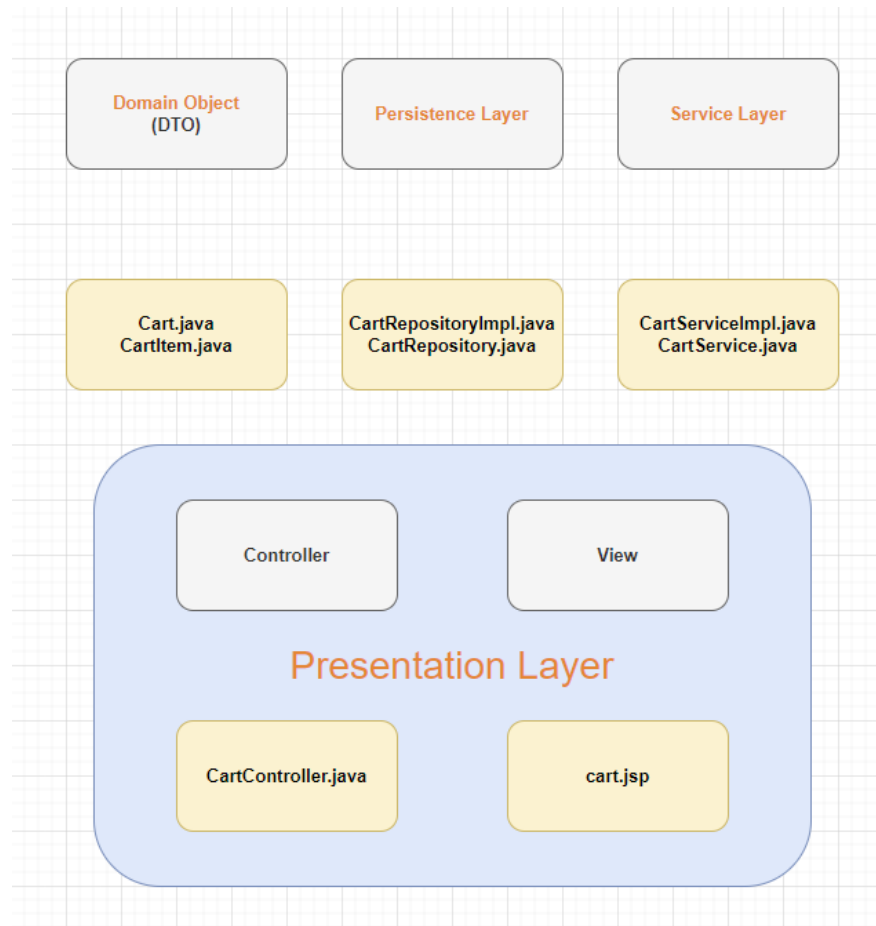


Spring Day 7

▼ Spring



Cart

CartController.java

```
package com.carshop.controller;

import javax.ws.rs.WebParam.Mode;
import javax.servlet.http.HttpServletRequest;

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.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
```

```

@Controller
@RequestMapping("/cart")
public class CartController {

    @Autowired
    private CartService cartService;

    // 경로(/cart) 로 요청이 들어오면 세션id 값을 가져와서 cart/세션값으로 다시 호출
    @GetMapping
    public String requestCartId(HttpServletRequest request) {
        String sessionId = request.getSession(true).getId();
        return "redirect:/cart/" + sessionId;
    }

    // create() 매서드는 장바구니를 새로 생성하고 응답을 body로 전달한다.
    @PostMapping
    public @ResponseBody Cart create(@RequestBody Cart cart) {
        return cartService.create(cart);
    }

    // 요청 URI가 /cart/cartId로 get방식으로 요청되면 처리되는 매서드.
    // 해당 cartId의 모든 정보를 읽어서 model(cart)속성에 등록하고 cart.jsp를 호출한다.
    @GetMapping("/{cartId}")
    public String requestCartList(@PathVariable(value = "cartId") String cartId, Model model) {

        Cart cart =cartService.read(cartId);
        model.addAttribute("cart", cart);
        return "cart";
    }

    // 해당 cart/cartId 값으로 요청하면 해당 장바구니에 등록된 모든 정보 읽어오기
    @PostMapping("/{cartId}")
    public @ResponseBody Cart read(@PathVariable(value = "cartId") String cartId) {

        return cartService.read(cartId);
    }

}

