```
Model Model2 Pattern
 1
 2
    1. MVC
 3
 4
      -http://localhost:8080/Model2Demo/servlets/MessageServlet?message=name
 5
      -messageView.jsp 로 포워딩된다.
 6
      -이 방법은 간단하지만, 명령어가 파라미터로 전달되게 되면 정보가 웹 브라우저를 통해 노출이 되어 악의적으로 사이트에 접근할
      수 있는 빌미를 제공한다는 단점이 있다.
 7
 8
      1)com.exmple.controller.MessageController.java
 9
         public class MessageController extends HttpServlet {
                                                              //1단계 : 사용자의 요청을 받는 서비스 메소드
10
           public void doGet(HttpServletRequest request,
11
                                  HttpServletResponse response) throws ServletException, IOException
                                   {
12
              requestPro(request, response);
           }
13
14
15
           public void doPost(HttpServletRequest request, //1단계 : 사용자의 요청을 받는 서비스 메소드
16
                                  HttpServletResponse response) throws ServletException, IOException
17
              requestPro(request, response);
           }
18
19
20
           private void requestPro(HttpServletRequest request, HttpServletResponse response)
21
              throws ServletException ,IOException{
              String message = request.getParameter("message");//2단계 : 사용자의 요청분석
22
23
              Object result = null;
              if (message == null || message.equals("base")) {//3단계 : 사용자의 요청에 따른 작업처리
24
25
                 result = "하하하.";
26
              } else if (message.equals("name")) {
27
                 result = "한지민 입니다.";
28
              } else {
                 result = "타입이 맞지 않습니다.";
29
30
31
              request.setAttribute("result", result);//4단계: request의 속성에 처리결과 저장
32
33
              // 5단계: RequestDispatcher를 사용하여 해당 뷰로 포워딩
              RequestDispatcher dispatcher = request.getRequestDispatcher("/messageView.jsp");
34
35
              dispatcher.forward(request, response);
36
           }
37
         }
38
39
40
      2)messageView.jsp
       <@ page language="java" contentType="text/html; charset=UTF-8"
41
42
              pageEncoding="UTF-8"%>
       < @ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
43
44
       <!DOCTYPE html>
       <html lang="ko">
45
46
         <head>
            <meta charset="UTF-8">
47
48
            <title>간단한 Controller 의 사용예제</title>
49
         </head>
50
         <body>
51
           결과:
            <c:set var="result" value="${requestScope.result}" />
52
53
            <c:out value="${result}"/>
54
         </body>
55
      </html>
56
57
58
      3)web.xml
59
         <servlet>
60
            <servlet-name>MessageServlet</servlet-name>
            <servlet-class>com.example.controller.MessageController</servlet-class>
61
62
         </servlet>
63
         <servlet-mapping>
64
            <servlet-name>MessageServlet</servlet-name>
```

```
65
             <url-pattern>/servlets/MessageServlet</url-pattern>
 66
          </servlet-mapping>
 67
 68
 69
 70
     2. Command Pattern
 71
       1)요청을 커맨드 메시지로 전달
 72
       2) 두가지 방법
 73
          a. servlet 에 요청 파라미터를 덧붙여 전달 : 윗방법
 74
            -ex)MessageController?message=name
 75
            -간편하다. 하지만, 파라미터가 노출된다.
