

Homework 4 for CSE 216: Sequence Diagram and Design Class Diagram

Due date: Part 1 is due on **Feb 16th (Monday) 10am in class**. Part 2 is due on **Feb 18th (Wednesday) 10am in class**.

1 Part 1: Sequence Diagram (10 points)

Your task in this part is to create a Sequence Diagram by “reverse engineering” the set of classes supplied below.

- The “found” message that starts the sequence is a `doGet(req,res)` message sent to the `ControllerServlet` class.
- Ignore any code in the catch blocks.
- Use the notation presented in Larman Ch 15.
- Include parameters and return values in the messages.
- Use the return value = message name(parameters) message form to show return values; don’t draw a separate line.
- In the lifeline boxes, always use the class name and use named instances where appropriate.
- Getting the sequence of messages correct is much more important than notational details like filling in the arrow heads on the message arrows.

```
1  /*
2  *  ControllerServlet.java
3  */
4
5  package edu.lehigh.cse216.project.controller;
```

```

6
7 import java.io.*;
8 import java.net.*;
9 import java.util.MissingResourceException;
10 import java.util.ResourceBundle;
11
12 import javax.servlet.*;
13 import javax.servlet.http.*;
14
15 public class ControllerServlet extends HttpServlet {
16     private ActionFactory factory = new ActionFactory();
17
18     public void init(ServletConfig config) throws ServletException {
19         super.init(config);
20         ResourceBundle bundle = null;
21         try {
22             bundle = ResourceBundle.getBundle("actions");
23         } catch (MissingResourceException e) {
24             throw new ServletException(e);
25         }
26         getServletContext().setAttribute("actionmappings", bundle);
27     }
28
29     protected void processRequest
30     (HttpServletRequest req, HttpServletResponse res)
31     throws ServletException, IOException {
32         try {
33             String actionClassName = getActionClass(req);
34             Action action =
35                 factory.getAction
36                 (actionClassName, getClass().getClassLoader());
37             ActionRouter router = action.perform(this, req, res);
38             router.route(this, req, res);
39         } catch (Exception ex) {
40             throw new ServletException(ex);
41         }
42     }
43
44     private String getClassName(HttpServletRequest req) {
45         String path = req.getServletPath();

```

```

46     int slash = path.lastIndexOf('/'),
47     period = path.lastIndexOf('.');
48     if (period > 0 && period > slash) {
49         path = path.substring(slash + 1, period);
50     }
51     return path;
52 }
53
54 private String getActionClass(HttpServletRequest req) {
55     ResourceBundle bundle = (ResourceBundle)
56         getServletContext().getAttribute("actionmappings");
57     return (String) bundle.getObject(getActionKey(req));
58 }
59
60 private String getActionKey(HttpServletRequest req) {
61     return getClassName(req);
62 }
63
64 protected void doGet
65     (HttpServletRequest request, HttpServletResponse response)
66     throws ServletException, IOException {
67     processRequest(request, response);
68 }
69
70 protected void doPost
71     (HttpServletRequest request, HttpServletResponse response)
72     throws ServletException, IOException {
73     processRequest(request, response);
74 }
75
76 public String getServletInfo() {
77     return "Controller Servlet";
78 }
79 }
80
81
82 /*
83 * ActionRouter.java
84 *
85 */

```

```

86
87 package edu.lehigh.cse216.project.controller;
88
89 import java.io.IOException;
90 import java.util.ResourceBundle;
91 import javax.servlet.ServletException;
92 import javax.servlet.http.HttpServlet;
93 import javax.servlet.http.HttpServletRequest;
94 import javax.servlet.http.HttpServletResponse;
95
96 public class ActionRouter {
97     private String key;
98     private final boolean isForward;
99
100    public ActionRouter(String key) {
101        this(key, true);
102    }
103
104    public ActionRouter(String key, boolean isForward) {
105        this.key = key;
106        this.isForward = isForward;
107    }
108
109    public void route(HttpServletRequest servlet,
110                      HttpServletRequest req,
111                      HttpServletResponse res)
112        throws ServletException, IOException {
113
114        ResourceBundle bundle = (ResourceBundle)
115            servlet.getServletContext().getAttribute("actionmappings");
116        String url = (String) bundle.getObject(key);
117
118        if (isForward) {
119            servlet.getServletContext().
120                getRequestDispatcher(res.encodeURL(url)).forward(req,
121        } else {
122            res.sendRedirect(res.encodeRedirectURL(url));
123        }
124    }
125

```

```

126 }
127
128 /*
129 * Action.java
130 *
131 */
132
133 package edu.lehigh.cse216.project.controller;
134
135 import java.io.IOException;
136 import javax.servlet.ServletException;
137 import javax.servlet.http.HttpServlet;
138 import javax.servlet.http.HttpServletRequest;
139 import javax.servlet.http.HttpServletResponse;
140
141 public interface Action {
142     public ActionRouter perform(HttpServletRequest servlet ,
143                               HttpServletRequest req ,
144                               HttpServletResponse res)
145     throws IOException , ServletException;
146 }
147
148 /*
149 * ActionFactory.java
150 *
151 */
152
153 package edu.lehigh.cse216.project.controller;
154
155 import java.util.Hashtable;
156
157 public class ActionFactory {
158     private Hashtable<String ,Action> actions =
159         new Hashtable<String ,Action>();
160
161     public ActionFactory() {
162     }
163
164     public Action getAction(String className, ClassLoader loader)
165     throws ClassNotFoundException , IllegalAccessException ,

```

```
166     InstantiationException {
167         Action action = actions.get(className);
168         if (action == null) {
169             Class klass = loader.loadClass(className);
170             action = (Action) klass.newInstance();
171             actions.put(className, action);
172         }
173         return action;
174     }
175 }
```

2 Part 2: Design Class Diagram (10 points)

Your task is to create a UML Class Diagram by “reverse engineering” the set of classes supplied in part 1. Use the notation presented in Larman Ch 16.

3 Submission

Please use pencil and paper to create both models and turn part 1 and 2 in on paper at the beginning of the class on Feb 16th and 18th, respectively.