```

car.jsp

```

<p>
    <a href="#" class="btn btn-primary">제품주문 &raquo;</a>
    <a href="c:url value='/cart' />" class="btn btn-warning">장바구니 &raquo;</a>
    <a href="c:url value='/cars' />" class="btn btn-success">제품목록 &raquo;</a>

```

cart.jsp

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Car Detail</title>
<script>
    function removeFromCart(action) {
        document.removeForm.action = action;
        document.removeForm.submit();
        window.location.reload();
    }

    function clearCart() {
        document.clearForm.submit();
    }

```

```

        window.location.reload();
    }
</script>

</head>
<body>
    <%@ include file="header.jsp"%>

    <div class="my-5">
        <div class="alert alert-dark">
            <div class="container">
                <h1>장바구니</h1>
            </div>
        </div>
    </div>

    <div class="container">
        <div class="d-grid gap-2 d-md-flex justify-content-md-end">
            <form:form name="clearForm" method="delete">
                <a href="javascript:clearCart()" class="btn btn-danger pull-left">삭제하기</a>
                <a href="#" class="btn btn-success float-right">주문하기</a>
            </form:form>
        </div>
        <div style="padding-top: 50px">
            <table class="table table-hover">
                <tr>
                    <th>제품</th>
                    <th>가격</th>
                    <th>수량</th>
                    <th>소계</th>
                    <th>비고</th>
                </tr>
                <form:form name="removeForm" method="put">
                    <c:forEach items="${cart.cartItems}" var="item">
                        <tr>
                            <td>${item.value.car.cid}-${item.value.car.cname}</td>
                            <td>${item.value.car.cprice}</td>
                            <td>${item.value.quantity}</td>
                            <td>${item.value.totalPrice}</td>
                            <td><a
                                href="javascript:removeFromCart('../cart/remove/${item.value.car.cid}')"
                                class="btn btn-danger btn-sm">삭제</a></td>
                        </tr>
                    </c:forEach>
                </form:form>
            <tr>
                <th></th>
                <th></th>
                <th>총액</th>
                <th>${cart.grandTotal}</th>
                <th></th>
            </tr>
        </table>

        <a href="<c:url value="/cars" />" class="btn btn-primary">
            &laquo; 쇼핑 계속하기</a>
    </div>
<hr>
<%@ include file="footer.jsp"%>

</div>
</body>
</html>

```

▼ web.xml(HiddenHttpMethodFilter 설정)



브라우저는 기본적으로 (GET, POST)만 지원한다.



(GET, POST) 이외의 HTTP 매서드(PUT, DELETE, PATCH 등)를 사용하기 위해서는
HiddenHttpMethodFilter 설정이 필요하다.

web.xml

```
<filter>
  <filter-name>httpMethodFilter</filter-name>
  <filter-class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>httpMethodFilter</filter-name>
  <servlet-name>appServlet</servlet-name>
</filter-mapping>
```

▼ addCartByNewItem() 추가

선택한 제품을 장바구니에 등록하는 매서드

CartController.java

```
@PostMapping("/add/{cid}")
public void addCartByNewItem(@PathVariable String cid, HttpServletRequest request) {
    //장바구니 ID. 즉, sessionId 가져오기
    String sessionId = request.getSession(true).getId();
    Cart cart = cartService.read(sessionId); //장바구니에 등록되어 있는 모든 정보 가져오기
    if (cart == null) {
        cart = cartService.create(new Cart(sessionId));

        //cid에 대한 정보 가져오기
        CarDTO car = carService.getCarById(cid);
        //cart 아이템으로 해당 제품을 등록
        cart.addCartItem(new CartItem(car));
        //해당 장바구니에 대한 정보 갱신
        cartService.update(sessionId, cart);
    }
}
```

car.jsp

```

        <p>
        <form:form name="addForm" method="put" target="cart">
        <a href="javascript:addToCart('/cart/add/${car.cid}')" class="btn btn-primary">제품주문 &raquo;</a>
        <a href="c:url value='/cart' />" class="btn btn-warning">장바구니 &raquo;</a>
        <a href="c:url value='/cars' />" class="btn btn-success">제품목록 &raquo;</a>
        </form:form>

    </div>
</div>

</body>

<iframe name="cart" style="display: none;"></iframe>

<script type="text/javascript">

//구매하려는 제품을 장바구니로 보낼 때 JS를 이용하여 화면전환 없이 submit()을 실행한다.
function addToCart(action) {
    document.addForm.action = action;
    document.addForm.submit();
    alert("제품이 장바구니에 추가되었습니다.");
}

```

Cart.java