 76
            -악의적 접근 빌미 제공
 77
 78
          b. 요청한 URL 자체를 명령어로 제출
 79
            -http://localhost:8080/Model2Demo/test.do
 80
 81
 82
     3. 요청 파라미터로 명령어를 전달하는 방법
 83
       -http://localhost:8080/Model2Demo/servlets/Controller?command=Message
 84
 85
       1)command.properties
          Message=com.javasoft.libs.Controller.MessageProcess
 86
 87
 88
 89
       2)com.example.controller.CommandProcess.java
 90
          package com.example.controller;
 91
 92
          import javax.servlet.http.HttpServletRequest;
 93
          import javax.servlet.http.HttpServletResponse;
 94
 95
          //요청 파라미터로 명령어를 전달하는 방식의 슈퍼 인터페이스
 96
          public interface CommandProcess {
            public String requestPro(HttpServletRequest request, HttpServletResponse response) throws
 97
            Throwable;
          }
 98
 99
100
101
       3)com.example.controller.MessageProcess.java
102
          package com.example.controller;
103
104
          import javax.servlet.http.HttpServletReguest;
105
          import javax.servlet.http.HttpServletResponse;
106
107
          public class MessageProcess implements CommandProcess {
108
109
            @Override
110
            public String requestPro(HttpServletRequest request,
111
               HttpServletResponse response) throws Throwable {
112
               request.setAttribute("message", "요청 파라미터로 명령어를 전달");
113
               return "/process.jsp";
114
115
          }
116
117
       4)com.example.controller.Controller.java
118
119
          package com.example.controller;
120
121
          import java.io.*;
122
          import java.util.*;
123
          import javax.servlet.*;
124
          import javax.servlet.http.*;
125
126
          public class Controller extends HttpServlet {
127
            private Map commandMap = new HashMap();
128
            //명령어와 명령어 처리 클래스를 쌍으로 저장
129
            //명령어와 처리클래스가 매핑되어 있는 properties 파일을 읽어서 Map객체인 commandMap에 저장
130
            //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 {
                  f = new FileInputStream(props); //Command.properties파일의 내용을 읽어옴
137
138
                  pr.load(f);//Command.properties파일의 정보를 Properties객체에 저장
139
               } catch (IOException e) {
140
                  throw new ServletException(e);
               } finally {
141
142
                  if (f != null) try { f.close(); } catch(IOException ex) {}
143
144
145
               Iterator keyIter = pr.keySet().iterator();//Iterator객체는 Enumeration객체를 확장시킨 개념의 객체
               while(keyIter.hasNext()) {//객체를 하나씩 꺼내서 그 객체명으로 Properties객체에 저장된 객체에 접근
146
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);
                  } catch (InstantiationException e) {
155
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;
               CommandProcess com=null;
177
178
               trv {
179
                  String command = request.getParameter("command");
180
                  com = (CommandProcess)commandMap.get(command);
181
                  view = com.requestPro(request, response);
               } catch(Throwable e) {
182
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>
```

```
195
            <init-param>
196
               <param-name>propertyConfig</param-name>
197
               <param-value>C:/WebHome/Model2Demo/WebContent/command.properties</param-value</pre>
             </init-param>
198
199
          </servlet>
200
          <servlet-mapping>
            <servlet-name>Controller</servlet-name>
201
202
             <url-pattern>/servlets/Controller</url-pattern>
203
          </servlet-mapping>
204
205
206
       6)process.jsp
207
          <@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
208
          <!DOCTYPE html>
209
          <html lang="ko">
210
             <head>
211
               <meta charset="UTF-8">
212
               <title>요청 파라미터로 명령어를 전달하는 예제</title>
213
             </head>
214
            <body>
215
               처리 결과:
216
               <c:set var="message" value="${requestScope.message}" />
217
               <c:out value="${message}"/>
218
             </body>
219
          </html>
220
221
222
223
     4. 요청 URI 자체를 명령어로 사용하는 방법
224
       http://localhost:8080/Model2Demo/message.do
225
226
       1)commandURI.properties
227
          /message.do=com.example.controller.MessageProcess
228
       2)CommandProcess.java <---이미 생성했음
229
230
       3)MessageProcess.java <---이미 생성했음
231
       4)ControllerURI.java
232
          package com.example.controller;
233
234
          import java.io.*;
235
          import java.util.*;
236
          import javax.servlet.*;
237
          import javax.servlet.http.*;
238
239
          public class ControllerURI extends HttpServlet {
240
241
            private Map commandMap = new HashMap();//명령어와 명령어 처리 클래스를 쌍으로 저장
242
            //명령어와 처리클래스가 매핑되어 있는 properties 파일을 읽어서 Map객체인 commandMap에 저장
243
244
            //명령어와 처리클래스가 매핑되어 있는 properties 파일은 Command.properties파일
245
            public void init(ServletConfig config) throws ServletException {
246
               String props = config.getInitParameter("propertyConfig1");//web.xml에서 propertyConfig에
               해당하는 init-param 의 값을 읽어옴
247
               Properties pr = new Properties();//명령어와 처리클래스의 매핑정보를 저장할 Properties객체 생성
248
               FileInputStream f = null;
249
               try {
250
                 f = new FileInputStream(props); //Command.properties파일의 내용을 읽어옴
251
                 pr.load(f);//Command.properties파일의 정보를 Properties객체에 저장
252
               } catch (IOException e) {
253
                 throw new ServletException(e);
               } finally {
254
255
                 if (f!= null) try { f.close(); } catch(IOException ex) {}
256
257
               Iterator keyIter = pr.keySet().iterator();//Iterator객체는 Enumeration객체를 확장시킨 개념의 객체
258
               while(keyIter.hasNext()) {//객체를 하나씩 꺼내서 그 객체명으로 Properties객체에 저장된 객체에 접근
```

```
259
                  String command = (String)keyIter.next();
260
                  String className = pr.getProperty(command);
261
                  try {
262
                    Class commandClass = Class.forName(className);//해당 문자열을 클래스로 만든다.
