Spring Day 10

▼ Spring

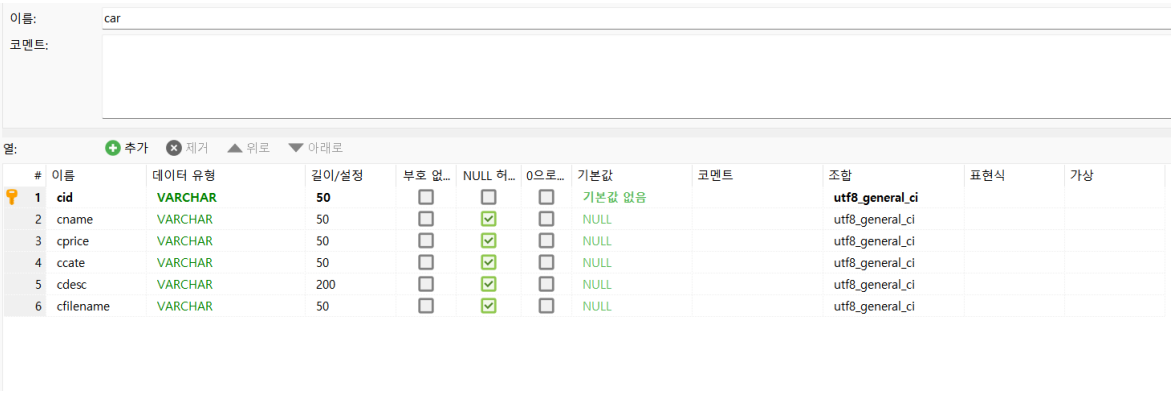
DB 연동하기

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/aa661e8d-e719-4d62-a76c-fc1e02a267ca/CarShop_JDBC.zip

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/e7c124f5-de02-4dd5-a743-c8d4d6e6f700/CarShop_MyBatis.zip