```

public void addCartItem(CartItem item) {
    String cid = item.getCar().getCid(); //등록할 제품 id 가져오기

    //이미 해당 제품이 등록이 되어 있는지 여부 확인
    if (cartItems.containsKey(cid)) {
        CartItem cartItem = cartItems.get(cid); //정보 가져와서
        // 개수만 하나 늘려준다.
        cartItem.setQuantity(cartItem.getQuantity() + item.getQuantity());
        cartItems.put(cid, cartItem); //변경정보 저장
    } else {
        cartItems.put(cid, item); //제품 정보 저장
    }

    updateGrandTotal(); //총액 다시 계산
}

```

CartRepository.java

```

void update(String cartId, Cart cart);

```

CartRepositoryImpl.java

```

@Override
public void update(String cartId, Cart cart) {

    listofCarts.put(cartId, cart);

}

```

CartService.java

```

void update(String cartId, Cart cart);

```

CartServiceImpl.java

```
@Override
public void update(String cartId, Cart cart) {

    CartRepository.update(cartId, cart);

}
```

cart.jsp

CarShop 홈 차량보기 게시판 회원관리 [login](#)

장바구니

[삭제하기](#) [주문하기](#)

제품	가격	수량	소계	비고
c0001-람보르기니	20000	1	20000	삭제
총액			20000	

[← 쇼핑 계속하기](#)

▼ removeCartItem() 추가

선택한 제품을 장바구니에서 삭제하는 매서드

CartController.java

```
@PostMapping("/remove/{cid}")
public void removeCartItem(@PathVariable String cid, HttpServletRequest request) {
    String sessionId = request.getSession(true).getId();
    Cart cart = cartService.read(sessionId); //장바구니에 등록되어 있는 모든 정보 가져오기
    if (cart == null)
        cart = cartService.create(new Cart(sessionId));

    //cid에 대한 정보 가져오기
    CarDTO car = carService.getCarById(cid);

    cart.removeCartItem(new CartItem(car));

    //해당 장바구니에 대한 정보 갱신
    cartService.update(sessionId, cart);
}
```

Cart.java

```
public void removeCartItem(CartItem item) {
    String cid = item.getCar().getCid(); //삭제할 제품 id 가져오기
    cartItems.remove(cid); //해당 제품 삭제
    updateGrandTotal(); //총액 다시 계산
}
```

```
}
```

cart.jsp

장바구니

삭제하기 주문하기

제품	가격	수량	소계	비고
c0002-그랜저	3500	1	3500	삭제
총액			3500	

« 쇼핑 계속하기

장바구니

삭제하기 주문하기

제품	가격	수량	소계	비고
총액			0	

« 쇼핑 계속하기

▼ delete() 추가

장바구니 리스트 비우기

CartController.java

```
@DeleteMapping("/{cartId}")
public void deleteCartList(@PathVariable(value = "cartId") String cartId) {
    cartService.delete(cartId);
}
```

CartRepository.java

```
void delete(String cartId);
```

CartRepositoryImpl.java

```
@Override
public void delete(String cartId) {

    listOfCarts.remove(cartId);

}
```

CartService.java

```
void delete(String cartId);
```

CartServiceImpl.java

```
@Override
public void delete(String cartId) {

    CartRepository.delete(cartId);

}
```

cart.jsp

장바구니

삭제하기 주문하기

제품	가격	수량	소계	비고
c0002-그랜저	3500	1	3500	삭제
총액			3500	

« 쇼핑 계속하기

장바구니

삭제하기 주문하기

제품	가격	수량	소계	비고
총액			0	

« 쇼핑 계속하기

▼ @PutMapping VS @PostMapping

멱등성 : 동일한 요청을 한 번 보내는 것과 여러 번 보내는 것이 같은 효과를 지닌다.

@PutMapping

1. 주로 수정할 때 사용한다.
2. 여러 번 호출할 경우에 클라이언트가 받는 응답은 동일하다(멱등성을 가진다.)

@PostMapping

1. 기타 작업을 할 때 사용한다.


▼ @RequestBody & @ResponseBody

스프링 프레임워크에서 API 통신(비동기 통신)을 구현하기 위해

@RequestBody 어노테이션과 @ResponseBody 어노테이션을 사용한다.


@RequestBody(요청)

JSON 기반의 HTTP Body를 자바 객체로 변환.

 클라이언트 → 서버

@ResponseBody(응답)

자바 객체를 JSON 기반의 HTTP Body로 변환.

 서버 → 클라이언트

▼ etc.

화면전환없이 submit() 실행.

