Spring Day 5

▼ Spring

File Upload

1. **pom.xml 의존성 추가**
   - commons-fileupload
   - commons-io

```
    <!-- https://mvnrepository.com/artifact/commons-fileupload/commons-fileupload -->
    <dependency>
        <groupId>commons-fileupload</groupId>
        <artifactId>commons-fileupload</artifactId>
        <version>1.5</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/commons-io/commons-io -->
    <dependency>
        <groupId>commons-io</groupId>
        <artifactId>commons-io</artifactId>
        <version>2.11.0</version>
    </dependency>
```

2. **servlet-context.xml 설정**

```
14
15      <!-- Handles HTTP GET requests for /resources/** by efficiently serving up static resources in the ${webappRoot}/r
16      <resources mapping="/resources/**" location="/resources/" />
17
18      <!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views directory -->
19      <beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
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      <beans:bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
27              <beans:property name="maxUploadSize"        value="10240000"/>
28      </beans:bean>
29
30  </beans:beans>
31
```

3. **JSP & DTO 설정**

CarDTO.java

```
package com.carshop.controller;

import org.springframework.web.multipart.MultipartFile;

public class CarDTO {

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

```
  private MultipartFile carimage;
  public String getCid() {
    return cid;
  }
  public void setCid(String cid) {
    this.cid = cid;
  }
  public String getCname() {
    return cname;
  }
  public void setCname(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 MultipartFile getCarimage() {
    return carimage;
  }
  public void setCarimage(MultipartFile carimage) {
    this.carimage = carimage;
  }

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

  public CarDTO() {
    super();
  }



}
```

addCar.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>

<head>
  <title>자동차 등록</title>
  <%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
      <meta charset="utf-8">

</head>
<body class="text-center">
<%@ include file="header.jsp" %>
<div class="alert alert-dark" role="alert">
<div class="container"><h1>차량 등록</h1>

      <form:form modelAttribute="NewCar" action="/admin/add?${_csrf.parameterName}=${_csrf.token}" class="form-horizontal" enctype=
        <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" />
          자동차 사진 :
          <form:input path="carimage" type="file" class="form-control" />

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


        </fieldset>
      </form:form>







    </div>
</div>







</body>
</html>
```

cars.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<head>
  <title>cars</title>
  <%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
      <meta charset="utf-8">

</head>
<body class="text-center">
<%@ include file="header.jsp" %>
<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">



        <img src='<c:url value="C:\\upload\\${car.getCarimage().getOriginalFilename()}"/>'/>
        <h3>${car.cid}</h3>
        <p>${car.cname}
        <p>${car.cprice}만원
        <p><a href="/car?id=${car.cid }" class="btn btn-Secondary" role="button">상세보기</a></p>
      <%-- <a href='<c:url value="/car?id=${car.cid }"/>' class="btn btn-Secondary" role="button"> --%>

        </div>


      </c:forEach>

    </div>
  </div>




</body>
</html>
```

4. **Controller**

> 💡 사진이 저장되는 위치를 윈도우만 고려하지 말고 추후 업로드 할 Cafe24의
> 운영체제인 리눅스를 고려하여 정하자.

CarController.java

```
package com.carshop.controller;

import java.io.File;
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.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;

@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";
  }

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

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

  @PostMapping("/admin/add")
  public String submitAddNewCar(@ModelAttribute("NewCar") CarDTO car) {

    MultipartFile carimage = car.getCarimage();

    String saveName = carimage.getOriginalFilename();
//    File saveFile = new File("/resources/images");
    File saveFile = new File("C:\\upload", saveName);

    if (carimage != null && !carimage.isEmpty()) {
      try {
        carimage.transferTo(saveFile);
      } catch (Exception e) {
        throw new RuntimeException("차량 이미지 업로드가 실패했습니다.");
      }
    }


    carService.setNewCar(car);
    return "redirect:/cars";
  }
```

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

  @GetMapping("/login")
  public String loginMethod() {
    return "login";
  }

  @GetMapping("//loginfailed")
  public String loginfailedMethod() {
    return "login";
  }

  @GetMapping("/logout")
  public String logoutMethod() {
    return "login";
  }

}
```