263
                    Object commandInstance = commandClass.newInstance();//해당클래스의 객체를 생성
                    commandMap.put(command, commandInstance);// Map객체인 commandMap에 객체 저장
264
265
                  } catch (ClassNotFoundException e) {
266
                    throw new ServletException(e);
267
                  } catch (InstantiationException e) {
268
                    throw new ServletException(e);
269
                  } catch (IllegalAccessException e) {
270
                    throw new ServletException(e);
271
                  }
272
               }
273
             }
274
275
             public void doGet(//get방식의 서비스 메소드
276
               HttpServletRequest request, HttpServletResponse response)
277
             throws ServletException, IOException {
278
               requestPro(request, response);
             }
279
280
281
             protected void doPost(//post방식의 서비스 메소드
282
               HttpServletRequest request, HttpServletResponse response)
283
             throws ServletException, IOException {
284
               requestPro(request, response);
285
             }
286
287
             //시용자의 요청을 분석해서 해당 작업을 처리
288
             private void requestPro(HttpServletRequest request, HttpServletResponse response)
289
             throws ServletException, IOException {
290
               String view = null;
291
               CommandProcess com=null;
292
               try {
293
                  String command = request.getRequestURI();
294
                  if (command.indexOf(request.getContextPath()) == 0) {
295
                    command = command.substring(request.getContextPath().length());
296
297
                  com = (CommandProcess)commandMap.get(command);
298
                  view = com.requestPro(request, response);
               } catch(Throwable e) {
299
300
                  throw new ServletException(e);
301
302
               RequestDispatcher dispatcher = request.getRequestDispatcher(view);
303
               dispatcher.forward(request, response);
304
             }
305
          }
306
307
308
        5)web.xml
309
          <servlet>
310
             <servlet-name>ControllerURI</servlet-name>
311
             <servlet-class>com.example.controller.ControllerURI</servlet-class>
312
             <init-param>
313
                <param-name>propertyConfig1</param-name>
314
                <param-value>C:/WebHome/Model2Demo/WebContent/commandURI.properties</param-v
               alue>
315
             </init-param>
316
          </servlet>
317
318
          <servlet-mapping>
319
             <servlet-name>ControllerURI</servlet-name>
320
             <url-pattern>*.do</url-pattern>
321
          </servlet-mapping>
322
323
        6)process.jsp <---이미 생성했음.
```

```
325
326
     5. Front Controller Pattern & Command Pattern 사용하기
327
        1)Front Controller Pattern
328
          -https://en.wikipedia.org/wiki/Front_controller
329
          -https://yeonyeon.tistory.com/103
330
          -https://www.oracle.com/java/technologies/front-controller.html
331
332
        2)Command Pattern
333
          -https://en.wikipedia.org/wiki/Command_pattern
334
          -https://gmlwid9405.github.io/2018/07/07/command-pattern.html
335
336
        3)MVC Pattern.png, MVC Pattern1.png 참조
337
          -Model
338
             --Business Logic과 Data 접근 로직을 담당
339
             --Java Code
340
341
          -View
342
             --User에게 보여주는 interface
343
             --JSP가 담당
344
345
          -Controller
346
             --페이지의 흐름을 제어하는 역할
347
             --Servlet이 담당
348
             --Front Controller pattern & Command pattern 필요
349
350
        4)ControllerServlet.java
351
352
          package com.example.controller;
353
354
          import java.io.IOException;
          import javax.servlet.RequestDispatcher;
355
356
          import javax.servlet.ServletException;
357
          import javax.servlet.annotation.WebServlet;
358
          import javax.servlet.http.HttpServlet;
359
          import javax.servlet.http.HttpServletRequest;
360
          import javax.servlet.http.HttpServletResponse;
361
          @WebServlet(name="ControllerServlet", urlPatterns = {"*.do"}, loadOnStartup = 0)
362
363
          public class ControllerServlet extends HttpServlet {
             @Override
364
             protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
365
             ServletException, IOException {
366
               this.processRequest(request, response);
367
             }
368
369
             @Override
             protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
370
             ServletException, IOException {
371
               this.processRequest(request, response);
372
             }
373
374
             private void processRequest(HttpServletRequest request, HttpServletResponse response)
             throws ServletException, IOException{
375
376
               String requestURI = request.getRequestURI();
377
                  String contentPath = request.getContextPath();
378
                  String commandURI = requestURI.substring(contentPath.length());
379
380
                  System.out.println("requestURI = " + requestURI);
