Event Driven Programming in Java Example – Part 1

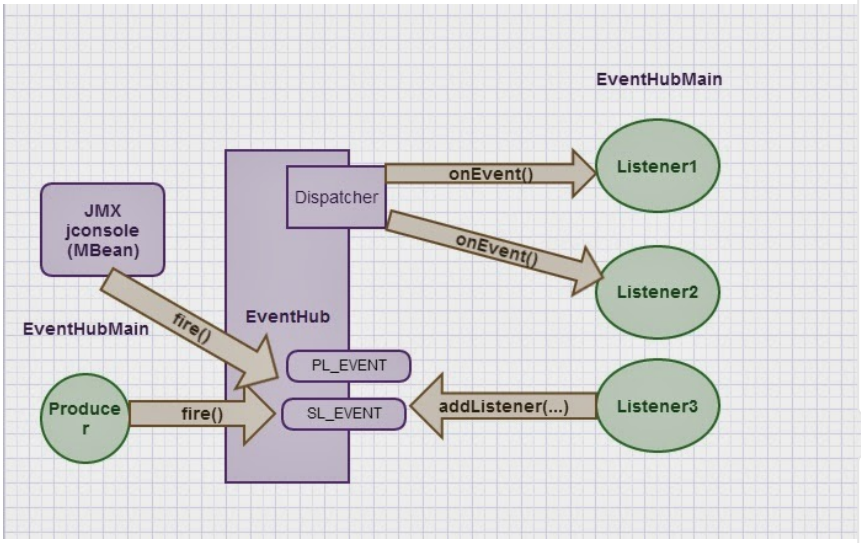
Posted on December 3, 2014 by Arulkumaran Kumaraswamipillai — No Comments ↓

Event Driven Architecture aka EDA loosely couples event producers and event consumers.

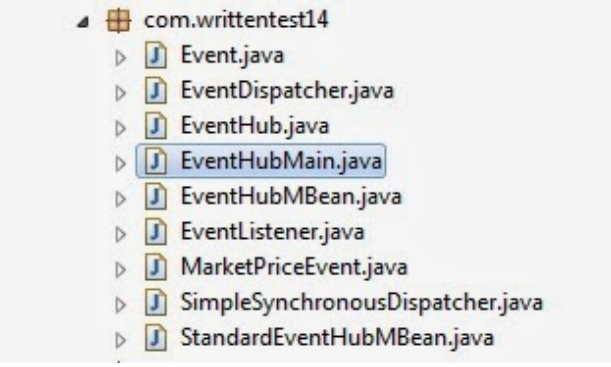
An event can be defined as “a change in state”. For example, when an event producer fires an event to notify all its registered listeners that either “securities” or “security prices” have been loaded, the listeners are notified to update their data via a synchronous or asynchronous dispatcher. Both the “Event” producers and listeners are loosely coupled via an “***EventHub***” and “**Event**“. An “*EventHub*” is used to register and unregister listeners.

The "*EventHub*" can be registed as a JMX bean to control behaviors at runtime via a jconsole like firing an event, count number of events, etc.

A number of tutorials will take you through writing event driven code in Java along with registering as MBean to interact via a **JMX** compliant tool like jconsole.



Let's define the interfaces and implementation classes.



Step 1: Define the "Event" class from which all other events can be derived from.

```
1  package com.writtentest14;
2
3  import java.util.Date;
4
5  public class Event {
6
7   private String id;
8   private Date timeStamp;
9
10  public Event(String id) {
```


16 Technical Key Areas

```
11    super();
12    this.id = id;
13    this.timeStamp = new Date();
14   }
15
16   public String getId() {
17    return id;
18   }
19
20   public void setId(String id) {
21    this.id = id;
22   }
23
24   public Date getTimeStamp() {
25    return timeStamp;
26   }
27
28   public void setTimeStamp(Date timeStamp) {
29    this.timeStamp = timeStamp;
30   }
31
32 }
33
```

Step 2: Define the interface for the listeners

```
1 package com.writtentest14;
2
3 public interface EventListener<T extends Event> {
4  void onEvent(T event);
5 }
6
```

Step 3: Define the dispatcher interface.

```
1  package com.writtentest14;
2
3  import java.util.List;
4
5
6  public interface EventDispatcher {
7
8      void dispatch(Event event, List<EventListene
9  }
10
```

Step 4: The dispatcher implementation. It could be synchronous or asynchronous dispatcher. Let's keep it simple by defining a synchronous dispatcher.

```
1  package com.writtentest14;
2
3  import java.util.List;
4
```

```
5  public class SimpleSynchronousDispatcher impleme
6
7      @Override
8      public void dispatch(Event event, List<Event
9          for (EventListener<Event> listener : lis
10             listener.onEvent(event);
11         }
12     }
13 }
14
```

