Tapestry5HowToRunTaskInThread

!IMPORTANT because TAPESTRY-2141 is resolved in T5.0.11 there are two versions of the code

see also Tapestry5HowToWorkQueue

Questions about running tasks in separate thread were asked more than few times on the mailing list. The code is simple but some important things must be considered.

Most important thing is to call PerthreadManager.cleanup() after the thread is finished.

Other thing is that the separate thread will have different instances of any threaded services and will not have any user specific data. For example if you load an Hibernate entity inside the request thread and pass it to the task, the task will get a different Hibernate Session instance and will fail if you try to save the instance while the task is running (in a separate thread that is)

To save you some trouble I've created a simple service that runs tasks in a separate thread and makes sure PerthreadManager.cleanup()is called. You can inject it anywhere and call:

```
_threadSource.runInThread(new Runnable(){
    public void run(){
        //do something
    }
});
```

interface ThreadSource

```
public interface ThreadSource {

    /** runs the task using default exception handler, which writes exception to the log */
    public abstract void runInThread(Runnable task);

    /** runs a task in a separate thread and makes sure that tapestry ioc resources are freed after
     * thread finishes.*/
    public abstract void runInThread(final Runnable task, final TaskExceptionHandler taskExceptionHandler);

}
```

interface TaskExceptionHandler

```
public interface TaskExceptionHandler {
    public void exceptionThrown(Object task, Throwable exception);
}
```

T5.0.11

implementation for T5.0.11 and above (T5.0.11-SNASPSHOT included).You can modify it any way you like it, and even add a thread pool.

```
import org.apache.tapestry.ioc.services.PerthreadManager;
import org.slf4j.Logger;

public class ThreadSourceImpl implements ThreadSource {

    private final PerthreadManager _perthreadManager;
    private final Logger _logger;

    public ThreadSourceImpl(PerthreadManager perthreadManager, Logger logger){
        _perthreadManager = perthreadManager;
        _logger = logger;
    }

    /* (non-Javadoc)
     * @see tapestryutil.services.ThreadSource#runInThread(java.lang.Runnable)
     */
    public void runInThread(Runnable task){
        runInThread(task,defaultTaskExceptionHandler);
    }

    /* (non-Javadoc)
     * @see tapestryutil.services.ThreadSource#runInThread(java.lang.Runnable, tapestryutil.services.
TaskExceptionHandler)
     */
    public void runInThread(final Runnable task, final TaskExceptionHandler taskExceptionHandler){
        new Thread(new Runnable(){

            public void run() {
                try {
                   task.run();
                } catch (Throwable e) {
                    taskExceptionHandler.exceptionThrown(task, e);
                } finally {
                    _perthreadManager.cleanup();
                }
            }

        }).start();
    }

    /** default exception handler that writes exception to the log */
    private final TaskExceptionHandler defaultTaskExceptionHandler = new TaskExceptionHandler(){
        public void exceptionThrown(Object task, Throwable exception) {
            _logger.error("Task failed :"+task, exception);
        }
    };

}
```

before T5.0.11

NOTICE: before T5.0.11 ThreadCleanupHub is used, and for T5.0.11 and later PerthreadManager replaces it.

Please notice that this workarround fixes only the issue with ThreadCleanupHub, but other ThreadLocal dependant code might fail. The workarround is also not needed for Java 6.

The bug mentioned for java 5 happens if ThreadLocal.initialValue() happens inside another ThreadLocal.initialValue(), this is exactly the case when any other threaded service is initialized. It is because PerThreadServiceCreator uses ThreadLocal and inside initValue() generates a new instance of a service by calling a build method, and build method for HibernateSesionManager adds a listener to ThreadCleanupHub (which gets initialized and calls initialValue()) and things get messed up... I'm probably not makin much sense ... so here's the workarround

declare a dummy listener inside ThreadSourceImpl:

```
    /** dummy listener to force initializing the ThreadLocal and avoid http://bugs.sun.com/bugdatabase/view_bug.
do?bug_id=5025230*/
    private static final ThreadCleanupListener dummyListener = new ThreadCleanupListener(){public void
threadDidCleanup() {}};
```

then add this as first line in the run method of the new created thread (before task.run()) adding the dummy listener is just to force creation of ThreadCleanupHub instance

```
_cleanupHub.addThreadCleanupListener(dummyListener);
```

for unpatient, here's the full code with workarround

```
import org.apache.tapestry.ioc.services.ThreadCleanupHub;
import org.apache.tapestry.ioc.services.ThreadCleanupListener;
import org.slf4j.Logger;

public class ThreadSourceImpl implements ThreadSource {

    private final ThreadCleanupHub _cleanupHub;
    private final Logger _logger;

    public ThreadSourceImpl(ThreadCleanupHub cleanupHub, Logger logger){
        _cleanupHub = cleanupHub;
        _logger = logger;
    }

    /* (non-Javadoc)
     * @see tapestryutil.services.ThreadSource#runInThread(java.lang.Runnable)
     */
    public void runInThread(Runnable task){
        runInThread(task,defaultTaskExceptionHandler);
    }

    /* (non-Javadoc)
     * @see tapestryutil.services.ThreadSource#runInThread(java.lang.Runnable, tapestryutil.services.
TaskExceptionHandler)
     */
    public void runInThread(final Runnable task, final TaskExceptionHandler taskExceptionHandler){
        new Thread(new Runnable(){

            public void run() {
                _cleanupHub.addThreadCleanupListener(dummyListener);
                try {
                   task.run();
                } catch (Throwable e) {
                    taskExceptionHandler.exceptionThrown(task, e);
                } finally {
                    _cleanupHub.cleanup();
                }
            }

        }).start();
    }

    /** dummy listener to force initializing the ThreadLocal and avoid http://bugs.sun.com/bugdatabase/view_bug.
do?bug_id=5025230*/
    private static final ThreadCleanupListener dummyListener = new ThreadCleanupListener(){public void
threadDidCleanup() {}};

    /** default exception handler that writes exception to the log */
    private final TaskExceptionHandler defaultTaskExceptionHandler = new TaskExceptionHandler(){
        public void exceptionThrown(Object task, Throwable exception) {
            _logger.error("Task failed :"+task, exception);
        }
    };

}
```