```

        <p>
        <form:form name="addForm" method="put" target="cart">
        <a href="javascript:addToCart('/cart/add/${car.cid}')" class="btn btn-primary">제품주문 &raquo;</a>
        <a href="c:url value='/cart' />" class="btn btn-warning">장바구니 &raquo;</a>
        <a href="c:url value='/cars' />" class="btn btn-success">제품목록 &raquo;</a>
        </form:form>

        </div>
    </div>

</body>

<iframe name="cart" style="display: none;"></iframe>

<script type="text/javascript">

//구매하려는 제품을 장바구니로 보낼 때 JS를 이용하여 화면전환 없이 submit()을 실행한다.
function addToCart(action) {
    document.addForm.action = action;
    document.addForm.submit();
    alert("제품이 장바구니에 추가되었습니다.");
}

```

CartItem.java(오타수정)

```

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

```

```

    public void setQuantity(int quantity) {
        this.quantity = quantity;
        this.updateTotalPrice();
    }

```

Spring Web Flow

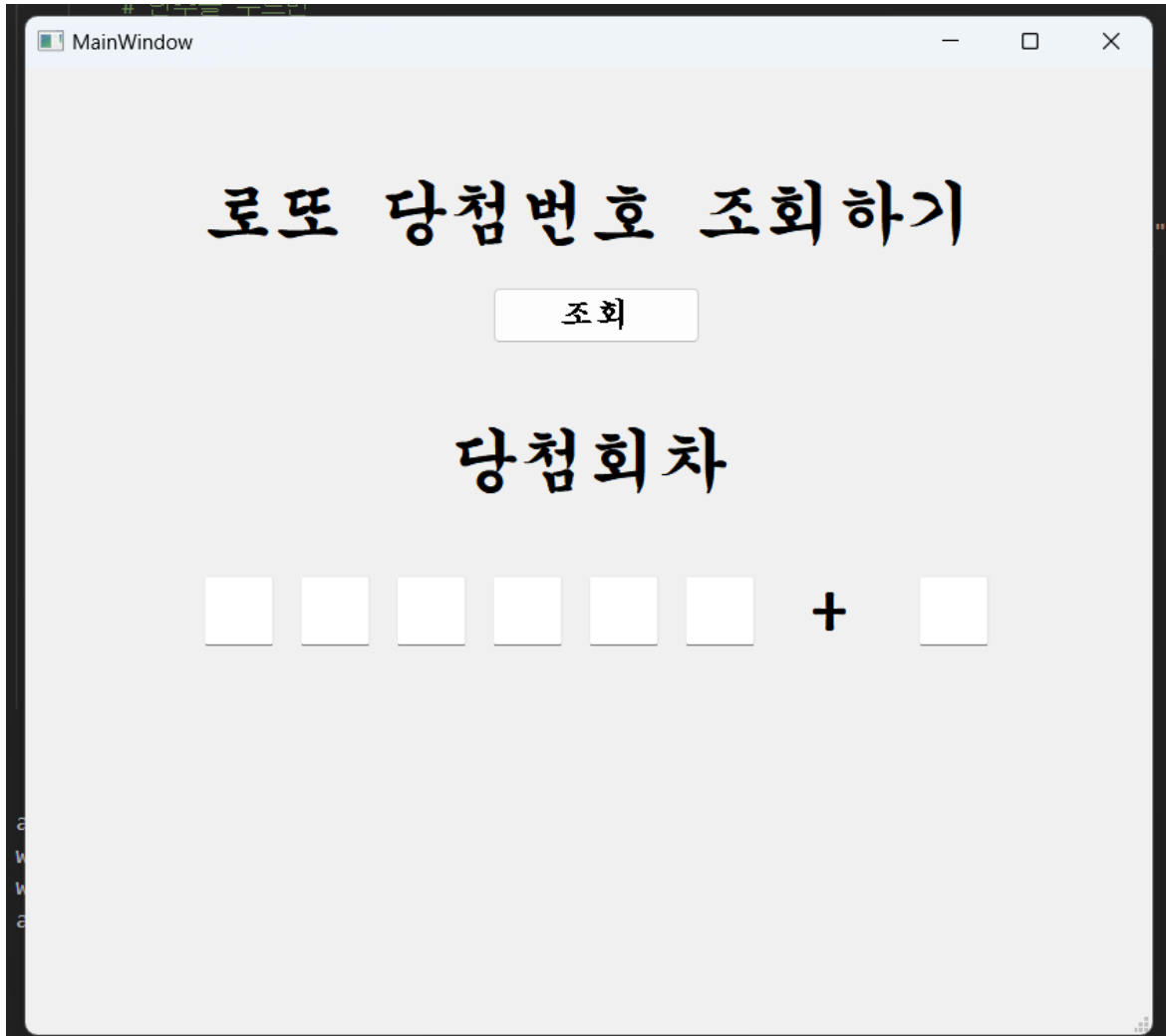
웹 페이지 구성이 복잡한 사이트를 개발하는 경우에
페이지들의 흐름을 추적하고 관리할수 있는 기능.

주문 처리 흐름 : 주문신청 → 주소입력 → 결제 → 주문완료

- 웹 페이지 흐름을 깔끔하게 한눈에 파악할 수 있다.
- Spring 외의 다른 프레임워크와도 연동해서 사용할 수 있다.
즉, Spring에 포함된 것은 아니다.
- 특정 컨트롤러를 사용하지 않고 적절한 흐름을 관리 할 수 있다.
- 사이트가 복잡할 때 흐름을 관리하여 결과적으로 사용하기 쉽게 만들어 준다.

▼ Python

로또 번호 조회 시스템