▼ Python

파이썬에서 DB 연동시키기

```
# !pip install pymysql
# !pip install requests
# !pip install PyQt5


import pymysql
import requests

conn = pymysql.connect(host='creatordodo.shop', user='difbfl4751', password='비밀번호', db='difbfl4751', charset='utf8')

cur = conn.cursor()

cur.execute("SELECT * FROM user")
result=cur.fetchall()

import pprint
pprint.pprint(result)
```

PyQt

파이썬 기반의 GUI(Graphic User Interface) 라이브러리.

riverbankcomputing 이라는 회사에서 개발 원래는 C 언어용으로 개발되었으나 파이썬 용으로 전환되었다. 현재 버전 6도 출시되었다.

▼ DB 연동 시스템 종류

TKinter

1. 파이썬 기본 GUI
2. 기본으로 설치되어 있는 시스템

3. 디자인이 예쁘지는 않다.

wxPython

PyQt

1. 아나콘다를 설치하면 기본으로 같이 설치해준다.
2. 디자인이 예쁘다.

윈도우 기본창 띄우기

```
# 윈도우 기본창 띄우기


# 필요한 모듈 읽어오기 및 기본적인 창 구성요소를 제공하는 모든 것은 PyQt5.QtWidgets 에 만들어져 있다.

import sys
from PyQt5.QtWidgets import QApplication, QWidget

class MyApp(QWidget):
    # 생성자 설정
    def __init__(self):
        super().__init__()
        self.initUI()

    # 창 기본 설정, 여기서는 제목만 설정하였다.
    def initUI(self):
        self.setWindowTitle("Hello World!!!")
        self.show()

# 파이썬은 기본적으로 인터프리터이기 때문에 한줄한줄 코드를 실행하고 마치게 된다.
# 열려진 윈도우 창을 계속 유지시켜야 한다.
app  = QApplication(sys.argv)
ex = MyApp()
sys.exit(app.exec_())
```

정지 단추 만들기

```
# 정지 단추 만들기.

import sys
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton
from PyQt5.QtCore import QCoreApplication

class MyApp(QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        self.setWindowTitle("Hello World!!!")
        btn = QPushButton('Quit', self)
        btn.clicked.connect(QCoreApplication.instance().quit)

        self.show()

app  = QApplication(sys.argv)
ex = MyApp()
sys.exit(app.exec_())
```

윈도우 메뉴 생성 & 동작

```
# 윈도우 메뉴 생성 & 동작

import sys
from PyQt5.QtWidgets import QApplication, QMainWindow, QAction, qApp
```

```
from PyQt5.QtGui import QIcon

class MyApp(QMainWindow):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        exitAction = QAction(QIcon('exit.png'), 'Exit', self)
        exitAction.setShortcut('Ctrl+Q')
        exitAction.setStatusTip('Exit Application')
        exitAction.triggered.connect(qApp.quit)

        self.statusBar()

        # 한개의 메뉴바 생성. file 메뉴, 단추를 클릭하면 종료하는 기능, 단축키까지 설정
        menubar = self.menuBar()
        menubar.setNativeMenuBar(False)
        filemenu = menubar.addMenu('&File')
        filemenu.addAction(exitAction)

        self.setWindowTitle('Menubar')
        self.setGeometry(300,300,300,200)
        self.show()

app  = QApplication(sys.argv)
ex = MyApp()
sys.exit(app.exec_())
```

QT Designer

▼ UI 첨부파일

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/2a5888ac-33db-45b1-9cc7-25aae0f3b3c6/Untitled.txt

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/c1e6a5b6-2fd5-4e01-a776-1b4da165b49f/Untitled.txt

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/8fd9ee1d-e4ad-40e0-9ca4-d9853113f40d/Untitled.txt

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/564de5c7-d656-4356-97d1-64e0b35b4eab/Untitled.txt

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/cdcd8d81-4334-4104-806c-50ddabe9537c/Untitled.txt

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/7b7318ed-d944-4995-b141-74f55c852399/Untitled.txt

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/ff57807d-cae6-402d-afe9-b7a6f1bb4951/Untitled.txt

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/b26cd4ac-d299-4424-925e-c1901b3023a7/Untitled.txt

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/f1ad73a2-941c-4b88-84e9-424657c0540c/Untitled.txt

아나콘다를 설치하면 기본으로 같이 설치해준다.

