

Spring Day 3

▼ Python

Stock Crawling

```
#####
#               네이버 증권 읽어오기               #
#####

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://finance.naver.com/item/main.nhn?code=005930'
driver.get(url)

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

time.sleep(5)

titles = html.select('p.no_today > em.no_up')

print(titles[0].text)
```

Stock API

```
# !pip install pykrx # 추가 조회 모듈

from pykrx import stock
# stock.get_market_ohlcv('20230328', '20230228', '005930')
stock.get_market_ohlcv('20230302', '20230302', '005930')

# !pip install yfinance

import yfinance as yf
yf.download('TSLA', start='2023-03-01')
```

Coin Crawling

```
import requests

url = "https://api.bithumb.com/public/ticker/ALL_KRW"

headers = {"accept": "application/json"}

response = requests.get(url, headers=headers)

print(response.text)

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 읽어들인 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[0]

```

Coin API

```

#####
#           Bithumb API           #
#####

# !pip install pybithumb

# 빗썸에서 제공하는 API 설치 (최초 한번만 설치)

import pybithumb as pb

print(pb.get_tickers())
# tickers는 빗썸에서 거래 가능한 종목명을 출력

print(len(pb.get_tickers()))

# 비트코인 가격 조회

print(pb.get_current_price("BTC"))

# 비트코인 가격 반복 조회

while True:
    print(pb.get_current_price("BTC"))
    time.sleep(3)

# 모든 코인 가격 조회

tickers = pb.get_tickers()

for ticker in tickers:
    print(ticker, pb.get_current_price(ticker))
    time.sleep(1)

# 자세한 코인 가격 조회

print(pb.get_market_detail("BTC"))

# ohlcv
# 시가 / 고가 / 저가 / 종가(현재가) / 거래량
# 튜플로 리턴한다.

# 모든 자세한 코인 가격 조회

# 네트워크를 이용하는 경우에는 특히 예외. 즉, 예외가 발생할 경우가 상당히 많다.
# 따라서 1) time.sleep() 을 이용한 시간 지연
#         2) try except 을 이용한 예외 처리가 중요하다.

tickers = pb.get_tickers()

try :
    for ticker in tickers:
        print(ticker, pb.get_market_detail(ticker))
        time.sleep(1)
except :
    print("예외가 발생 되었습니다.")

# 상승장 하락장 알람 프로그램

# 이동평균선 사용 : 이동평균 값보다 현재가가 높다면 상승장

# 이동 평균 값보다 현재가가 낮다면 하락장

# 03/01 1000

```

```

# 03/02 1100
# 03/03 1200
# 03/04 1100
# 03/05 1000

# 03/06 1400    상승장
# 03/06 1000    하락장

# 받아온 데이터는 기본적으로 데이터프레임이다.
# 데이터 프레임에서 한개의 열. 즉, Series 이다.

btc = pb.get_ohlcv("BTC")
close = btc['close']
print(close)

# 수동으로 이동평균 계산

btc = pb.get_ohlcv("BTC")
close = btc['close']

print((close[0] + close[1] + close[2] + close[3] + close[4]) / 5)
print((close[1] + close[2] + close[3] + close[4] + close[5]) / 5)
print((close[2] + close[3] + close[4] + close[5] + close[6]) / 5)

# 자동으로 이동평균 계산

btc = pb.get_ohlcv("BTC")
close = btc['close']

roll5 = close.rolling(5)
mean5 = roll5.mean()
print(mean5)

# 5일간의 평균을 계산하기 때문에 최초 4일은 계산값이 없이 NaN으로 표시.
# 최신 5일간의 평균은 그날의 현재가가 바뀌기 때문에 다음날이 되기 전까지 지속적으로 변동한다.

# 상승장 하락장 파악

btc = pb.get_ohlcv("BTC")
close = btc['close']

roll5 = close.rolling(5)
mean5 = roll5.mean()
last_mean5 = mean5[-2]

# 비트코인 현재값
price = pb.get_current_price("BTC")

# 비교

if price > last_mean5:
    print('상승장')
else:
    print('하락장')