```
# 로또 번호 조회 시스템

import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic
from PyQt5.QtCore import *
import time
import requests
from bs4 import BeautifulSoup as bs # 데이터 분석을 용이하게 정제
from selenium import webdriver      # 웹 브라우저 컨트롤(크롬)

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

class MyWindow(QMainWindow, from_class):

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

        # 단추를 누르면
        self.pushButton.clicked.connect(self.PrintFunction)

    def PrintFunction(self) :
```

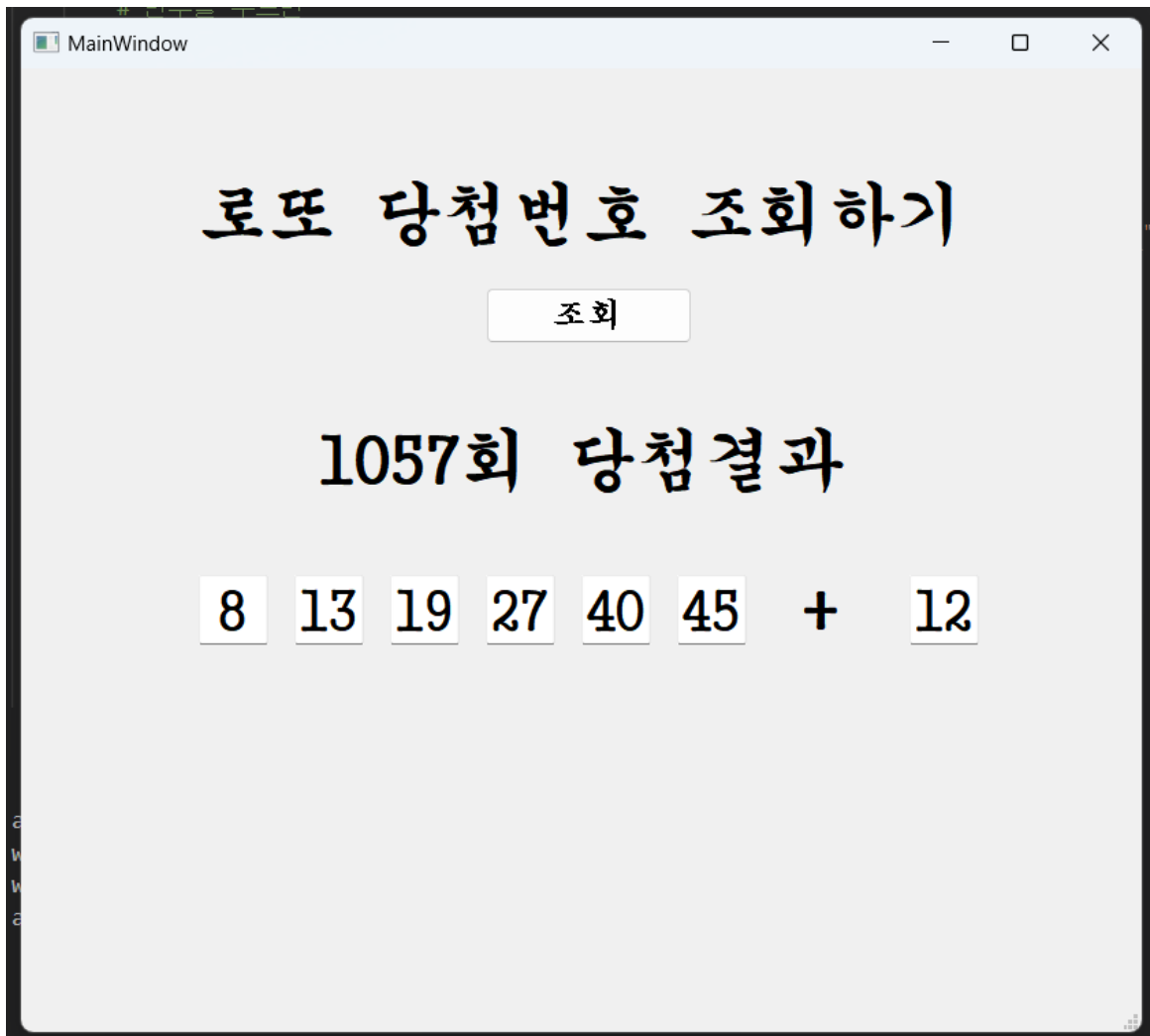
```

driver = webdriver.Chrome('chromedriver.exe')
driver.get("https://dhlottery.co.kr/gameResult.do?method=byWin&wiselog=C_A_1_1")
txt = driver.page_source