UI Connect

```
# QT Designer 로 생성한 윈도우 창을 여는 방법

# 윈도우 기본창 띄우기

import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic

from_class = uic.loadUiType("mywindow.ui")[0]

class MyWindow(QMainWindow, from_class):

    def __init__(self):
        super().__init__()
        self.setupUi(self)

app  = QApplication(sys.argv)
window = MyWindow()
window.show()
app.exec_()
```

PushButton

```
# QT Designer 로 생성한 윈도우 창을 여는 방법

# 윈도우 기본창 띄우기

# 클릭 1개 인식

import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic

from_class = uic.loadUiType("mywindow1.ui")[0]

class MyWindow(QMainWindow, from_class):

    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.pushButton.clicked.connect(self.btn_clicked)

    def btn_clicked(self):
        print("클릭 확인")
```

```
app  = QApplication(sys.argv)
window = MyWindow()
window.show()
app.exec_()
```

```
# QT Designer 로 생성한 윈도우 창을 여는 방법

# 윈도우 기본창 띄우기

# 클릭 2개 인식

import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic

from_class = uic.loadUiType("mywindow2.ui")[0]

class MyWindow(QMainWindow, from_class):

    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.pushButton_1.clicked.connect(self.btn1_clicked)
        self.pushButton_2.clicked.connect(self.btn2_clicked)

    def btn1_clicked(self):
        print("클릭 1 확인")

    def btn2_clicked(self):
        print("클릭 2 확인")




app  = QApplication(sys.argv)
window = MyWindow()
window.show()
app.exec_()
```

RadioButton

```
# 라디오 버튼 인식
# Radio Button은 여러개 중 하나만 선택 가능하므로 처리가 쉽다.

import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic

from_class = uic.loadUiType("mywindow3.ui")[0]

class MyWindow(QMainWindow, from_class):

    def __init__(self):
        super().__init__()
        self.setupUi(self)

        self.radioButton_1.clicked.connect(self.radioFunction)
        self.radioButton_2.clicked.connect(self.radioFunction)
        self.radioButton_3.clicked.connect(self.radioFunction)

    def radioFunction(self):
        if self.radioButton_1.isChecked() :
            print("radio button 1 Checked")
        elif self.radioButton_2.isChecked() :
            print("radio button 2 Checked")
        elif self.radioButton_3.isChecked() :
            print("radio button 3 Checked")




app  = QApplication(sys.argv)
window = MyWindow()
window.show()
app.exec_()
```

CheckBox

```
# Check Box
# 체크박스는 중복선택이 가능함에 따라 코드가 조금 복잡해진다.

import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic

from_class = uic.loadUiType("mywindow4.ui")[0]

class MyWindow(QMainWindow, from_class):

    def __init__(self):
        super().__init__()
        self.setupUi(self)

        self.check1.stateChanged.connect(self.chkFunction)
        self.check2.stateChanged.connect(self.chkFunction)
        self.check3.stateChanged.connect(self.chkFunction)

        self.checkBox1.stateChanged.connect(self.chkBoxFunction)
        self.checkBox2.stateChanged.connect(self.chkBoxFunction)
        self.checkBox3.stateChanged.connect(self.chkBoxFunction)

# 여러개가 동시 선택될 수 있기 때문에 elif 를 사용하지 않는다.
    def chkFunction(self):
        if self.check1.isChecked() :
            print("check1 button Checked")
        if self.check2.isChecked() :
            print("check2 button Checked")
        if self.check3.isChecked() :
            print("check3 button Checked")
        print()

    def chkBoxFunction(self):
        if self.checkBox1.isChecked() :
            print("checkBox1 button Checked")
        if self.checkBox2.isChecked() :
            print("checkBox2 button Checked")
        if self.checkBox3.isChecked() :
            print("checkBox3 button Checked")
        print()



app  = QApplication(sys.argv)
window = MyWindow()
window.show()
app.exec_()
```

