Runnable Example

```
public static void testBackground() {
        Xots.getService().submit(new UserOutput());
}

@Tasklet(session = Tasklet.Session.CLONE, context = Tasklet.Context.XSPSCOPED)
private static class UserOutput extends AbstractXotsXspRunnable {

        public UserOutput() {

    }

    public void run() {
            try {
                if (getContext().getSessionScope().containsKey("javaXotsOutput")) {
                    getContext().getSessionScope().put("javaXotsOutput", null);
                }
                StringBuilder sb = new StringBuilder();
                Database currDb = Factory.getSession(SessionType.CURRENT).getCurrentDatabase();
                View states = currDb.getView("AllStates");
                View people = currDb.getView("AllContactsByState");
                for (Document state : states.getAllDocuments()) {
                        String name = state.getItemValueString("Name");
                    String key = state.getItemValueString("Key");
                    sb.append("Processing " + name + "....<br/>");
                    StringBuilder names = new StringBuilder();
                    for (Document doc : people.getAllDocumentsByKey(key, true)) {
                            String personName = doc.getItemValueString("FirstName") + " "
                                + doc.getItemValueString("LastName");
                        names.append(personName);
                        names.append(", ");
                    }
                    if (names.length() > 2) {
                            sb.append(names.substring(0, names.length() - 2) + "<br/>");
                    } else {
                            sb.append("No names found.<br/>");
                    }
                }
                getContext().getSessionScope().put("javaXotsOutput", sb.toString());
            } catch (Throwable t) {
                    XotsUtil.handleException(t, getContext());
                getContext().getSessionScope().put("javaXotsOutput", "ERROR");
            }
    }
}
```

Lines 1-3 are a utility method to trigger the Xots task. It creates a new instance of the `UserOutput` class and passes it to the Xots service.

Lines 6 onwards are the `UserOutput` class. Note that it extends the `AbstractXotsXspRunnable` class.

Line 5 contains annotations that determine two aspects that Xots applies to the object. The first tells it to clone the current user session, so `SessionType.CURRENT` will be the same as `SessionType.CURRENT` elsewhere. `SessionType.NATIVE` will still be available if required. The second tells it the scope is XSPSCOPED, which means all the XPages-related objects - scope maps, FacesContext and XSPContext - will be made available to the tasklet.

Lines 8 - 10 are the constructor for the class. In this scenario, the scopes have everything required. But if there was additional information required - for example a UNID of a document to run against - that could be passed into the constructor. Obviously Domino objects cannot be passed into the constructor of a Xots tasklet. So you would need to pass distinguishing information and retrieve the Domino object during the `run` method.

Lines 12 onwards are the `run` method. A class that implements Runnable will need to have this method, it's the one that is triggered when Xots runs the tasklet. Because this is a Runnable - a background task that is just kicked off and allowed to complete - it doesn't return anything.

Lines 14 - 16 show how to get a handle on scoped variables and other objects passed into the Xots context object. Here the code checks whether the sessionScope variable already exists and if so clears it.

Lines 17 - 20 get the database as the current user and two views, one of states and one of contacts by state.

Lines 21 onwards iterate each state in the view of states. A message is added to the StringBuilder giving the name of the state being processed.

Lines 26 - 31 iterate all contacts for that state, creating a comma-separated string of names for all contacts in that state.

Lines 32 - 36 add the names (or a message saying there were no names) and a carriage return.

Line 38 posts the result to the scoped variable. Next time the user accessing or refreshes a page that uses that variable, the result can be displayed. Alternatively there couold be JavaScript on the page to preiodically refresh until the scoped variable is not null.

Lines 39 - 41 contain the error handling. This shows how to output the error to OpenLog using `XotsUtil`. By passing the context, the logging will be able to output the current database and the Xots class the error is triggered from.