Step 5: Define the ***EventHub***. Binds and unbinds listeners and invokes the dispatcher to dispatch the events.

```
1  package com.writtentest14;
2
3  import java.util.Collections;
4  import java.util.List;
5  import java.util.concurrent.ConcurrentHashMap;
6  import java.util.concurrent.ConcurrentMap;
7  import java.util.concurrent.CopyOnWriteArrayList
8  import java.util.concurrent.atomic.AtomicLong;
9
10 /**
11  * register and unregister event listeners
12  */
13
14 public class EventHub {
15
16  private static final EventHub INSTANCE = create
17
18  private ConcurrentMap<String, List<EventListene
19                                    new Concurren
20
21  private EventDispatcher synchronousDispatcher;
22
23  private AtomicLong eventCount = new AtomicLong(
24
25  public EventHub() {
26  }
27
28  public static EventHub instance() {
29   return INSTANCE;
30  }
31
32  public EventHub(EventDispatcher synchronousDisp
33   this.synchronousDispatcher = synchronousDispat
34  }
35
36  public long getEventCount() {
37   return this.eventCount.get();
38  }
39
40  private long getNextEventNumber() {
41   return this.eventCount.incrementAndGet();
42  }
43
44  protected EventDispatcher getSynchronousDispatc
45   return this.synchronousDispatcher;
46  }
```

```
47
48   public void setSynchronousDispatcher(EventDispa
49    this.synchronousDispatcher = dispatcher;
50   }
51
52   public void fire(Event event) {
53    dispatch(event, getSynchronousDispatcher());
54   }
55
56   public synchronized void addListener(String eve
57    List<EventListener<Event>> listeners = this.re
58    if (listeners != null) {
59     listeners.add(listener);
60    } else {
61     listeners = new CopyOnWriteArrayList<EventLis
62     listeners.add(listener);
63     this.registeredListeners.put(eventId, listene
64    }
65
66   }
67
68   public void removeListener(String eventId, Even
69    List<EventListener<Event>> listeners = this.re
70    if (listeners != null) {
71     listeners.remove(listener);
72
73    }
74   }
75
76   protected void dispatch(Event event, EventDispa
77    getNextEventNumber();
78    List<EventListener<Event>> listeners = getList
79    if (!listeners.isEmpty()) {
80
81     dispatcher.dispatch(event, listeners);
82
83    }
84   }
85
86    private static EventHub createInstance() {
87           EventHub instance = new EventHub(new Si
88           return instance;
89       }
90
91   private List<EventListener<Event>> getListeners
92    List<EventListener<Event>> listeners = this.re
93    return (listeners != null) ? listeners : Colle
94   }
95
96 }
97
98
```

Step 6: Finally, the ***EventHubMain*** that has the main method to run, and creates **3 listeners** as anonymous inner classes, and also acts as a producer to fire events. The listeners and the producer are decoupled via *EventHub* as the producer and listeners don't interact with each other, but via the ***EventHub*** and ***Event*** classes.

```
1  package com.writtentest14;
2
3  import java.util.concurrent.TimeUnit;
4
5  public class EventHubMain {
6
7   private static final String PRICE_LOAD_EVENT =
8   private static final String SECURITY_LOAD_EVENT
9
10  public static void main(String[] args) {
11
12   // Anonymous listener1
13   EventHub.instance().addListener(PRICE_LOAD_EVE
14
15    @Override
16    public void onEvent(Event event) {
17     System.out.println(PRICE_LOAD_EVENT + " rece
18     try {
19      TimeUnit.SECONDS.sleep(10);
20     } catch (InterruptedException e) {
21      // TODO Auto-generated catch block
22      e.printStackTrace();
23     }
24    }
25
26   });
27
28   // Anonymous listener2
29   EventHub.instance().addListener(SECURITY_LOAD_
30
31    @Override
32    public void onEvent(Event event) {
33     System.out.println(SECURITY_LOAD_EVENT + " r
34     try {
35      TimeUnit.SECONDS.sleep(10);
36     } catch (InterruptedException e) {
37      // TODO Auto-generated catch block
38      e.printStackTrace();
39     }
40    }
41
42   });
43
44   // Anonymous listener3
45   EventHub.instance().addListener(PRICE_LOAD_EVE
46
47    @Override
48    public void onEvent(Event event) {
49     System.out.println(PRICE_LOAD_EVENT + " rece
50     try {
51      TimeUnit.SECONDS.sleep(10);
52     } catch (InterruptedException e) {
53      // TODO Auto-generated catch block
54      e.printStackTrace();
55     }
56    }
57
58   });
59
60   // Event dispatcher
61   while (true) {
62    System.out.println("Event fired " + PRICE_LOA
63    EventHub.instance().fire(new Event(PRICE_LOAD_
64
65     try {
```

```
66      TimeUnit.SECONDS.sleep(5);
67     } catch (InterruptedException e) {
68      e.printStackTrace();
69     }
70
71     System.out.println("Event fired " + SECURITY_
72     EventHub.instance().fire(new Event(SECURITY_L
73
74    }
75   }
76
77 }
78
```

Finally, the output if you run the above class, which runs for ever in a while loop.

```
1 Event fired PL_EVENT.............
2 PL_EVENT received by listener class com.writtente
3 PL_EVENT received by listener class com.writtente
4 Event fired SL_EVENT.............
5 SL_EVENT received by listener class com.writtente
6 Event fired PL_EVENT.............
7 PL_EVENT received by listener class com.writtente
8
```

In the next part, let's integrate it with JMX.

Popular Posts

♦ 11 Spring boot interview questions & answers

825 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

765 views

18 Java scenarios based interview Questions and Answers

399 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

388 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

295 views

♦ 7 Java debugging interview questions & answers

293 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

285 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

279 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

239 views

001B: ♦ Java architecture & design concepts interview questions & answers

201 views

Bio | Latest Posts

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ 01: ♥♦ 13 Spring basics Q1 – Q7 interview questions & answers

Event Driven Programming in Java – Part 2 ›

Posted in Event Driven Programming, member-paid

Leave a Reply

Logged in as geethika. Log out?

Comment

Post Comment

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from “Blah blah” to “Wow”? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category

© Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.

↑