com.fr.stable.fun.SessionManageProcessor

-
-
-
-
-
-
-
-
-

sessioninfoID

sessioninfoSessionManageProcessorsessionsession

SessionManageProcessor.java

```
package com.fr.stable.fun;

import com.fr.stable.fun.mark.Immutable;

/**
 * Created by Administrator on 2015/11/5 0005.
 */
public interface SessionManageProcessor extends Immutable {

    String XML_TAG = "SessionManageProcessor";

    int CURRENT_LEVEL = 1;

    /**
     * Session
     *
     * @param sessionID id
     */
    void registSessionCreate(String sessionID);

    /**
     * Session
     *
     * @param sessionID id
     */
    void registSesssionDestroy(String sessionID);

    /**
     * Session
     */
    String getServerState();
}
```

SessionPoolManager.java

```
package com.fr.web.core;

...
```

```java
/**
 * SessionSession
 */
public class SessionPoolManager {

    ...

    /**
     * sessionID -> SessionIDInforsessionIDMap
     *
     * @param sessionIDInfor sessionID SessionIDInfor
     */
    public static String addSessionIDInfor(SessionProvider sessionIDInfor) throws Exception {
        if (sessionIDInfor == null) {
            FineLoggerFactory.getLogger().error("create empty sessionProvider!");
            return null;
        }

        String sessionID = SessionPoolManager.randomSessionID();
        ExtraClassManagerProvider pluginProvider = PluginModule.getAgent(PluginModule.ExtraCore);
        if (pluginProvider != null) {
            Set<SessionPrivilegeFilterProvider> set = pluginProvider.getArray(SessionPrivilegeFilterProvider.
XML_TAG);
            for (SessionPrivilegeFilterProvider provider : set) {
                sessionID = provider.encodeSessionID(sessionID);
            }
        }

        while (sessionIDMap.containsKey(sessionID)) {
            sessionID = SessionPoolManager.randomSessionID();
        }
        // sessionIDSessionIDInforsessionIDMap
        sessionIDMap.put(sessionID, sessionIDInfor);
        //
        if (ClusterStatusHelper.isClusterEnv()) {
            //
            sessionStateService.put(
                    sessionID,
                    new SessionShareInfo()
                            .startTime(sessionIDInfor.getStartTime())
                            .sessionId(sessionID)
                            .clusterNodeId(ClusterStatusHelper.getCurrentNodeId())
            );
        }

        recordSessionCreate(sessionID);
        Calculator.putThreadSavedNameSpace(SessionIDInfo.asNameSpace(sessionID));
        sessionIDInfor.setSessionID(sessionID);
        addDelayedTask(sessionID);
        return sessionID;
    }

    private static void recordSessionCreate(String sessionID) {
        ExtraClassManagerProvider pluginProvider = PluginModule.getAgent(PluginModule.ExtraCore);
        if (pluginProvider != null) {
            SessionManageProcessor sessionManageProcessor = pluginProvider.getSingle(SessionManageProcessor.
XML_TAG);
            if (sessionManageProcessor != null) {
                sessionManageProcessor.registSessionCreate(sessionID);
            }
        }
    }

    private static void recordSessionDestroy(String sessionID) {
        ExtraClassManagerProvider pluginProvider = PluginModule.getAgent(PluginModule.ExtraCore);
        if (pluginProvider != null) {
            SessionManageProcessor sessionManageProcessor = pluginProvider.getSingle(SessionManageProcessor.
XML_TAG);
            if (sessionManageProcessor != null) {
                sessionManageProcessor.registSesssionDestroy(sessionID);
            }
```

```
        }
    }



    /**
     * Session
     *
     * @param sessionID session
     */
    public static void closeSession(final String sessionID) {
        synchronized (SessionPoolManager.class) {
            boolean addDirectly = sessionToDelete.add(sessionID);
            if (!addDirectly) {
                //false
                return;
            }
        }
        //sessiontabsessionMap
        TimeoutSessionRecorder.removeSession(sessionID);
        closeSessionPool.submit(new Runnable() {
            @Override
            public void run() {
                try {
                    processCloseSession(sessionID);
                    synchronized (SessionPoolManager.class) {
                        sessionToDelete.remove(sessionID);
                    }
                } catch (Throwable e) {
                    FineLoggerFactory.getLogger().error(e.getMessage(), e);
                }
            }
        });
    }

    private static void processCloseSession(String sessionID) {
        FineLoggerFactory.getLogger().info("Session closed: {}", sessionID);
        SessionProvider sessionIDInfor = sessionIDMap.remove(sessionID);
        if (sessionIDInfor != null) {
            // session.
            AllHistoryAndLiveSession.getInstance().addHistroySession(sessionIDInfor);
            SessionHelper.remove(sessionID);
            sessionIDInfor.release();
            recordSessionDestroy(sessionID);

            WebContextProvider context = sessionIDInfor.getWebContext();
            if (context != null) {
                RemoteAddressManager.remove(context.getAddress(), sessionID);
            }
            // session
            try {
                sessionStateService.delete(sessionID);
            } catch (Exception e) {
                FineLoggerFactory.getLogger().error("Failed to delete session '" + sessionID + "' from state
service", e);
            }
        }
    }

    ...
}
```

FR	9.0		

FR	10.0		

plugin.xml

```
<extra-core>
        <SessionManageProcessor class="your class name"/>
</extra-core>
```

SessionManageProcessor processor = PluginModule.getAgent(PluginModule.ExtraCore).getSingle(SessionManageProcessor.XML_TAG);Session

SessionPoolManager

10.0getServerState

demodemo-session-manage-processor