html = bs(txt) # 벅스 문법과 약간 다름

time.sleep(3)
lotto = html.select('div.win_result > h4')[0].text
lottery = html.select('span.ball_645')
self.title.setText(lotto)
self.lineEdit.setText(lottery[0].text)
self.lineEdit_2.setText(lottery[1].text)
self.lineEdit_3.setText(lottery[2].text)
self.lineEdit_4.setText(lottery[3].text)
self.lineEdit_5.setText(lottery[4].text)
self.lineEdit_6.setText(lottery[5].text)
self.lineEdit_7.setText(lottery[6].text)

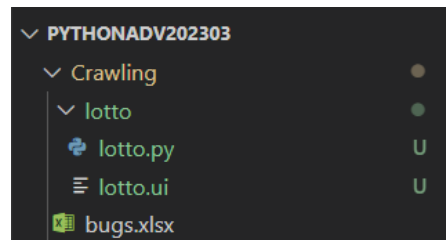
app = QApplication(sys.argv)
window = MyWindow()
window.show()
app.exec_()

```

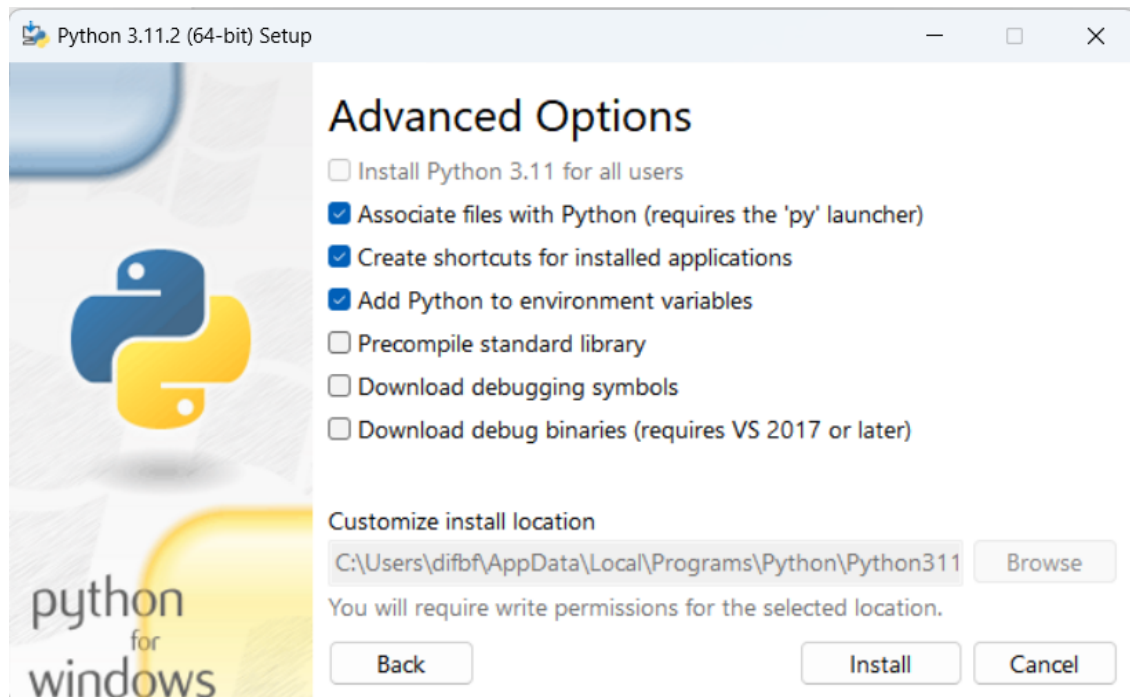


윈도우 앱 만들기

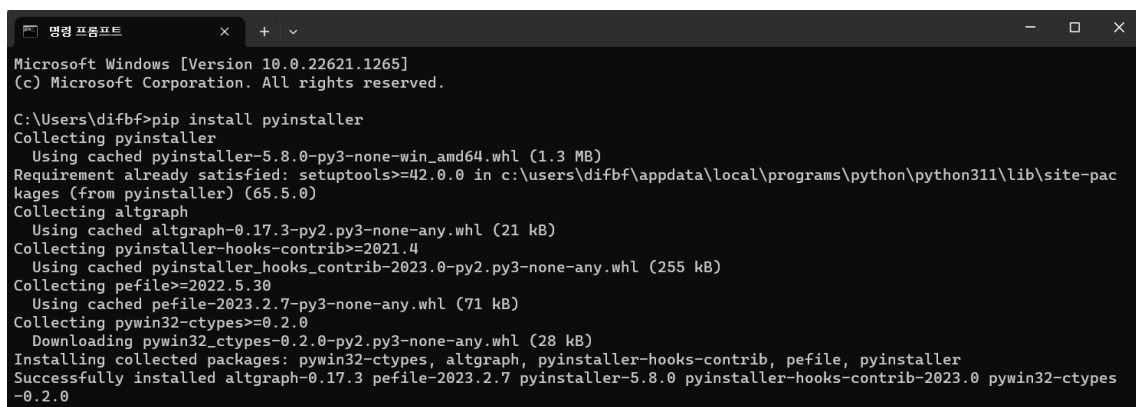
1. 앱으로 만들 폴더 만들기.



2. Python 설치.



3. Pyinstaller 설치.



4. 해당폴더로 이동.