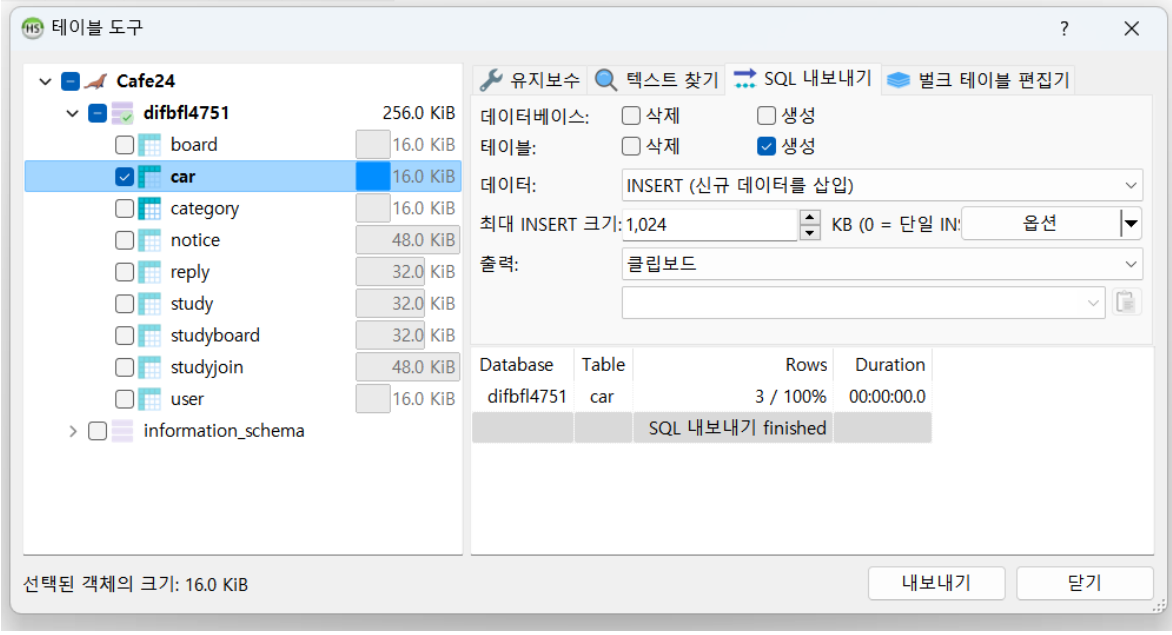
JDBC

실습용 DB 데이터 입력



#	이름	데이터 유형	길이/설정	부호 없...	NULL 허...	0으로...	기본값	코멘트	조합	표현식	가상
1	cid	VARCHAR	50	☐	☐	☐	기본값 없음		utf8_general_ci		
2	cname	VARCHAR	50	☐	☑	☐	NULL		utf8_general_ci		
3	cprice	VARCHAR	50	☐	☑	☐	NULL		utf8_general_ci		
4	ccate	VARCHAR	50	☐	☑	☐	NULL		utf8_general_ci		
5	cdesc	VARCHAR	200	☐	☑	☐	NULL		utf8_general_ci		
6	cfilename	VARCHAR	50	☐	☑	☐	NULL		utf8_general_ci		

cid	cname	cprice	ccate	cdesc	cfilename
c0001	아반테	3000	준형	신규	c0001.jpg
c0002	그랜저	4000	준대형	중고	c0002.jpg
c0003	페라리	8000	준중형	신규	c0003.jpg




```
CREATE TABLE `car` (
  `cid` VARCHAR(50) NOT NULL COLLATE,
  `cname` VARCHAR(50) NULL DEFAULT NULL,
  `cprice` VARCHAR(50) NULL DEFAULT NULL,
  `ccate` VARCHAR(50) NULL DEFAULT NULL,
  `cdesc` VARCHAR(200) NULL DEFAULT NULL,
  `cfilename` VARCHAR(50) NULL DEFAULT NULL,
  PRIMARY KEY (`cid`) USING BTREE
)
ENGINE=InnoDB
;

-- 테이블 데이터 difbfl4751.car:~3 rows (대략적) 내보내기
INSERT INTO `car` (`cid`, `cname`, `cprice`, `ccate`, `cdesc`, `cfilename`) VALUES
  ('c0001', '아반테', '3000', '준형', '신규', 'c0001.jpg'),
  ('c0002', '그랜저', '4000', '준대형', '중고', 'c0002.jpg'),
  ('c0003', '페라리', '8000', '준중형', '신규', 'c0003.jpg');
```

JDBC 연결 설정

1. **pom.xml 설정(의존성 주입)**

Home » org.springframework » spring-jdbc » 5.3.19

Spring JDBC » 5.3.19

Spring JDBC provides an abstraction layer that simplifies code to use JDBC and the parsing of database-vendor specific error codes.

License	Apache 2.0
Categories	JDBC Extensions
Tags	sql jdbc spring
Organization	Spring IO
HomePage	https://github.com/spring-projects/spring-framework
Date	Apr 13, 2022
Files	pom (2 KB) jar (418 KB) View All
Repositories	Central Spring Releases USIT
Ranking	**#111 in MvnRepository (See Top Artifacts)** **#1 in JDBC Extensions**
Used By	**4,014 artifacts**
Vulnerabilities	**Vulnerabilities from dependencies:** CVE-2022-22971 CVE-2022-22970

Note: There is a new version for this artifact

New Version	6.0.6

Home » org.apache.commons » commons-dbcp2 » 2.5.0

Apache Commons DBCP » 2.5.0

Apache Commons DBCP software implements Database Connection Pooling

License	Apache 2.0
Categories	JDBC Pools
Tags	sql jdbc apache pool commons
HomePage	http://commons.apache.org/dbcp/
Date	Jul 13, 2018
Files	jar (192 KB) View All
Repositories	Central
Ranking	**#483 in MvnRepository (See Top Artifacts)** **#3 in JDBC Pools**
Used By	**897 artifacts**
Vulnerabilities	**Vulnerabilities from dependencies:** CVE-2022-45868 CVE-2022-23221 CVE-2021-42392 View 2 more ...

Home » mysql » mysql-connector-java » 8.0.28

MySQL Connector Java » 8.0.28

MySQL Connector/J is a JDBC Type 4 driver, which means that it is pure Java implementation of the MySQL protocol and does not rely on the MySQL client libraries. This driver supports auto-registration with the Driver Manager, standardized validity checks, categorized SQLExceptions, support for large update counts, support for local and offset date-time variants from the java.time package, support for JDBC-4.x XML processing, support for per connection client information and support for the NCHAR, NVARCHAR ...