```

스타벅스 위치 가져오기

```

#####
#           스타벅스 위치           #
#####

import requests
from selenium import webdriver      # 웹 브라우저 컨트롤(크롬)
from bs4 import BeautifulSoup as bs # 데이터 분석을 용이하게 정제
import pandas as pd
import time
from selenium.webdriver.common.by import By

driver = webdriver.Chrome('chromedriver.exe') # 버전 주의
url = 'https://www.starbucks.co.kr/store/store_map.do'
driver.get(url)

time.sleep(3) # 3초간 대기

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

# 파이썬으로 브라우저에서 단추 클릭하기

```

```

area_btn = '#container > div > form > fieldset > div > section > article.find_store_cont > article > header.loca_search > h3 > a'
driver.find_element(By.CSS_SELECTOR , area_btn).click()

time.sleep(2)

seoul_btn = '#container > div > form > fieldset > div > section > article.find_store_cont > article > article:nth-child(4) > div.loca'
driver.find_element(By.CSS_SELECTOR , seoul_btn).click()

time.sleep(2)

all_btn = '#mCSB_2_container > ul > li:nth-child(1) > a'
driver.find_element(By.CSS_SELECTOR , all_btn).click()

time.sleep(3)

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

time.sleep(3)

from pprint import pprint

shops = html.select('ul.quickSearchResultBoxSidoGugun > li')
print(shops[0])

# area_btn = driver.find_element(By.CLASS_NAME , "loca_search")
# area_btn.click()
# time.sleep(3)

# seoul_btn = driver.find_element(By.CLASS_NAME , "sido_arae_box")
# li = seoul_btn.find_elements(By.TAG_NAME , "li")
# li[0].click()
# time.sleep(3)

# all_btn = driver.find_element(By.CLASS_NAME , "gugun_arae_box")
# li = all_btn.find_elements(By.TAG_NAME , "li")
# li[0].click()
# time.sleep(3)

```

▼ etc

QtDesigner 위치 : C:\Users\user\anaconda3\Library\bin

▼ Spring

@PathVariable - 1 Variable

```

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.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/vari")
public class VariableController {

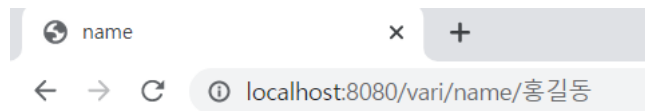
    @GetMapping("/name/{name}")
    public String namesMethod(@PathVariable String name, Model model) {
        model.addAttribute("data", "성명 : " + name);

        return "/study/name";
    }
}

```

```
}
```

```
<%@ 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>name</title>
</head>
<body>
<h1>${data }</h1>
</body>
</html>
```



성명 : 홍길동

@PathVariable - 2 Variables or More

```
package com.carshop.study;

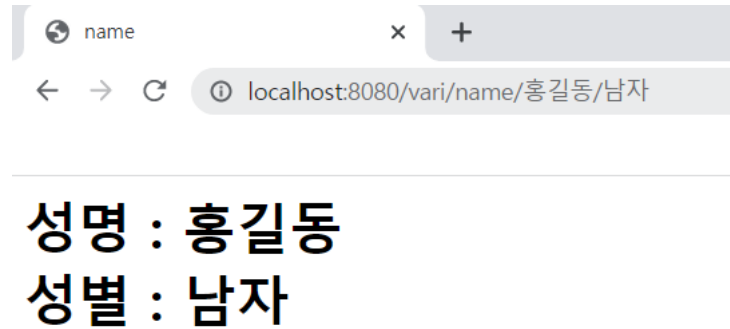
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/vari")
public class VariableController {

    @GetMapping("/name/{name}/{gender}")
    public String namesMethod2(
        @PathVariable String name,
        @PathVariable String gender,
        Model model) {
        model.addAttribute("data", "성명 : " + name + "<br>" + "성별 : " + gender);

        return "/study/name";
    }
}
```