                  System.out.println("contentPath = " + contentPath);
381
                  System.out.println("commandURI = " + commandURI);
382
383
384
                  CommandFactory factory = CommandFactory.getInstance();
385
                  Command command = factory.createCommand(commandURI);
386
                  ActionForward forward = command.execute(request, response);
387
```

324

```
388
                   if(forward.isRedirect()) response.sendRedirect(request.getContextPath() +
                   forward.getPath());
389
                   else {
390
                      RequestDispatcher dispatcher = request.getRequestDispatcher(forward.getPath());
391
                      dispatcher.forward(request, response);
392
393
                } catch (Exception e) {
                   request.setAttribute("exception", e);
394
395
                   RequestDispatcher dispatcher = request.getRequestDispatcher("/error.jsp");
396
                   dispatcher.forward(request, response);
397
                }
398
             }
399
           }
400
401
402
        5)command.java
403
404
           package com.example.controller;
405
406
           import java.io.IOException;
407
           import javax.servlet.ServletException;
408
           import javax.servlet.http.HttpServletRequest;
409
           import javax.servlet.http.HttpServletResponse;
410
411
           public interface Command {
412
              ActionForward execute(HttpServletRequest request, HttpServletResponse response) throws
              ServletException, IOException;
413
           }
414
415
416
        6)AcdtionForward.java
417
           package com.example.controller;
418
419
           public class ActionForward {
420
              private String path;
421
              private boolean isRedirect;
422
423
424
                this.path = null;
425
                this.isRedirect = true;
426
              }
427
428
              public ActionForward() {}
429
430
              public ActionForward(String path, boolean isRedirect) {
431
                this.path = path;
432
                this.isRedirect = isRedirect;
433
              }
434
435
              public String getPath() {
436
                return path;
437
              }
438
439
              public void setPath(String path) {
440
                this.path = path;
441
442
443
              public boolean isRedirect() {
444
                return isRedirect;
445
446
447
              public void setRedirect(boolean isRedirect) {
448
                this.isRedirect = isRedirect;
449
              }
450
           }
451
452
```

```
453
        7)LoginActionCommand.java
454
455
           package com.example.controller;
456
457
          import java.io.IOException;
458
459
          import javax.servlet.ServletException;
460
          import javax.servlet.http.HttpServletRequest;
461
          import javax.servlet.http.HttpServletResponse;
462
463
           public class LoginActionCommand implements Command {
464
465
             @Override
466
             public ActionForward execute(HttpServletRequest request, HttpServletResponse response)
                  throws ServletException, IOException {
467
468
                ActionForward forward = new ActionForward("/loginaction.jsp", true);
469
                return forward;
470
             }
471
          }
472
473
474
        8)CommandFactory.java
475
476
          package com.example.controller;
477
478
          import java.lang.reflect.Constructor;
479
          import java.util.HashMap;
480
          import java.util.Map;
481
482
          public class CommandFactory {
483
             private static CommandFactory factory;
484
             private Map<String, String> map;
485
486
487
                this.factory = null;
488
                this.map = new HashMap<String, String>();
489
             }
490
             private CommandFactory() {
491
                this.map.put("/loginaction.do", "com.example.controller.LoginActionCommand");
492
493
494
             public static CommandFactory getInstance() {
495
                if(factory == null) factory = new CommandFactory();
496
                return factory;
497
             }
498
499
             public Command createCommand(String commandURI) throws Exception{
500
                String className = this.map.get(commandURI);
501
                if(className == null) return null;
502
                try {
503
                  Class<?> cls = Class.forName(className);
504
                  Constructor<?> constructor = cls.getConstructor(null);
505
                  Command command = (Command)constructor.newInstance();
506
                  return command;
507
                } catch (Exception e) {
508
                  throw e;
509
510
             }
511
          }
512
513
        9)error.jsp
514
515
           <@ page language="java" contentType="text/html; charset=UTF-8"
516
             pageEncoding="UTF-8" isErrorPage="true"%>
517
           <@@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
518
           <h3>Error Info</h3>
           <c:forEach var="stack" items="${requestScope.exception.stackTrace}">
519
```

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