License	GPL 2.0
Categories	JDBC Drivers
Tags	database sql jdbc driver connector mysql
Organization	Oracle Corporation
HomePage	http://dev.mysql.com/doc/connector-j/en/
Date	Jan 17, 2022
Files	pom (2 KB) jar (2.4 MB) View All
Repositories	Central
Ranking	**#68 in MvnRepository** (See Top Artifacts) **#1 in** JDBC Drivers
Used By	6,902 artifacts
Vulnerabilities	Vulnerabilities from dependencies: CVE-2022-3510 CVE-2022-3509 CVE-2022-3171 View 1 more ...

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
  <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-jdbc</artifactId>
      <version>5.3.19</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.apache.commons/commons-dbcp2 -->
  <dependency>
      <groupId>org.apache.commons</groupId>
      <artifactId>commons-dbcp2</artifactId>
      <version>2.5.0</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
  <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.28</version>
  </dependency>
```

2. **servlet-context.xml 설정**

```
  <beans:bean id="dataSource"
              class="org.springframework.jdbc.datasource.DriverManagerDataSource">
      <beans:property name="driverClassName" value="com.mysql.cj.jdbc.Driver"/>
      <beans:property name="url" value="jdbc:mysql://localhost:3306/difbfl4751?serverTimezone=UTC"/>
      <beans:property name="username" value="difbfl4751"/>
      <beans:property name="password" value="비밀번호"/>
  </beans:bean>
```

```
<beans:bean id="dataSource"
              class="org.springframework.jdbc.datasource.DriverManagerDataSource">
       <beans:property name="driverClassName" value="com.mysql.cj.jdbc.Driver"/>
       <beans:property name="url" value="jdbc:mysql://localhost:3306/difbfl4751?serverTimezone=UTC"/>
       <beans:property name="username" value="difbfl4751"/>
       <beans:property name="password" value="비밀번호"/>
   </beans:bean>
```

3. **DB 연결하기**


```
private JdbcTemplate template;

  @Autowired
  public void setJDBCTemplate(DataSource dataSource) {
    this.template = new JdbcTemplate(dataSource);
  }
```

JDBC 사용하기

CarDTO.java

```
package com.carshop.controller;

import java.io.Serializable;

import org.springframework.web.multipart.MultipartFile;

@SuppressWarnings("serial")
public class CarDTO implements Serializable {
  //private static final long serialVersionUID = 3414129893883786050L;

  private String cid, cname, cprice, ccate, cdesc;
  private MultipartFile carimage;
  private String cfilename;

  public String getCfilename() {
    return cfilename;
  }

  public void setCfilename(String cfilename) {
    this.cfilename = cfilename;
  }

  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, String cfilename) {
    this.cid = cid;
    this.cname = cname;
    this.cprice = cprice;
    this.ccate = ccate;
    this.cdesc = cdesc;
    this.cfilename = cfilename;
  }

  public CarDTO() {
    super();
  }




}
```

CarRowMapper.java

```
package com.carshop.controller;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class CarRowMapper implements RowMapper<CarDTO> {

  public CarDTO mapRow(ResultSet rs, int rowNum) throws SQLException {
    CarDTO car = new CarDTO();

    car.setCid(rs.getString(1));
    car.setCname(rs.getString(2));
    car.setCprice(rs.getString(3));
    car.setCcate(rs.getString(4));
    car.setCdesc(rs.getString(5));
    car.setCfilename(rs.getString(6));


    return car;

  }


}
```

CarRepositoryImpl.java

```
package com.carshop.controller;

import java.util.*;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
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", null);
//    CarDTO car2 = new CarDTO("c0002", "그랜저", "3500", "중형", "신규", null);
//    CarDTO car3 = new CarDTO("c0003", "아반테", "2000", "대형", "신규", null);
//    CarDTO car4 = new CarDTO("c0004", "K5", "3000", "중형", "신규", null);
//
//    listOfCars.add(car1);
//    listOfCars.add(car2);
//    listOfCars.add(car3);
//    listOfCars.add(car4);
//
//  }

//  @Override
//  public List<CarDTO> getAllCarList() {
//
//    return listOfCars;
//  }

  public List<CarDTO> getCarListByCategory(String category) {
    List<CarDTO> carsByCategory = new ArrayList<CarDTO>();
    for (int i = 0; i < listOfCars.size(); i++) {
      CarDTO carDTO = listOfCars.get(i);
      if (category.equalsIgnoreCase(carDTO.getCcate())) {
        carsByCategory.add(carDTO);
      }
    }
    return carsByCategory;
  }

//  @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;
//  }

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

  private JdbcTemplate template;