```
C:\develop\pythonadv202303\Crawling\lotto3>cd C:\develop\pythonadv202303\Crawling\lotto3
C:\develop\pythonadv202303\Crawling\lotto3>
```









5. 윈도우 앱 만들기.

```
C:\develop\pythonadv202303\Crawling\lotto3>pyinstaller lotto.py
```

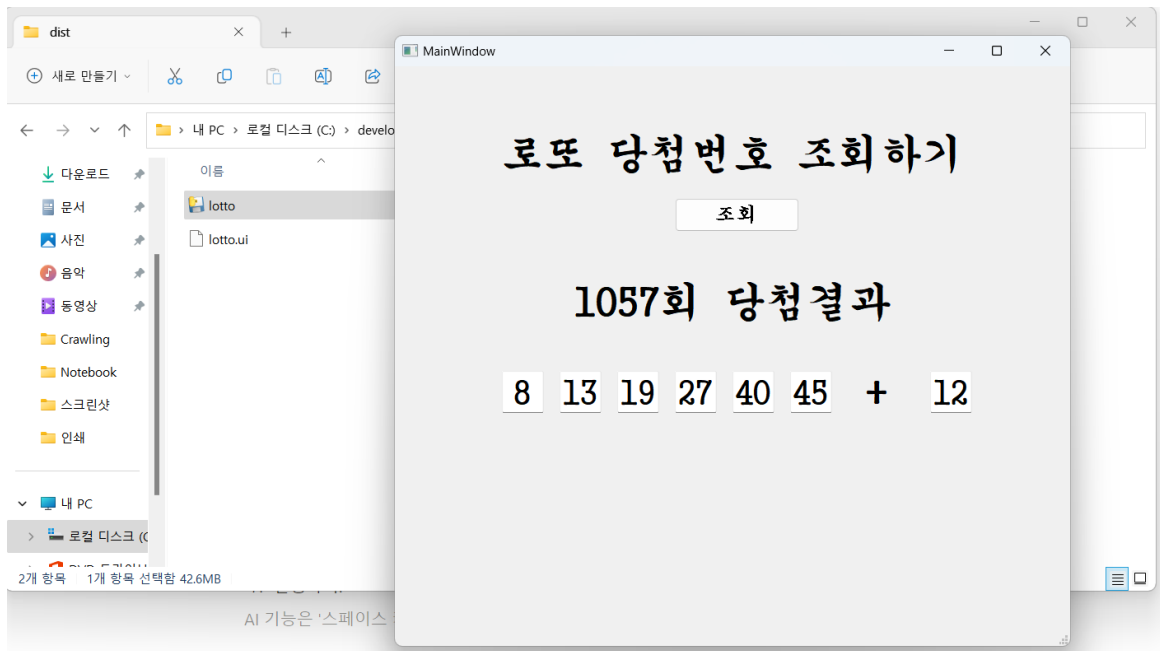
실행시 창 하나 띄우기 & 폴더안에 한개의 실행파일로 만들기.

