

## ComputeAnalytics.java

```
1 package analytics;
2
3 import java.io.File;
18
19 public class ComputeAnalytics {
20
21     public static synchronized void
computeOrders(HttpServletRequest req) {
22         HttpSession s = req.getSession();
23         if (s.getAttribute("order") == null) return;
24         Order order = (Order) s.getAttribute("order");
25         order.computeAllCosts();
26     }
27
28     public static synchronized void
computeUsersAnalytics(HttpServletRequest req) {
29         if (req.getAttribute("users") == null) return;
30
31         HttpSession s = req.getSession();
32         Account user = (Account) s.getAttribute("AUTH");
33         Map<String, long[]> accounts = (HashMap<String, long[]>)
req.getAttribute("users");
34         long[] avgTimes = new long[4];
35
36         if (req.getAttribute("allCheckoutAvgTime") == null) {
37             req.setAttribute("allCheckoutAvgTime", 0);
38         }
39
40         if (req.getAttribute("allAddToCartAvgTime") == null) {
41             req.setAttribute("allAddToCartAvgTime", 0);
42         }
43
44         if (Utils.isValidUser(user) &&
45             !accounts.containsKey(s.getId()) &&
46             s.getAttribute("checkoutAvgTime") != null &&
47             s.getAttribute("addToCartAvgTime") != null)
48         {
49             avgTimes[0] = (long)
s.getAttribute("checkoutAvgTime");
50             avgTimes[1] = (long)
s.getAttribute("addToCartAvgTime");
51             accounts.put(s.getId(), avgTimes);
52         }
53
```

## ComputeAnalytics.java

```
54     int count = 1;
55     for (long[] accountAvgTimes: accounts.values()) {
56         long allCheckoutAvgTime = accountAvgTimes[0];
57         long allAddToCartAvgTime = accountAvgTimes[1];
58
59         req.setAttribute("allCheckoutAvgTime",
60             allCheckoutAvgTime/count);
61         req.setAttribute("allAddToCartAvgTime",
62             allAddToCartAvgTime/count);
63     }
64 }
65
66 public static synchronized void
67 addUserOrders(HttpServletRequest req) {
68     HttpSession currentSession = req.getSession();
69     Account ac = (Account)
70     currentSession.getAttribute("AUTH");
71     List<String> orders = (ArrayList<String>)
72     currentSession.getAttribute("orders");
73     String poDir = (String) req.getAttribute("poDir");
74
75     if (orders != null) {
76         OrderDAO orderDao = new OrderDAO(new File(poDir));
77         req.setAttribute("orderDao", new OrderDAO(new
78             File(poDir)));
79         req.setAttribute("totalAllOrders",
80             orderDao.getP0s().length);
81
82         if (ac != null && Utils.isValidUser(ac)) {
83             currentSession.setAttribute("totalOrders",
84                 orderDao.getP0s(ac.getUsername()).length);
85
86             for (File f: orderDao.getP0s(ac.getUsername())) {
87                 if (!orders.contains(f.getAbsolutePath())) {
88                     orders.add(f.getAbsolutePath());
89                 }
90             }
91         }
92     }
93 }
```

## ComputeAnalytics.java

```
90     public static synchronized void
computeAvgStartCheckoutTime(HttpServletRequest req) {
91         HttpSession currentSession = req.getSession();
92         String url = req.getRequestURI();
93         String path = url.replaceAll("/eFoods/", "");
94
95         long checkoutAvgTime = 0;
96         int checkoutAvgCount = 0;
97         Order order = (Order)
currentSession.getAttribute("order");
98
99         Map<String, long[]> accounts = (HashMap<String, long[]>)
req.getAttribute("users");
100
101         if (path.toLowerCase().contains("checkout") &&
102             req.getParameter("confirm") != null &&
103             !order.getItems().isEmpty() &&
104             currentSession.getAttribute("checkoutTime") ==
null)
105         {
106             currentSession.setAttribute("checkoutTime", new
Date());
107
108             if (currentSession.getAttribute("checkoutAvgTime") !=
null) {
109                 checkoutAvgTime = (long)
currentSession.getAttribute("checkoutAvgTime");
110             }
111
112             if (currentSession.getAttribute("checkoutAvgCount") !=
null) {
113                 checkoutAvgCount = (int)
currentSession.getAttribute("checkoutAvgCount");
114             }
115
116             Date startTime = (Date)
currentSession.getAttribute("startTime");
117             Date checkoutTime = (Date)
currentSession.getAttribute("checkoutTime");
118
119             long diff = checkoutTime.getTime() -
startTime.getTime();
120             long seconds = TimeUnit.MILLISECONDS.toSeconds(diff);
121
```

## ComputeAnalytics.java

```
122         checkoutAvgCount++;
123         checkoutAvgTime = (checkoutAvgTime + seconds) /
checkoutAvgCount;
124
125         currentSession.setAttribute("checkoutAvgTime",
checkoutAvgTime);
126         currentSession.setAttribute("checkoutAvgCount",
checkoutAvgCount);
127
128         accounts.put(currentSession.getId(), new long[]{(long)
currentSession.getAttribute("checkoutAvgTime")});
129     }
130 }
131
132 public static synchronized void
computeAvgAddToCartTime(HttpServletRequest req) {
133     HttpSession currentSession = req.getSession();
134     String url = req.getRequestURI();
135     String path = url.replaceAll("/eFoods/", "");
136
137     long addToCartAvgTime = 0;
138     int addToCartAvgCount = 0;
139     Order order = (Order)
currentSession.getAttribute("order");
140     Map<String, long[]> accounts = (HashMap<String, long[]>)
req.getAttribute("users");
141
142     if (path.toLowerCase().contains("api/cart") &&
143         req.getParameter("addToCart") != null &&
144         order.getItems().isEmpty() &&
145         currentSession.getAttribute("addToCartTime") ==
null)
146     {
147         currentSession.setAttribute("addToCartTime", new
Date());
148
149         if (currentSession.getAttribute("addToCartAvgTime") !=
null) {
150             addToCartAvgTime = (long)
currentSession.getAttribute("addToCartAvgTime");
151         }
152
153         if (currentSession.getAttribute("addToCartAvgCount") !=
null) {
```

## ComputeAnalytics.java

```
154         addToCartAvgCount = (int)
currentSession.getAttribute("addToCartAvgCount");
155     }
156
157     Date startTime = (Date)
currentSession.getAttribute("startTime");
158     Date addToCartTime = (Date)
currentSession.getAttribute("addToCartTime");
159
160     long diff = addToCartTime.getTime() -
startTime.getTime();
161     long seconds = TimeUnit.MILLISECONDS.toSeconds(diff);
162
163     addToCartAvgCount++;
164     addToCartAvgTime = (addToCartAvgTime + seconds) /
addToCartAvgCount;
165
166     currentSession.setAttribute("addToCartAvgTime",
addToCartAvgTime);
167     currentSession.setAttribute("addToCartAvgCount",
addToCartAvgCount);
168
169     accounts.put(currentSession.getId(), new long[]{(long)
currentSession.getAttribute("addToCartAvgTime")});
170     }
171 }
172 }
173
```