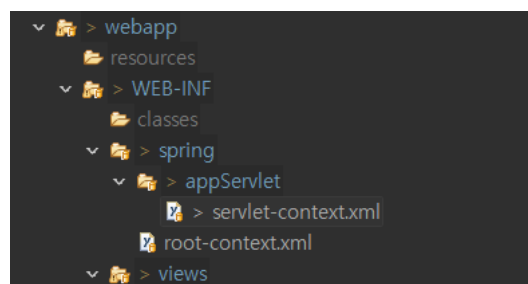
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<meta charset="utf-8"><%@ taglib uri="http://java.sun.com/jsp/jstl/core"
    prefix="c"%>
<%@ page session="false"%>
<html>
<head>
<title>name</title>
</head>
<body>
    <h1>${data }</h1>
</body>
</html>
```



@MatrixVariable



MatrixVariable을 사용하기 위해서는 반드시 사전에 설정을 먼저 해주어야 한다.



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans:beans xmlns="http://www.springframework.org/schema/mvc"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xmlns:beans="http://www.springframework.org/schema/beans"
5     xmlns:context="http://www.springframework.org/schema/context"
6     xsi:schemaLocation="http://www.springframework.org/schema/mvc https://www.springf
7     http://www.springframework.org/schema/beans https://www.springframework.org/s
8     http://www.springframework.org/schema/context https://www.springframework.org,
9
10    <!-- DispatcherServlet Context: defines this servlet's request-processing infrastr
11
12    <!-- Enables the Spring MVC @Controller programming model -->
13    <annotation-driven enable-matrix-variables="true" />
14
15    <!-- Handles HTTP GET requests for /resources/** by efficiently serving up static
16    <resources mapping="/resources/**" location="/resources/" />
17
18    <!-- Resolves views selected for rendering by @Controllers to .jsp resources in t
19    <beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResol
20        <beans:property name="prefix" value="/WEB-INF/views/" />
21        <beans:property name="suffix" value=".jsp" />
22    </beans:bean>
23
24    <context:component-scan base-package="com.carshop.*" />
25
26

```

```

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.MatrixVariable;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;

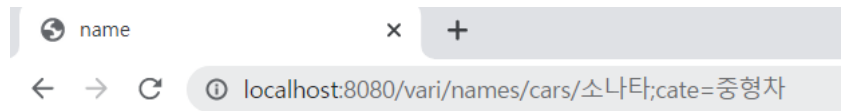
@Controller
@RequestMapping("/vari")
public class VariableController {

    @GetMapping("/names/cars/{carId}")
    public String namesMethod3(
        @PathVariable String carId,
        @MatrixVariable String cate,
        Model model) {
        model.addAttribute("data", "carId : " + carId + "<br>" + "category : " + cate);

        return "/study/name";
    }
}

```

localhost:8080/vari/names/cars/소나타;cate=중형차



carId : 소나타
category : 중형차

@RequestParam

가장 많이 사용하는 방식이다.

```
package com.carshop.study;

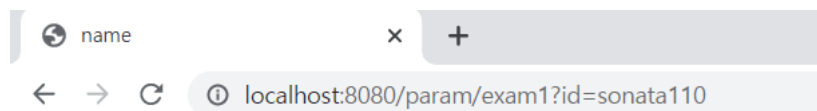
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.MatrixVariable;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;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

@Controller
@RequestMapping("/param")
public class ParamController {

    @GetMapping("/exam1")
    public String exam1(@RequestParam String id, Model model) {
        model.addAttribute("data", "자동차 아이디 : " + id);

        return "/study/name";
    }

}
```