  @Autowired
  public void setJDBCTemplate(DataSource dataSource) {
    this.template = new JdbcTemplate(dataSource);
  }


//  기존 방식(DB 연결없이 임시로 변수를 만들어서 사용한 매서드)
//  @Override
//  public List<CarDTO> getAllCarList() {
//
//    return listOfCars;
//  }
```

```
  @Override
  public List<CarDTO> getAllCarList() {
    String sql = "SELECT * FROM car";
    List<CarDTO> listOfCars = template.query(sql, new CarRowMapper());
    return listOfCars;
  }

  @Override
  public CarDTO getCarById(String carId) {

    CarDTO carInfo = null;

    String sql = "SELECT count(*) FROM car where cid=?";
    int rowCount = template.queryForObject(sql, Integer.class, carId);

    if(rowCount != 0) {
      sql = "SELECT * FROM car where cid=?";
      carInfo = template.queryForObject(sql, new Object[] {carId}, new CarRowMapper());
    }


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


    return carInfo;
  }

  @Override
  public void setNewCar(CarDTO car) {
    String sql = "INSERT INTO car (cid, cname, cprice, ccate, cdesc, cfilename) "
        + "VALUE (?,?,?,?,?,?)";

    template.update(sql, car.getCid(),
              car.getCname(),
              car.getCprice(),
              car.getCcate(),
              car.getCdesc(),
              car.getCfilename());
  }


}
```

CarController.java

```
//  @PostMapping("/add")
//  public String submitAddNewCar(@ModelAttribute("NewCar") CarDTO car, HttpServletRequest request) {
//
//    MultipartFile carimage = car.getCarimage();
//    String saveName = carimage.getOriginalFilename();
//
////    String uploadpath = request.getRealPath("/resources/images");
//
////    File saveFile = new File("C:\\upload", saveName);
//    File saveFile = new File(uploadPath + "\\images", saveName);
////    System.out.println(saveFile.getPath());
//
//
//    if (carimage != null && !carimage.isEmpty()) {
//      try {
//        carimage.transferTo(saveFile);
//      } catch (Exception e) {
////        throw new RuntimeException("차량 이미지 업로드가 실패했습니다.");
//        e.printStackTrace();
//      }
//    }
//
//
//    carService.setNewCar(car);
//    return "redirect:/cars";
//  }

  @PostMapping("/add")
  public String submitAddNewCar(@ModelAttribute("NewCar") CarDTO car, HttpServletRequest request) {


    MultipartFile carimage = car.getCarimage();
    String saveName = carimage.getOriginalFilename();

    File saveFile = new File(uploadPath + "\\images", saveName);
```

```
    if (carimage != null && !carimage.isEmpty()) {
      try {
        carimage.transferTo(saveFile);
        car.setCfilename(saveName);
      } catch (Exception e) {
        e.printStackTrace();
      }
    }


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

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">
<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="/resources/images/${car.getCarimage().getOriginalFilename()}"/>'/> --%>
<%--        <img src='<c:url value="/images/${car.getCarimage().getOriginalFilename()}"/>'/> --%>



              <img src="<c:url value="/resources/images/${car.cfilename}"/>"
                style="width: 60%" />

          <h3>${car.cid}</h3>
          <p>${car.cname}
        <p>${car.cprice}만원
        <p><a href="/cars/car?id=${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>
```

car.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<script src="https://code.jquery.com/jquery-3.6.3.min.js" integrity="sha256-pvPw+upLPUjgMXY0G+8O0xUf+/Im1MZjXxxgOcBQBXU=" crossorig

<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">
<div class="alert alert-dark" role="alert">
<div class="container"><h1>차량 상세보기</h1></div>
</div>


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

          <img src="<c:url value="/resources/images/${car.cfilename}"/>"
            style="width: 60%" />


      <h3>${car.cid}</h3>
      <p>${car.cname}
        <p>${car.cprice}만원
        <p>${car.ccate}
        <p>${car.cdesc}


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

<!--        기존 방식을 AJAX로 적용 -->
        <a href="javascript:addCartFunction()" 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.");
// }

function addCartFunction() {
  $.ajax({
    type:"POST",
    url:"/cart/ajaxAdd",
    data:{cid:"${car.cid}"},
    beforeSend : function(xhr)
        {   /*데이터를 전송하기 전에 헤더에 csrf값을 설정한다*/
            xhr.setRequestHeader("${_csrf.headerName}", "${_csrf.token}");
        },
    success:function(result) {
      alert("제품이 장바구니에 추가되었습니다.")
    },
    error:function(request, status, error) {
      alert(request.status + " " + request.responseText);
    }

  })
}

</script>




</html>
```

▼ Python

Pandas

- 파이썬에서 가장 많이 사용되는 라이브러리이다.
- **데이터분석**의 전분야와 **인공지능**에서도 넘파이와 더불어 아주 많이 사용된다.

