`HttpSessionActivationListener` 确实是一个让人比较蛋疼的 `listener`.下面详细说明.

Jetty

一开始看 Jetty 源码中的 `JDBCSessionManager#addSession()` 方法的时候, 看到了如下的代码片段:

```
protected void addSession(AbstractSession session) {
    ...

    synchronized (session) {
        session.willPassivate();
        storeSession(((JDBCSessionManager.Session) session));    // 保存到数据库中
        session.didActivate();
    }
}
```

主要就是集中在 `session.willPassivate()` 和 `session.didActivate()` 的实现上,这两个实现差不多,先看下 `session.willPassivate()`:

```
public void willPassivate() {
    synchronized (this) {
        HttpSessionEvent event = new HttpSessionEvent(this);
        for (Iterator<Object> iter = getAttributeMap().values().iterator(); iter.hasNext();
 ) {
            Object value = iter.next();
            if (value instanceof HttpSessionActivationListener) {
                HttpSessionActivationListener listener = (HttpSessionActivationListener) va
lue;
                listener.sessionWillPassivate(event);
            }
        }
    }
}
```

也就是说这里会调用 `HttpSessionActivationListener#sessionWillPassivate()` 方法.先稍微等下往下看, 先自个编写一个实现了 `HttpSessionActivationListener` 接口的类,并在 `web.xml` 中配置,有人可能会怀疑这么简单的东西,貌似没有啥必要做实验.  ,实践才是检验真理的唯一标准.另外,我们需要配置 Jetty 使用 JDBC 的方式来存储 sessions.启动服务器:

```
----------MyHttpSessionActivationListener----------
HelloFilter 实例化...
HelloFilter init()...
name: Tom
age : 24
```

说明我们的 `HttpSessionActivationListener` 被正常调用了,然后做一个请求,现在问题来了,很纳闷, `sessionWillPassivate()` 、 `sessionDidActivate()` 两个方法并没有按照我们预想的那样被调用:

```
HelloFilter doFilter() start...
HelloFilter doFilter() end...
```

是不是 Jetty 的实现有问题? 接下来在 Tomcat 中也进行了测试,即使是使用 `JDBCStore` 存储 sessions 的方式,结果和 Jetty 一致.是不是有点诡异,翻下 servlet 规范中对 `HttpSessionActivationListener` 的描述:

> 容器必须在迁移会话时通知实现了 HttpSessionActivationListener 的所有会话属性。它们必须在序列化会话

> 之前通知钝化监听器，在反序列化之后通知激活监听器。

将重点集中在 `会话属性` 这几个字上.也就是说, `HttpSessionActivationListener` 和 `HttpSessionListener` 这些 listener 是有些差异的.既然是会话属性,那么按照下面的方式来做就可以.

```
1.      session.setAttribute("sessionActivation", new MyHttpSessionActivationListener());
```

接着直接在 JSP 中编写如上的代码测试(Jetty 仍然使用 JDBC 存储 session).然后请求这个页面:

```
----------MyHttpSessionActivationListener----------
HelloFilter doFilter() end...
----------Session Will Passivate----------
----------Session Did Acrivate-----------
```

嗯,现在确实是生效了.因为 Jetty JDBC Session 采用的是每有一个新的请求,就会立即插入到数据库中.上面的 `addSession()` 方法也证实了这一点.这里先不过关心这种每有新的请求就插到数据库是否有问题.

