Model Model2 Pattern

1. MVC
   -http://localhost:8080/Model2Demo/servlets/MessageServlet?message=name
   -messageView.jsp 로 포워딩된다.
   -이 방법은 간단하지만, 명령어가 파라미터로 전달되게 되면 정보가 웹 브라우저를 통해 노출이 되어 악의적으로 사이트에 접근할 수 있는 빌미를 제공한다는 단점이 있다.

   1)com.exmple.controller.MessageController.java

```
public class MessageController extends HttpServlet {
   public void doGet(HttpServletRequest request,          //1단계 : 사용자의 요청을 받는 서비스 메소드
                     HttpServletResponse response) throws ServletException, IOException
                     {
      requestPro(request, response);
   }

   public void doPost(HttpServletRequest request, //1단계 : 사용자의 요청을 받는 서비스 메소드
                     HttpServletResponse response) throws ServletException, IOException
                     {
      requestPro(request, response);
   }

   private void requestPro(HttpServletRequest request, HttpServletResponse response)
      throws ServletException ,IOException{
      String message = request.getParameter("message");//2단계 : 사용자의 요청분석
      Object result = null;
      if (message == null || message.equals("base")) {//3단계 : 사용자의 요청에 따른 작업처리
         result = "하하하.";
      } else if (message.equals("name")) {
         result = "한지민 입니다.";
      } else {
         result = "타입이 맞지 않습니다.";
      }
      request.setAttribute("result", result);//4단계 : request의 속성에 처리결과 저장

      // 5단계 : RequestDispatcher를 사용하여 해당 뷰로 포워딩
      RequestDispatcher dispatcher = request.getRequestDispatcher("/messageView.jsp");
      dispatcher.forward(request, response);
   }
}
```

   2)messageView.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html lang="ko">
   <head>
      <meta charset="UTF-8">
      <title>간단한 Controller 의 사용예제</title>
   </head>
   <body>
      결과:
      <c:set var="result" value="${requestScope.result}" />
      <c:out value="${result}"/>
   </body>
</html>
```

   3)web.xml

```
<servlet>
   <servlet-name>MessageServlet</servlet-name>
   <servlet-class>com.example.controller.MessageController</servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>MessageServlet</servlet-name>
```

```
        <url-pattern>/servlets/MessageServlet</url-pattern>
    </servlet-mapping>



2. Command Pattern
  1)요청을 커맨드 메시지로 전달
  2)두가지 방법
    a. servlet 에 요청 파라미터를 덧붙여 전달 : 윗방법
      -ex)MessageController?message=name
      -간편하다. 하지만, 파라미터가 노출된다.
      -악의적 접근 빌미 제공

    b. 요청한 URL 자체를 명령어로 제출
      -http://localhost:8080/Model2Demo/test.do


3. 요청 파라미터로 명령어를 전달하는 방법
  -http://localhost:8080/Model2Demo/servlets/Controller?command=Message

  1)command.properties
    Message=com.javasoft.libs.Controller.MessageProcess


  2)com.example.controller.CommandProcess.java
    package com.example.controller;

    import javax.servlet.http.HttpServletRequest;
    import javax.servlet.http.HttpServletResponse;

    //요청 파라미터로 명령어를 전달하는 방식의 슈퍼 인터페이스
    public interface CommandProcess {
      public String requestPro(HttpServletRequest request, HttpServletResponse response) throws
      Throwable;
    }


  3)com.example.controller.MessageProcess.java
    package com.example.controller;

    import javax.servlet.http.HttpServletRequest;
    import javax.servlet.http.HttpServletResponse;

    public class MessageProcess implements CommandProcess {

      @Override
      public String requestPro(HttpServletRequest request,
        HttpServletResponse response) throws Throwable {
        request.setAttribute("message", "요청 파라미터로 명령어를 전달");
        return "/process.jsp";
      }
    }