자동차 아이디 : sonata110

@ModelAttribute

ModelAttribute를 설정하여 Model 속성을 만들어두면

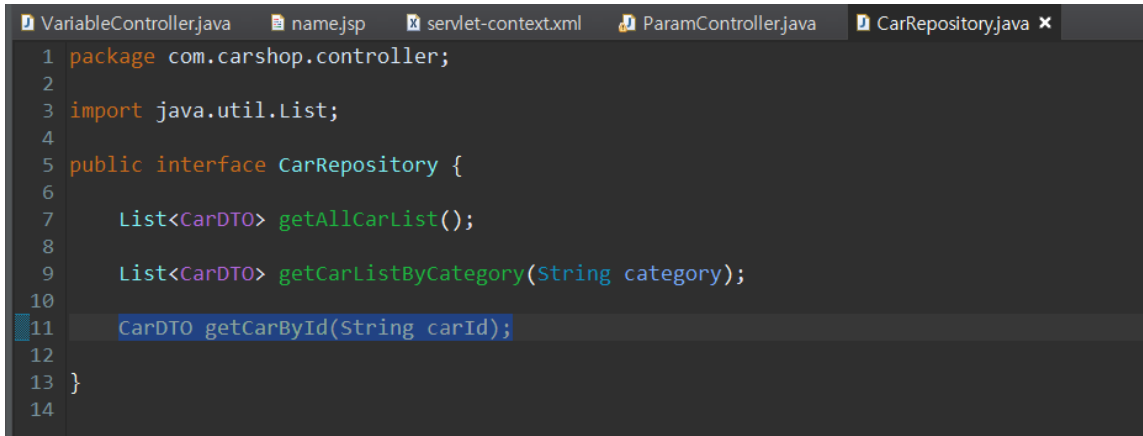
다른 매서드에서 Model객체를 사용할 때 ModelAttribute에서 설정한 속성을 사용할 수 있다.

```
@ModelAttribute
public void addAttributes(Model model) {
    model.addAttribute("addTitle", "신규 차량 등록");
}
```

▼ 일치하는 정보를 찾아서 출력하기(실습)

1. Repository 설정

carId 값으로 조회하기 위해 설정(CarRepository.java)



```
1 package com.carshop.controller;
2
3 import java.util.List;
4
5 public interface CarRepository {
6
7     List<CarDTO> getAllCarList();
8
9     List<CarDTO> getCarListByCategory(String category);
10
11     CarDTO getCarById(String carId);
12
13 }
14
```

CarRepositoryImpl.java

```
@Override
public CarDTO getCarById(String carId) {

    CarDTO carInfo = null;

    for(int i = 0; i < listOfCars.size(); i++) {
        CarDTO carDTO = listOfCars.get(i);
        if(carDTO != null && carDTO.getCid() != null && carDTO.getCid().equals(carId)) {
            carInfo = carDTO;
        }
    }

    if(carInfo == null) {
        throw new IllegalArgumentException("자동차 ID 가 " + carId + "인 자동차는 없습니다.");
    }

    return carInfo;
}
```

2. Service 설정

CarService.java

```
1 package com.carshop.controller;
2
3 import java.util.*;
4
5 public interface CarService {
6
7     List<CarDTO> getAllCarList();
8
9     List<CarDTO> getCarListByCategory(String carCategory);
10
11     CarDTO getCarById(String carId);
12
13 }
14
```

CarServiceImpl.java

```
@Override
public CarDTO getCarById(String carId) {

    CarDTO carById = carRepository.getCarById(carId);

    return carById;
}
```

3. Controller & JSP 설정

CarController.java

```
@GetMapping("/car")
public String requestCarById(@RequestParam("id") String carId, Model model) {
    CarDTO carById = carService.getCarById(carId);
    model.addAttribute("car", carById);
    return "car";
}
```

cars.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>cars</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-rbs
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KFdBie4V
</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-controls="
<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>차량 보기(김도영)</h1></div>
</div>

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

    <c:forEach items="${carList}" var="car">
      <div class="col-md-4">
        <h3>${car.cid}</h3>
        <p>${car.cname}</p>
        <p>${car.cprice}만원</p>
        <p><a href="/car?id=${car.cid}" class="btn btn-Secondary" role="button">상세보기</a></p>
        <p><a href='<c:url value="/car?id=${car.cid }"/>' class="btn btn-Secondary" role="button"> -->

      </div>

    </c:forEach>

  </div>
</div>

</body>
</html>

```

car.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>cars</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-rbs
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KfDBIe4zV

</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-controls="
      <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>
    </div>
  </div>
</nav>