如果不使用 JDBC,配置 `storeDir` 目录来存储 sessions,那么在服务器正常关闭的情况下, `sessionWillPassivate()` 将会被调用(仍然是调用过 `session.setAttribute("sessionActivation", new MyHttpSessionActivationListener())`),当服务器下次启动的时候, `didActivate()` 将被调用.可以看下 Jetty 序列化文件:

```
00000000  00 19 74 68 61 6E 33 66  32 74 35 76 6A 35 31 6F  ..than3f2t5vj51o
00000010  39 67 39 67 63 71 6F 38  62 73 38 00 19 74 68 61  9g9gcqo8bs8..tha
00000020  6E 33 66 32 74 35 76 6A  35 31 6F 39 67 39 67 63  n3f2t5vj51o9g9gc
00000030  71 6F 38 62 73 38 00 00  01 4C C5 3E E6 97 00 00  qo8bs8...L.>....
00000040  01 4C C5 3E E6 97 00 00  00 00 00 00 00 01 AC ED  .L.>............
00000050  00 05 77 13 00 11 73 65  73 73 69 6F 6E 41 63 74  ..w...sessionAct
00000060  69 76 61 74 69 6F 6E 73  72 00 31 6C 69 73 74 65  ivationsr.1liste
00000070  6E 65 72 73 2E 73 65 73  73 69 6F 6E 2E 4D 79 48  ners.session.MyH
00000080  74 74 70 53 65 73 73 69  6F 6E 41 63 74 69 76 61  ttpSessionActiva
00000090  74 69 6F 6E 4C 69 73 74  65 6E 65 72 39 CF 71 69  tionListener9.qi
000000a0  58 AE 28 2A 02 00 02 49  00 03 61 67 65 4C 00 04  X.(*...I..ageL..
000000b0  6E 61 6D 65 74 00 12 4C  6A 61 76 61 2F 6C 61 6E  namet..Ljava/lan
000000c0  67 2F 53 74 72 69 6E 67  3B 78 70 00 00 00 1A 74  g/String;xp....t
000000d0  00 03 54 6F 6D 00 00 07  08                       ..Tom....
```

所以上面还忽略掉了一个问题就是实现了 `HttpSessionActivationListener` 的时候同时需要实现 `Serializable` 接口,否则 Jetty 会序列化会出现问题.

Tomcat

上面已经说过了 Jetty,那么现在再来谈谈 Tomcat,也分为文件和 JDBC 的方式:

文件存储

`sessionWillPassivate()` 、 `sessionDidActivate()` 的调用方式与 Jetty 类似.

JDBC 存储

和 Jetty 不一样的是, Tomcat 不会每次有新的 session 就插入到数据库,而且在服务器正常关闭的情况下才会持久化到数据库中,那么 `sessionWillPassivate()` 方法也只会在服务器正常关闭的情况下才会被调用一次(当然也有方式设置空闲的时候持久化到数据库中).

当服务器启动的时候,只有当第一次请求的时候, 如果 session 在数据库中存在的话,那么会将其从数据库中反序列化出来.然后调用 `sessionDidActivate()` 方法.

但是 Tocmat 存在一个什么问题呢? 如果你的 `HttpSessionActivationListener` 实现没有实现

`Serializable` 的话, 那么持久化到数据库的时候不会报告任何错误信息.这可能是让人困惑的一点.

Jetty vs Tomcat

从上面看到, 对于 `HttpSessionActivationListener` 的实现, Jetty 和 Tomcat 在使用 JDBC 的方式上还是存在不少差异的.下面总结一下各自的问题:

- Jetty 每次有新的 session 就会插入数据,并发数高的时候可能会导致性能问题.另外, `sessionDidActivate()` 在持久化到数据库之后就被调用,是否有点不符合规范?
- Tomcat 对于没有实现 `Serializable` 的 `HttpSessionActivationListener`, 那么会导致 `HttpSessionActivationListener` 不生效, 这点可能会让人产生困惑.
- 如果一个 listener 就只是单单的实现 `HttpSessionActivationListener`, 那么在启动的时候确实会将其作为一个 listener, 并会加载和实例化.但是对于后面是没啥用的.因为其 `sessionWillPassivate()` 、 `sessionDidActivate()` 肯定不会被调用.
- Jetty 和 Tomcat 持久化机制不一样, Jetty 对于调用 `session.setAttribute(key, value)` 设置的参数, 在服务器正常终止后都会进行持久化,如果没有实现 `Serializable` ,那么持久化会失败.Tomcat 则不一样,如果没有实现 `Serializable` 接口,那么对这种数据就不进行持久化.
- 如果使用注解的方式配置 `HttpSessionActivationListener`, Jetty 不会将其作为一个 listener.但是使用 `web.xml` 配置就没问题,导致不一致.具体可见 `WebListenerAnnotation#apply()` 方法(未检查是否为 `HttpSessionActivationListener` 类型).关于这个问题可以查看 https://bugs.eclipse.org/bugs/show_bug.cgi?id=465074 .

最后, 虽然 `HttpSessionActivationListener` 在容器中的行为可能会导致程序的行为不一致, 但是大部分情况下可能很少使用此 listener. 如果要使用, 稍微注意下.