  4)com.example.controller.Controller.java
    package com.example.controller;

    import java.io.*;
    import java.util.*;
    import javax.servlet.*;
    import javax.servlet.http.*;

    public class Controller extends HttpServlet {
      private Map commandMap = new HashMap();
      //명령어와 명령어 처리 클래스를 쌍으로 저장
      //명령어와 처리클래스가 매핑되어 있는 properties 파일을 읽어서 Map객체인 commandMap에 저장
      //명령어와 처리클래스가 매핑되어 있는 properties 파일은 Command.properties파일
```

```
131
132         public void init(ServletConfig config) throws ServletException {
133            String props = config.getInitParameter("propertyConfig");//web.xml에서 propertyConfig에
               해당하는 init-param 의 값을 읽어옴
134            Properties pr = new Properties();//명령어와 처리클래스의 매핑정보를 저장할 Properties객체 생성
135            FileInputStream f = null;
136            try {
137               f = new FileInputStream(props); //Command.properties파일의 내용을 읽어옴
138               pr.load(f);//Command.properties파일의 정보를  Properties객체에 저장
139            } catch (IOException e) {
140               throw new ServletException(e);
141            } finally {
142               if (f != null) try { f.close(); } catch(IOException ex) {}
143            }
144
145            Iterator keyIter = pr.keySet().iterator();//Iterator객체는 Enumeration객체를 확장시킨 개념의 객체
146            while( keyIter.hasNext() ) {//객체를 하나씩 꺼내서 그 객체명으로 Properties객체에 저장된 객체에 접근
147               String command = (String)keyIter.next();
148               String className = pr.getProperty(command);
149               try {
150                  Class commandClass = Class.forName(className);//해당 문자열을 클래스로 만든다.
151                  Object commandInstance = commandClass.newInstance();//해당클래스의 객체를 생성
152                  commandMap.put(command, commandInstance);// Map객체인 commandMap에 객체 저장
153               } catch (ClassNotFoundException e) {
154                  throw new ServletException(e);
155               } catch (InstantiationException e) {
156                  throw new ServletException(e);
157               } catch (IllegalAccessException e) {
158                  throw new ServletException(e);
159               }
160            }
161         }
162
163         public void doGet(//get방식의 서비스 메소드
164            HttpServletRequest request, HttpServletResponse response) throws ServletException,
               IOException {
165            requestPro(request, response);
166         }
167
168         protected void doPost(//post방식의 서비스 메소드
169            HttpServletRequest request, HttpServletResponse response) throws ServletException,
               IOException {
170            requestPro(request, response);
171         }
172
173         //시용자의 요청을 분석해서 해당 작업을 처리
174         private void requestPro(HttpServletRequest request, HttpServletResponse response)
175            throws ServletException, IOException {
176            String view = null;
177            CommandProcess com=null;
178            try {
179               String command = request.getParameter("command");
180               com = (CommandProcess)commandMap.get(command);
181               view = com.requestPro(request, response);
182            } catch(Throwable e) {
183               throw new ServletException(e);
184            }
185            RequestDispatcher dispatcher =request.getRequestDispatcher(view);
186            dispatcher.forward(request, response);
187         }
188      }
189
190
191   5)web.xml
192      <servlet>
193         <servlet-name>Controller</servlet-name>
194         <servlet-class>com.example.controller.Controller</servlet-class>
```

```
        <init-param>
            <param-name>propertyConfig</param-name>

            <param-value>C:/WebHome/Model2Demo/WebContent/command.properties</param-value
            >
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>Controller</servlet-name>
        <url-pattern>/servlets/Controller</url-pattern>
    </servlet-mapping>


6)process.jsp
    <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
    <!DOCTYPE html>
    <html lang="ko">
        <head>
            <meta charset="UTF-8">
            <title>요청 파라미터로 명령어를 전달하는 예제</title>
        </head>
        <body>
            처리 결과:
            <c:set var="message" value="${requestScope.message}" />
            <c:out value="${message}"/>
        </body>
    </html>



