#JSP MVC 모델2 패턴을 이용한 서블릿 클래스 구현

**public** **class** ControllerSelvlet **extends** HttpServlet {

**private** Map commandHandlerMap = **new** java.util.HashMap();//<명령어,핸들러 클래스 의 매핑정보를 설정파일에저장

**public** **void** init(ServletConfig config) **throws** ServletException{//컨트롤러 서블릿에서 설정파일을 읽어오기 가장 좋은 위치는 init()메서드 이메서드는 서블릿을 생성하고 초기화 할때 호출되는 메서드

String configFile = config.getInitParameter("configFile");//web.xml에 설정한 configfile로설정된 /WEB-INF/commandHandler.properties 매핑정보를 불러온다

Properties prop = **new** Properties();//불러온 매핑정보를 저장할 properties객체를 생성한다

FileInputStream fi = **null**;

**try** {

String configFilePath = config.getServletContext().getRealPath(configFile);//fileinputstream 객체를 생설할때 절대경로를 집어넣어야해서 getrealpath()실행해주었고 getServletContext()는 페이지에대한 서블릿 실행 환경 정보를 담고있는 application 내장객체를 리턴

//fileim

//절대경로를 구할수있는 페이지에대한 서블릿 실행환경정보를 담고있는 application내장객체리턴하고

//리턴한 객체는 절대경로를 구할수있는 getRealPath()를 사용할수있어

//web.xml에 servlet에 설정해준 configfile의 절대경로를 getRealPath()를 사용하여 구한다

//ServletContext는 웹어플리케이션 전역에서 사용할수있는 값으로 하나의서블릿이 서블릿 컨테이너와 통신하기 위해서 사용되어지는 메서드들을 가지고있음

//ServletConfig는 특정서블릿에 등록하여 사용하는값

fi= **new** FileInputStream(configFilePath);//구해온 절대경로를 전달해줄 FileInputStream객체에 저장해준다

prop.load(fi);//저장한객체 fi를이용하여 properties객체에 올려준다

}**catch**(IOException e) {

**throw** **new** ServletException(e);

}**finally** {

**if**(fi != **null**)

**try** {

fi.close();

}**catch**(IOException ex) {

}

}

Iterator keylter = prop.keySet().iterator();//properties객체는 Map컬랙션중하나이고 키값에해당하는 값을 하나씩 가져오기위해서 keySet()메소드로 모든키의 Set<String>컬랜션을만들어주고 해당 리스트를 순회하며 값을 가져오기위해 Iterator객체를생성

/\*

\* Iterator객체는 리스트를 순회할수있게 해주는 객체로 데이터베이스 커서와 동작이 유사하다 Iterator의 장점

\* 1. 컬렉션에서 요소를 제어하는 기능

\* 2. next() 및 previous()를 써서 앞뒤로 이동하는 기능

\* 3. hasNext()를 써서 더 많은 요소가 있는지 확인하는 기능

\*/

**while**(keylter.hasNext()) {//Iterator의 기능인 hasNext()를 사용하여 리스트에 값이있는지 확인

String command = (String)keylter.next();//커서와같이 첫부분을 가리키는곳에서 데이터를 가져와 String에 저장한다

String handlerClassName = prop.getProperty(command);//해당키워드 command에 해당하는 key값을 가져와 handlerClassName에저장한다

**try** {

Class handlerClass = Class.*forName*(handlerClassName);//키값에해당한 classname을 forName()를통해 Class인객체(JVM이로딩하는)를 가져온다

Object handlerInstance = handlerClass.~~newInstance~~();//만들어진클래스 객체를 이용하여 객체를 만들어준다

commandHandlerMap.put(command, handlerInstance);//위의 HashMap에 해당하는키와 키값을 넣어준다

}**catch**(ClassNotFoundException e) { //class객체를 생성하므로발생할수있는오류

**throw** **new** ServletException(e);

}**catch**(InstantiationException e) {//객체를 생성하므로 발생할수있는 오류

**throw** **new** ServletException(e);

}**catch**(IllegalAccessException e) { //클래스를 호출하는데 접근이 허락 되지 않았다는 오류

**throw** **new** ServletException(e);

}

}

}

@Override

**protected** **void** doGet(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

// Get방식에서 호출되는 메소드, URL에 정보가 포함되어 보안에 약함. 기본 호출 메소드.

process(req,resp);

}

@Override

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

// Post방식에서 호출되는 메소드, URL에 정보가 포함되지 않아 안전하며, 헤더에 정보를 실음.

process(req,resp);

}

**private** **void** process(HttpServletRequest request, HttpServletResponse response)**throws** IOException, ServletException {

String command = request.getParameter("cmd");//url에서 cmd로부터 얻어온값을 반환

commandHandler handler = (commandHandler)commandHandlerMap.get(command);//얻어온값을통해 Map에해당하는 키값을 가져와 commandHandler형객체를 만들어준다

**if** (handler == **null**) {

handler = **new** nullHandler();

}

String viewPage = **null**;

**try** {

viewPage = handler.process(request, response);//만들어진 handler의 오버라이드된 process를 실행시켜준다

}**catch**(Throwable e) {

**throw** **new** ServletException(e);

}

RequestDispatcher dispatcher = request.getRequestDispatcher(viewPage);//forward시켜주기위해 해당하는 RequestDispatcher객체를 생성시켜줌

/\*요청을 보내는 방법으로는 RequestDispatcher.forward()와 RequestDispatcher.include() 두 가지 방법이 있다\*/

dispatcher.forward(request, response);

}

}

Throwable은 예외처리를 할 수 있는 최상위 클래스

web.xml 파일에 <servlet> 태그로 서블릿에 관한 환경설정을 하였습니다. <servlet> 태그 안에 서블릿의 이름, 초기 파라미터, 객체 생성 여부 등의 서블릿에 관한 속성들을 설정할 수 있었습니다. 만일 서블릿 속성 중에서 <init-param> 속성이 지정되었다면, web.xml에서 서블릿으로 파라미터를 전달한 것이므로 서블릿 페이지 내에서 <init-param>의 변수를 추출해서 사용해야 합니다.

이때 web.xml <servlet>태그에서 설정한 정보를 서블릿 페이지 내에서 추출할때에 ServletConfig객체에서 제공하는 메소드를 사용한다(ServletConfig객체는 서블릿이 실행될 때 자동으로 생성)

ServletConfig객체가 생성되는 시점

**Servlet 객체 생성 (1회) -> init() 호출 (1회) -> service(), doGet(), doPost() 호출 (요청시 매번) -> destory() 호출 (1회)**