```
C:\develop\pythonadv202303\Crawling\lotto3>pyinstaller lotto.py -w -F
```

6. 윈도우 앱 폴더안에 ui파일 통합.

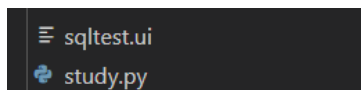
<div> <div></div> <div> <div></div> <div>내 PC</div> </div> <div> <div></div> <div>로컬 디스크 (C:)</div> </div> <div> <div></div> <div>develop</div> </div> <div> <div></div> <div>pythonadv202303</div> </div> <div> <div></div> <div>Crawling</div> </div> <div> <div></div> <div>lotto2</div> </div> <div> <div></div> <div>dist</div> </div> <div> <div></div> <div>lotto</div> </div> <div> <div></div> <div></div> </div> <div> <div></div> <div></div> </div> </div>				
이름	수정된 날짜	유형	크기	
 api-ms-win-crt-stdio-l1-1-0.dll	2023-03-08 오후 3:21	응용 프로그램 확장	18KB	
 api-ms-win-crt-string-l1-1-0.dll	2023-03-08 오후 3:21	응용 프로그램 확장	18KB	
 api-ms-win-crt-time-l1-1-0.dll	2023-03-08 오후 3:21	응용 프로그램 확장	14KB	
 api-ms-win-crt-utility-l1-1-0.dll	2023-03-08 오후 3:21	응용 프로그램 확장	12KB	
 base_library	2023-03-08 오후 4:42	압축(ZIP) 폴더	1,710KB	
 libcrypto-1_1.dll	2023-03-08 오후 3:21	응용 프로그램 확장	3,361KB	
 libssl-1_1.dll	2023-03-08 오후 3:21	응용 프로그램 확장	687KB	
 lotto	2023-03-08 오후 4:42	응용 프로그램	2,813KB	
 lotto.ui	2023-03-08 오후 2:32	UI 파일	7KB	
 pyexpat	2023-03-08 오후 4:37	Python Extension ...	195KB	
 python3.dll	2023-03-08 오후 4:37	응용 프로그램 확장	66KB	

7. 실행.



윈도우 앱 만들기(실습)

폴더 만들기.



study.py

```
import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic
import requests # 파이썬으로 웹페이지 연결
from bs4 import BeautifulSoup as bs # 분석을 용이하게 정제
import time
import datetime
import pymysql

form_class = uic.loadUiType("sqltest.ui")[0]

class MyWindow(QMainWindow, form_class):
    def __init__(self):
        super().__init__()
        self.setupUi(self)

# 단추 인식 시키기
self.pushButton.clicked.connect(self.btn_clicked)

def btn_clicked(self):
    conn = pymysql.connect(host='creatordodo.shop', user='difbfl4751', password='비밀번호', db='difbfl4751', charset='utf8')
    cur = conn.cursor()
    cur.execute("SELECT * FROM study")
    result = cur.fetchall()

    for i, data in enumerate(result):
        self.tableWidget.setItem(i, 0, QTableWidgetItem(str(data[1], 'utf-8')))
        self.tableWidget.setItem(i, 1, QTableWidgetItem(str(data[2], 'utf-8')))
        self.tableWidget.setItem(i, 2, QTableWidgetItem(str(data[5], 'utf-8')))
        self.tableWidget.setItem(i, 3, QTableWidgetItem(datetime.datetime.strptime(data[9], '%Y-%m-%d-%H-%M-%S')))
```

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

윈도우 앱 만들기.

```
C:\develop\pythonadv202303\Crawling\study>pyinstaller study.py -w -F
```

실행.