4. 요청 URI 자체를 명령어로 사용하는 방법
   http://localhost:8080/Model2Demo/message.do

   1)commandURI.properties
     /message.do=com.example.controller.MessageProcess

   2)CommandProcess.java <---이미 생성했음
   3)MessageProcess.java  <---이미 생성했음
   4)ControllerURI.java
     package com.example.controller;

     import java.io.*;
     import java.util.*;
     import javax.servlet.*;
     import javax.servlet.http.*;

     public class ControllerURI extends HttpServlet {

        private Map commandMap = new HashMap();//명령어와 명령어 처리 클래스를 쌍으로 저장

        //명령어와 처리클래스가 매핑되어 있는 properties 파일을 읽어서 Map객체인 commandMap에 저장
        //명령어와 처리클래스가 매핑되어 있는 properties 파일은 Command.properties파일
        public void init(ServletConfig config) throws ServletException {
            String props = config.getInitParameter("propertyConfig1");//web.xml에서 propertyConfig에
            해당하는 init-param 의 값을 읽어옴
            Properties pr = new Properties();//명령어와 처리클래스의 매핑정보를 저장할 Properties객체 생성
            FileInputStream f = null;
            try {
                f = new FileInputStream(props); //Command.properties파일의 내용을 읽어옴
                pr.load(f);//Command.properties파일의 정보를  Properties객체에 저장
            } catch (IOException e) {
                throw new ServletException(e);
            } finally {
                if (f != null) try { f.close(); } catch(IOException ex) {}
            }
            Iterator keyIter = pr.keySet().iterator();//Iterator객체는 Enumeration객체를 확장시킨 개념의 객체
            while( keyIter.hasNext() ) {//객체를 하나씩 꺼내서 그 객체명으로 Properties객체에 저장된 객체에 접근
```

```
            String command = (String)keyIter.next();
            String className = pr.getProperty(command);
            try {
               Class commandClass = Class.forName(className);//해당 문자열을 클래스로 만든다.
               Object commandInstance = commandClass.newInstance();//해당클래스의 객체를 생성
               commandMap.put(command, commandInstance);// Map객체인 commandMap에 객체 저장
            } catch (ClassNotFoundException e) {
               throw new ServletException(e);
            } catch (InstantiationException e) {
               throw new ServletException(e);
            } catch (IllegalAccessException e) {
               throw new ServletException(e);
            }
         }
      }

      public void doGet(//get방식의 서비스 메소드
         HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
         requestPro(request, response);
      }

      protected void doPost(//post방식의 서비스 메소드
         HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
         requestPro(request, response);
      }

      //시용자의 요청을 분석해서 해당 작업을 처리
      private void requestPro(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
         String view = null;
         CommandProcess com=null;
         try {
            String command = request.getRequestURI();
            if (command.indexOf(request.getContextPath()) == 0) {
              command = command.substring(request.getContextPath().length());
            }
            com = (CommandProcess)commandMap.get(command);
            view = com.requestPro(request, response);
         } catch(Throwable e) {
            throw new ServletException(e);
         }
         RequestDispatcher dispatcher =request.getRequestDispatcher(view);
         dispatcher.forward(request, response);
      }
   }


5)web.xml
   <servlet>
      <servlet-name>ControllerURI</servlet-name>
      <servlet-class>com.example.controller.ControllerURI</servlet-class>
      <init-param>
         <param-name>propertyConfig1</param-name>

         <param-value>C:/WebHome/Model2Demo/WebContent/commandURI.properties</param-v
         alue>
      </init-param>
   </servlet>

   <servlet-mapping>
      <servlet-name>ControllerURI</servlet-name>
      <url-pattern>*.do</url-pattern>
   </servlet-mapping>

