Configuration changes required in H2O applications

Added by Ravichandran Nagarajan, last edited by BALAKRISHNAN R. on Oct 15, 2014

Overview

To enable the collapsing of backend on frontend as part of the 12c weblogic migration, there are quite a few changes required in configurations of application deployed in H2O cluster. We will see those changes, their implications in H2O application in this section.

Changes required in CommonProperties

Due to the collapsing of the H2O cluster, the different CommonProperties maintained by application on front-end and back-end requires merging. There are two points that affect the property files and eventually leading to changes that needs to be performed in the properties file / code.

1. Property keys duplicated between frontend and backend with different values will have problems while merging.
2. Moving to 12c, our packaging becomes more industry standard, which leads them to be placed under META-INF of respective jar. Due to this we want to reduce the complexity of maintaining separate property file for TEST, PP and PRO merging them altogether.

For scenario 1, simply the properties are renamed in frontend with a different key and required implementation is changed accordingly if needed.
For scenario 2, the properties are managed with prefix for environment. So, key1 will become key1.TEST, key1.PRE, key1.TEST.PRE, key1.PRE.TEST and key1.PRO to represent different values for environment.

Changes required in DataSource

Due to the collapsing of FrontEnd and BackEnd, the Default DataSource usage in front end needs to be addressed.

Two entries of DataSource.properties has been introduced for FEDomain and BEDomain as

- ***clientPool*** that points to ***weblogic.jdbc.clientPool*** representing ***FE*** clientPool
- ***clientXAPool*** that points to ***weblogic.jdbc.jts.clientPool*** representing ***BE***

Projects using the getDataSource() default method in front end, are requested to perform code changes for new collapsed 12c targeted environment.

Default DataSource usage in front end, have two scenario:

1. If Project code contains getDataSource() that is being used **only** in FrontEnd
    - **Solution:** Use ***clientPool*** mentioned in the DataSource entry above.
2. If Project code contains getDataSource() that is being used **in both** FrontEnd and BackEnd
    - **Solution:** Use always **clientXAPool** and perform a **regression test** to verify that using a distributed datasource does not give problems to FE executions.

Attention: The default datasource in FrontEnd is **clientPool** whereas in BackEnd it is **clientXAPool**.