▼ 데이터분석 훈련 웹사이트

전세계

Kaggle: Your Machine Learning and Data Science Community

Kaggle is the world's largest data science community with powerful tools and resources to help you achieve your data science goals.

https://www.kaggle.com/

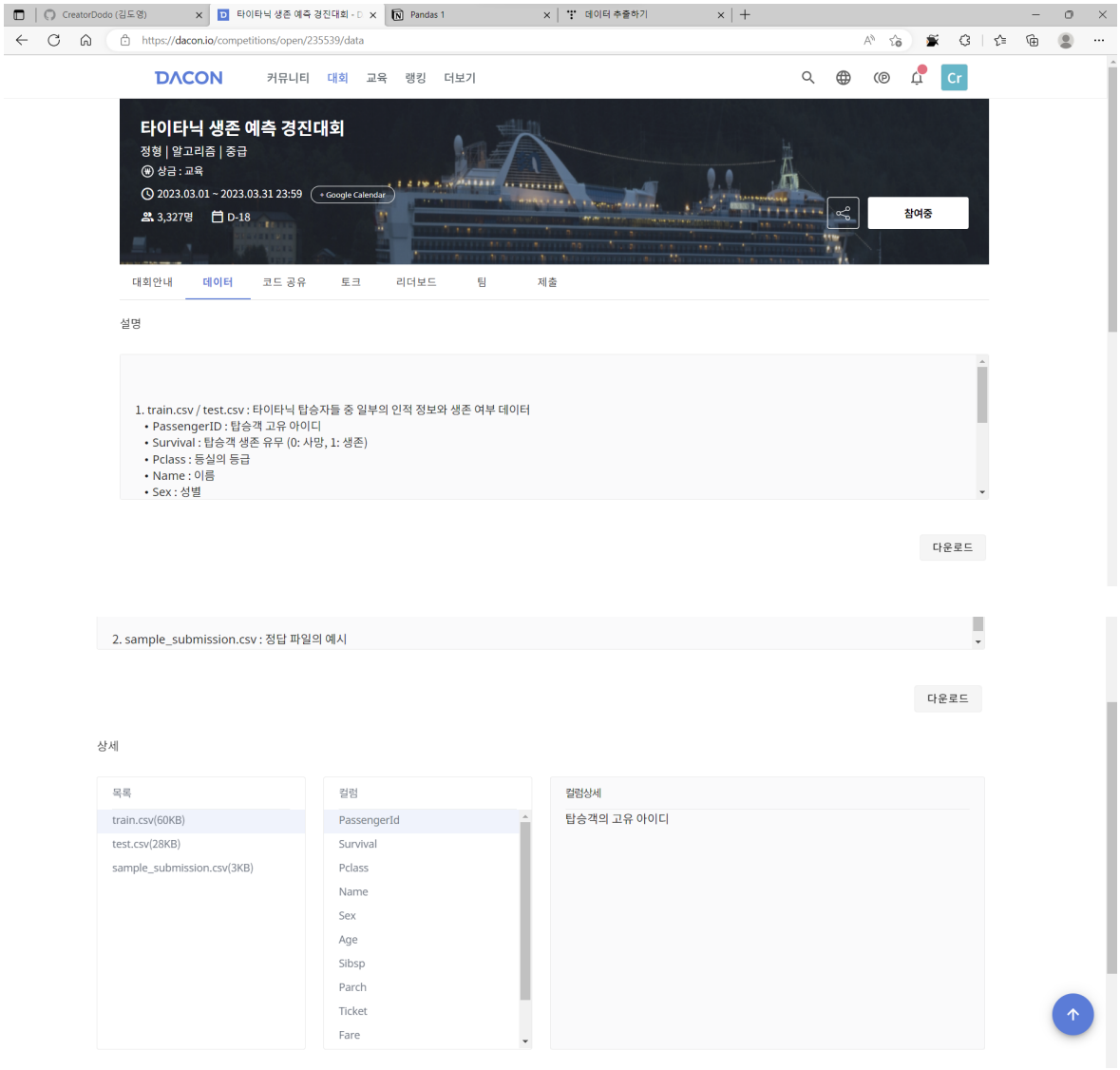
국내

데이터사이언티스트 AI 컴피티션

7만 AI 팀이 협업하는 데이터 사이언스 플랫폼. AI 경진대회와 대상 맞춤 온/오프라인 교육, 문제 기반 학습 서비스를 제공합니다.