6)process.jsp <---이미 생성했음.
```

5. Front Controller Pattern & Command Pattern 사용하기

1)Front Controller Pattern

-https://en.wikipedia.org/wiki/Front_controller

-https://yeonyeon.tistory.com/103

-https://www.oracle.com/java/technologies/front-controller.html

2)Command Pattern

-https://en.wikipedia.org/wiki/Command_pattern

-https://gmlwjd9405.github.io/2018/07/07/command-pattern.html

3)MVC Pattern.png, MVC Pattern1.png 참조

-Model

--Business Logic과 Data 접근 로직을 담당

--Java Code

-View

--User에게 보여주는 interface

--JSP가 담당

-Controller

--페이지의 흐름을 제어하는 역할

--Servlet이 담당

--Front Controller pattern & Command pattern 필요

4)ControllerServlet.java

```
package com.example.controller;

import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(name="ControllerServlet", urlPatterns = {"*.do"}, loadOnStartup = 0)
public class ControllerServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        this.processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        this.processRequest(request, response);
    }

    private void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException{
        try {
        String requestURI = request.getRequestURI();
            String contentPath = request.getContextPath();
            String commandURI = requestURI.substring(contentPath.length());

            System.out.println("requestURI = " + requestURI);
            System.out.println("contentPath = " + contentPath);
            System.out.println("commandURI = " + commandURI);

            CommandFactory factory = CommandFactory.getInstance();

            Command command = factory.createCommand(commandURI);
            ActionForward forward = command.execute(request, response);
```

```
            if(forward.isRedirect()) response.sendRedirect(request.getContextPath() +
            forward.getPath());
            else {
               RequestDispatcher dispatcher = request.getRequestDispatcher(forward.getPath());
               dispatcher.forward(request, response);
            }
         } catch (Exception e) {
            request.setAttribute("exception", e);
            RequestDispatcher dispatcher = request.getRequestDispatcher("/error.jsp");
            dispatcher.forward(request, response);
         }
      }
   }
```

5)command.java

```
   package com.example.controller;

   import java.io.IOException;
   import javax.servlet.ServletException;
   import javax.servlet.http.HttpServletRequest;
   import javax.servlet.http.HttpServletResponse;

   public interface Command {
      ActionForward execute(HttpServletRequest request, HttpServletResponse response) throws
      ServletException, IOException;
   }
```

6)AcdtionForward.java

```
   package com.example.controller;

   public class ActionForward {
      private String path;
      private boolean isRedirect;

      {
         this.path = null;
         this.isRedirect = true;
      }

      public ActionForward() {}

      public ActionForward(String path, boolean isRedirect) {
         this.path = path;
         this.isRedirect = isRedirect;
      }

      public String getPath() {
         return path;
      }

      public void setPath(String path) {
         this.path = path;
      }

      public boolean isRedirect() {
         return isRedirect;
      }

      public void setRedirect(boolean isRedirect) {
         this.isRedirect = isRedirect;
      }
   }
```

7)LoginActionCommand.java

```
package com.example.controller;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class LoginActionCommand implements Command {

    @Override
    public ActionForward execute(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        ActionForward forward = new ActionForward("/loginaction.jsp", true);
        return forward;
    }
}
```

8)CommandFactory.java

```
package com.example.controller;

import java.lang.reflect.Constructor;
import java.util.HashMap;
import java.util.Map;

public class CommandFactory {
    private static CommandFactory factory;
    private Map<String, String> map;

    {
        this.factory = null;
        this.map = new HashMap<String, String>();
    }
    private CommandFactory() {
        this.map.put("/loginaction.do", "com.example.controller.LoginActionCommand");
    }

    public static CommandFactory getInstance() {
        if(factory == null) factory = new CommandFactory();
        return factory;
    }

    public Command createCommand(String commandURI) throws Exception{
        String className = this.map.get(commandURI);
        if(className == null) return null;
        try {
            Class<?> cls = Class.forName(className);
            Constructor<?> constructor = cls.getConstructor(null);
            Command command = (Command)constructor.newInstance();
            return command;
        } catch (Exception e) {
            throw e;
        }
    }
}
```

9)error.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" isErrorPage="true"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<h3>Error Info</h3>
<c:forEach var="stack" items="${requestScope.exception.stackTrace}">
```

```
520             ${stack}
521         </c:forEach>
```