driver = webdriver.Chrome('chromedriver.exe')   # 버전 주의
url = 'https://www.bithumb.com/react/'
driver.get(url)

time.sleep(3)   # 3초간 대기

txt = driver.page_source    # 이 때 읽어온 데이터는 그냥 글자
html = bs(txt)

coins = html.select('strong.MarketRow_sort-real__5zeND')
coin = coins[0]

coins = [coin.text for coin in coins]

coins[1]
```

▼ 정규 표현식

```
from urllib.request import urlretrieve
import re
li_list = []

for data in tlist:
    # 제목과 썸네일 영역 추출
    li_list.extend(data.findAll('li')) # 해당 부분을 찾아 병합

pprint(li_list)

# 썸네일과 제목 추출
for li in li_list:
    img = li.find('img')
    title = img['title']
    img_src = img['src']
    # print(title, img_src)
    # 특수 문자 제거 정규 표현식
    title = re.sub('[^0-9a-zA-Zㄱ-힣]', '', title)
    urlretrieve(img_src, title + '.jpg')
```

▼ 파이썬으로 유튜브 검색하기

```
!pip install webdriver_manager

# 파이썬으로 브라우저 크롬 컨트롤
from selenium import webdriver      # 웹 브라우저 컨트롤(크롬)
from bs4 import BeautifulSoup as bs # 데이터 분석을 용이하게 정제
import pandas as pd
from pprint import pprint
import time

from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.keys import Keys

driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
url = 'https://www.youtube.com'
driver.get(url)

time.sleep(3)   # 3초간 대기

# 검색어 창을 찾자

search = driver.find_element(by=By.XPATH, value='//input[@id="search"]')

# 찾은 검색어 창에 검색어 입력
search.send_keys('chat gpt')
search.send_keys(Keys.ENTER)
```