```

```

        </li>
    </ul>
    <form role="search">
        <input class="form-control" type="search" placeholder="Search" aria-label="Search">
    </form>
</div>
</nav>
<div class="alert alert-dark" role="alert">
<div class="container"><h1>차량 상세보기</h1></div>
</div>

<div class="container">
    <div class="row" align="center">
        <h3>${car.cid}</h3>
        <p>${car.cname}</p>
        <p>${car.cprice}만원</p>
        <p>${car.ccate}</p>
        <p>${car.cdetc}</p>
    </div>
</div>

</body>
</html>

```

▼ 자동차 추가하기(실습)

1. Repository 설정

CarRepository.java

```

1 package com.carshop.controller;
2
3 import java.util.List;
4
5 public interface CarRepository {
6
7     List<CarDTO> getAllCarList();
8
9     List<CarDTO> getCarListByCategory(String category);
10
11     CarDTO getCarById(String carId);
12
13     void setNewCar(CarDTO car);
14
15 }
16

```

CarRepositoryImpl.java

```

@Override
public void setNewCar(CarDTO car) {
    listofCars.add(car);
}

```

2. Service 설정

CarService.java

```
1 package com.carshop.controller;
2
3 import java.util.*;
4
5 public interface CarService {
6
7     List<CarDTO> getAllCarList();
8
9     List<CarDTO> getCarListByCategory(String carCategory);
10
11     CarDTO getCarById(String carId);
12
13     void setNewCar(CarDTO car);
14
15 }
16
```

CarServiceImpl.java

```
@Override
public void setNewCar(CarDTO car) {
    carRepository.setNewCar(car);
}
```

3. Controller & JSP 설정

CarController.java

```
@GetMapping("/add")
public String requestAddCarForm(@ModelAttribute("NewCar") CarDTO car) {
    return "addCar";
}

@PostMapping("/add")
public String submitAddNewCar(@ModelAttribute("NewCar") CarDTO car) {
    carService.setNewCar(car);

    return "redirect:/cars";
}

@ModelAttribute
public void addAttributes(Model model) {
    model.addAttribute("addTitle", "신규 차량 등록");
}
```



form 의 modelAttribute 속성값은 사용자 요청 URL 에 매칭되는 Controller 안에 요청 처리 메서드의 매개변수로 선언한 객체이름과 일치해야 한다.

```

@GetMapping("/add")
public String requestAddCarForm(@ModelAttribute("NewCar") CarDTO car) {
    return "addCar";
}

@PostMapping("/add")
public String submitAddNewCar(@ModelAttribute("NewCar") CarDTO car) {
    carService.setNewCar(car);

    return "redirect:/cars";
}

```

```

<div class="container"><h1>차량 등록</h1>

    <form:form modelAttribute="NewCar" class="form-horizontal">
        <fieldset>
            <legend>
                ${addTitle }
            </legend>
            자동차 ID :
            <form:input path="cid" class="form-control" />
            자동차 이름 :
            <form:input path="cname" class="form-control" />
            자동차 가격 :
            <form:input path="cprice" class="form-control" />

```

addCar.jsp

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
<!DOCTYPE html>
<html>
<head>
<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-rbsA2VE"
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KFdBIE4zVF0s6"
<meta charset="UTF-8">
<title>자동차 등록</title>
</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-controls="
            <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>차량 등록</h1>

    <form:form modelAttribute="NewCar" class="form-horizontal">
        <fieldset>

```

```

<legend>
${addTitle }
</legend>
자동차 ID :
<form:input path="cid" class="form-control" />
자동차 이름 :
<form:input path="cname" class="form-control" />
자동차 가격 :
<form:input path="cprice" class="form-control" />
자동차 카테고리 :
<form:input path="ccate" class="form-control" />
자동차 소개 :
<form:textarea path="cdesc" cols="50" rows="2" class="form-control" />

<input type="submit" class="btn btn-primary" value="등록">

</fieldset>
</form:form>

</div>
</div>

</body>
</html>

```

▼ etc

root-context.xml : 프론트엔드 처리.

servlet-context.xml : 백엔드 처리.

web.xml : 톰캣 처리.

pom.xml : 의존성 처리.

Session : 계속 기억할 때.

Application : 전체 페이지(프로젝트 전체에서 사용)

PageContext : 한 페이지에서만 사용.

Request : 요청할 때(페이지 2개)