Label

```
# 라벨 처리

import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic

from_class = uic.loadUiType("mywindow5.ui")[0]

class MyWindow(QMainWindow, from_class):

    def __init__(self):
        super().__init__()
        self.setupUi(self)

        self.MelonButton.clicked.connect(self.melonFunction)
        self.GenieButton.clicked.connect(self.genieFunction)

    def melonFunction(self) :
        self.label.setText("멜론 차트 조회")

    def genieFunction(self) :
        self.label.setText("지니 차트 조회")
```

```
app  = QApplication(sys.argv)
window = MyWindow()
window.show()
app.exec_()
```

```
# 라벨 처리

import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic

from_class = uic.loadUiType("mywindow5.ui")[0]

class MyWindow(QMainWindow, from_class):

    def __init__(self):
        super().__init__()
        self.setupUi(self)

        self.MelonButton.clicked.connect(self.melonFunction)
        self.GenieButton.clicked.connect(self.genieFunction)
        self.DeleteButton.clicked.connect(self.deleteFunction)
        self.PrintButton.clicked.connect(self.printFunction)

    def melonFunction(self) :
        self.label.setText("멜론 차트 조회")

    def genieFunction(self) :
        self.label.setText("지니 차트 조회")

    def deleteFunction(self) :
        self.label.clear()

    def printFunction(self) :
        print(self.label.text())

app  = QApplication(sys.argv)
window = MyWindow()
window.show()
app.exec_()
```

TextBrowser

Label은 화면상에서는 수정할 수 없는 한 줄 글자를 보여주는 위젯이였다.

그런데 **Label**은 크기를 넘어가면 글자가 잘려서 보이지 않게 되는 것이 특징이다.

이와 달리 **TextBrowser**는 여러줄의 긴 데이터를 보여주기에 적합하다.

```
# Text Browser

import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic

from_class = uic.loadUiType("mywindow7.ui")[0]

class MyWindow(QMainWindow, from_class):

    def __init__(self):
        super().__init__()
        self.setupUi(self)

        self.PrintBrowserButton.clicked.connect(self.printFunction)
        self.SetTextButton.clicked.connect(self.setFunction)
        self.AppendTextButton.clicked.connect(self.appendFunction)
        self.ClearButton.clicked.connect(self.clearFunction)

    def printFunction(self) :
        print(self.textBrowser.toPlainText())

    def setFunction(self) :
        self.textBrowser.setPlainText("텍스트 브라우저 글자 변경 테스트")
```

```
    def appendFunction(self) :
        self.textBrowser.append("텍스트 추가 테스트")

    def clearFunction(self) :
        self.textBrowser.clear()

app  = QApplication(sys.argv)
window = MyWindow()
window.show()
app.exec_()
```

Line Edit

한줄 짜리 글자를 입력받을 수 있는 입력 위젯이다.

```
# Line Edit

import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic

from_class = uic.loadUiType("mywindow8.ui")[0]

class MyWindow(QMainWindow, from_class):

    def __init__(self):
        super().__init__()
        self.setupUi(self)

        self.lineEdit.textChanged.connect(self.lineEditFuntion)
        self.lineEdit.returnPressed.connect(self.returnFunction)
        self.LineEditChangeButton.clicked.connect(self.changeFunction)

    def lineEditFuntion(self) :
        self.Label_Text.setText(self.lineEdit.text())

    def returnFunction(self) :
        print(self.lineEdit.text())

    def changeFunction(self) :
        self.lineEdit.setText("글자 변경")

app  = QApplication(sys.argv)
window = MyWindow()
window.show()
app.exec_()
```