https://dacon.io/

데이터 가져오기



https://s3-us-west-2.amazonaws.com/secure.notion-static.com/18ca49bf-db3d-4db2-8a4f-f69fd0057d9f/data.zip

Code

Pandas

```
# !pip install pandas # 판다스 설치

import pandas as pd

# 판다스로 분석할 데이터 가져오기

df = pd.read_csv('train.csv')   # 타이타닉 데이터 읽어오기
df

df = pd.read_csv('gapminder.tsv', sep='\t')
df
```

```
df.head() # 위쪽에서 5개의 데이터만 출력

df.tail() # 아래쪽에서 5개의 데이터만 출력

df.shape    # shape는 괄호가 없다. 레코드(행)와 피처값(열)의 개수 출력.

type(df) # 자료형 확인

df.info()

# object      -> String
# int64       -> int
# float64     -> float
# datetime64  -> datetime

# 원하는 데이터를 추출하기

# 열단위 추출과 행단위 추출

country_df = df['country']
print(country_df)

type(country_df)

country_df.head()

country_df.tail()

subset = df[['country', 'continent', 'year']]
subset

type(subset)

# 필요로 하는 데이터만 추출하기 - 행단위 추출
# loc(인덱스로 추출) / iloc (행번호)
df.head()

# loc(인덱스로 추출)

df.loc[100]

# df.loc[-1] # loc는 인덱스 값을 찾기 때문에 오류가 발생한다.

number_of_rows = df.shape[0]
last_row_index = number_of_rows - 1
df.loc[last_row_index]

# iloc

# loc는 인덱스를 사용하여 데이터를 추출, iloc는 행번호를 사용하여 데이터를 추출한다.

print(df.iloc[1])
print(df.loc[1])

# iloc는 음수로 추출 가능하다.
print(df.iloc[-1])

# iloc는 여러 데이터를 한번에 추출할 수 있다.

df.iloc[[0,10,100]]

# 슬라이싱 기법으로 데이터 추출

subset = df.loc[:,['year', 'pop']]
subset

# 슬라이싱 기법으로 데이터 추출

print(df.loc[:,['year', 'pop']])
print(df.iloc[:,[2, 4]])
# print(df.loc[:,[2, 4]]) # loc는 숫자로 추출 불가능

list(range(5))

df.iloc[:,range(5)]

# 실무에서는 range보다는 슬라이싱 구문을 더 선호한다.

df.iloc[:, :3]

df.iloc[:, 0:6:2]

# 0, 99, 999 행의 0,3,5 열의 데이터만 추출하세요.

print(df.iloc[[0, 99, 999], [0, 3, 5]])
```

```
# 실무적으로 열이름 등을 번호로 사용하는 빈도가 더 많지만
# 나중에 데이터 확인이 어려울 수 있다.


# 그렇기 때문에 대규모 분석에 있어서는 loc속성이 더 유용하다고 볼 수 있다.
print(df.loc[[0, 99, 999], ['country', 'lifeExp', 'gdpPercap']])

# 기초 통계 산출
df.head()   # 괄호 안을 생략하면 앞에서 5개

df.head(7)  # 인덱스(0 ~ 6) 7개 추출

df.head(n=7)  # 인덱스(0 ~ 6) 7개 추출

# 연도별로 묶어서 기대 수명의 평균을 계산

df.groupby('year')['lifeExp'].mean()

type(df.groupby('year'))

df.groupby('year')

df.groupby(['year', 'continent'])['lifeExp'].mean()

df.groupby(['year', 'continent'])[['lifeExp','gdpPercap']].mean()


# 그룹화한 데이터의 개수

df.groupby('continent')['country'].nunique()


# 시각화 기본

# !pip install matplotlib

import matplotlib.pyplot as plt


# 연도별로 묶어서 기대 수명의 평균을 계산
print(df.groupby('year')['lifeExp'].mean())

print(df.groupby('year')['lifeExp'].mean().plot())
```

Series

```
# Pandas의 Series와 Dataframe

# 시리즈(Series)

s = pd.Series(['apple', 32])

print(type(s))
print(s)

# 시리즈의 기본 인덱스는 0부터 증가된다.
# 시리즈의 인덱스는 원하는 값으로 설정할 수 있다.

pd.Series(['진', '제이홉'])

pd.Series(['진', '제이홉'], index=['1st','2nd'])

# Dataframe 만들기


bts = pd.DataFrame(
    data={'Name': ['진','제이홉'],
          'Age': [25, 27],
          'Etc': ['BTS','BTS']},
    index=['1st','2nd'],
    columns=['Name', 'Age', 'Etc'])

print(bts)
print(type(bts))



# bts = pd.DataFrame({
#     '이름':['진', '제이홉'],
#     '나이':[25, 27],
#     '소속':['BTS', 'BTS']
# })
```

```
# print(bts)
# print(type(bts))

# 시리즈처럼 데이터프레임도 인덱스를 생략하면 0부터 자동부여된다.
# 또한 시리즈처럼 인덱스를 직접 지정할 수 있다.

bts = pd.DataFrame([['진', 25, 'BTS'],
        ['제이홉', 27, 'BTS']],
index=['1st', '2nd'],
columns=['name', 'age', 'etc']
)
bts
# print(bts)
# print(type(bts))


# 시리즈(Series) 기초

# 데이터프레임(Dataframe)은 시리즈의 모음이다.

bts = pd.DataFrame([['진', 25, 'BTS'],
        ['제이홉', 27, 'BTS']],
index=['1st', '2nd'],
columns=['name', 'age', 'etc']
)

bts.loc['2nd']

type(bts.loc['2nd'])

bts2 = bts.loc['2nd']
type(bts2)

print(bts2)

# 출력시 자료형을 object으로 인식한다.

# index 속성

bts2.index

# values 속성

bts2.values

# keys 매서드

bts2.keys()

# keys()와 index는 같은 결과를 추출한다. 하지만 keys()는 매서드로 동작한다.

# 활용방법

bts2.index[2]

bts2.keys()[2]

# 시리즈에서 사용되는 기초 통계 매서드들

bts['age']

bts['age'].mean()   # 평균

bts['age'].min()    # 최소값

bts['age'].max()    # 최대값

bts['age'].std()    # 표준편차

# 시리즈 응용편

sci = pd.read_csv('scientists.csv')
sci

ages = sci['Age']

print(ages.mean())

# 나이가 평균 나이보다 많은 사람만 추출

ages[ages > ages.mean()]

ages > ages.mean()

val = [False, True, True, True, False, False, False, True]
print(ages[val])
```

```
# 시리즈나 데이터프레임의 모든 데이터에 대하여 한번에 연산하는 것을 "브로드캐스팅"이라고 한다.

# 브로드캐스팅1

print(ages)
print(ages + ages)

# 브로드캐스팅2

print(ages)
print(ages * ages)

# 브로드캐스팅3
print(ages)
print(ages + 100)
```

Dataframe

```
# 데이터프레임(Dataframe)

sci[sci['Age'] > sci['Age'].mean()]

sci['Age'] > sci['Age'].mean()

sci[[False, True, True, True, False, False, False, True]]

# 숫자의 경우는 2배가 되고 문자의 경우는 같은 값이 두번 출력된다.

sci * 2

# 자료형 바꾸기

sci['Born']

sci['Died']

# 날짜처럼 보이지만 글자인 데이터를 진짜 날짜로 형변환하기

born_temp = pd.to_datetime(sci['Born'], format='%Y-%m-%d')
died_temp = pd.to_datetime(sci['Died'], format='%Y-%m-%d')
print(born_temp)
print(type(born_temp))
print(died_temp)

sci['born_dt'] = born_temp

sci

sci['died_dt'] = died_temp
sci

sci['days_dt'] = sci['died_dt'] - sci['born_dt']
sci

sci.drop(['died_bt'], axis=1)
```

▼ etc.

한 권으로 끝내는 <판다스 노트>

판다스는 파이썬에서 가장 널리 쓰이는 라이브러리 가운데 하나입니다. 데이터 분석 전문가가 파이썬으로 데이터 분석을 한다면, 아마 대부분은 가장 먼저 판다스 라이브러리를 임포트할 …

https://wikidocs.net/book/4639