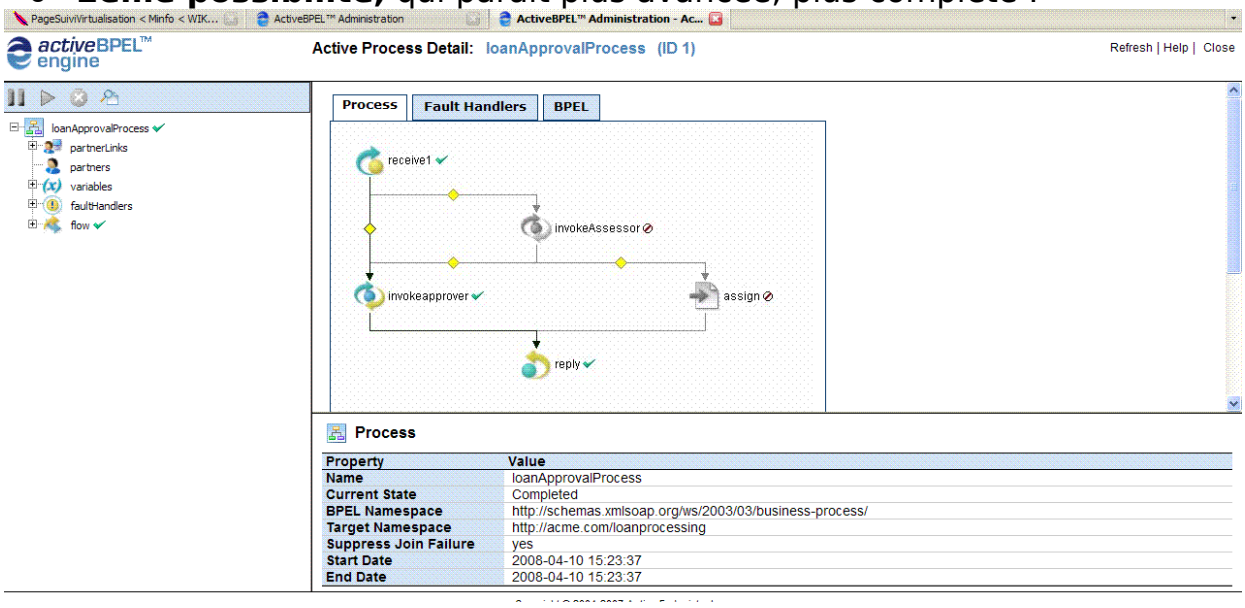
Compte-Rendu : Monitoring dans ActiveBPEL

Dans le but de surveiller le mieux possible une orchestration BPEL, nous avons étudié les possibilités offertes par ActiveBPEL.

- **1ère possibilité :**
  ActiveBPEL fournit la surveillance du moteur, et si vous le souhaitez, vous pouvez déclencher une alerte pour prévenir l'administrateur lorsque le moteur cesse de fonctionner ou si une propriété qui est surveillée provoque une erreur ou un avertissement.
  Vous pouvez concevoir un processus BPEL afin de faire un service d'alerte de surveillance. Le processus doit être basé sur un fichier WSDL nommé monitorAlert.wsdl. Ce fichier est situé dans le dossier d'ActiveBPEL Designer installation/support.
  Ce fichier WSDL définit les opérations "handleAlert", qui reçoivent les informations du serveur en ce qui concerne :
  - l'état du moteur,
  - les propriétés surveillées du moteur,
  - leur niveau de contrôle
  - et la valeur seuil.

- **2ème possibilité,** qui parait plus avancée, plus complète :

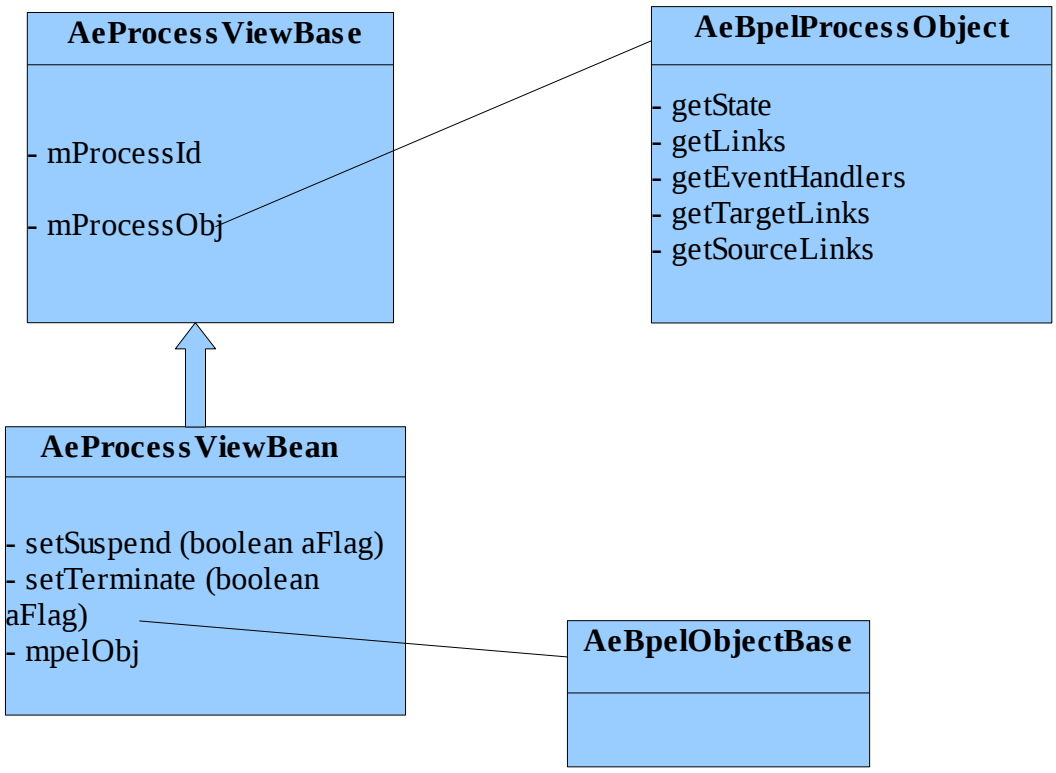


  Nous avons remarqué que l'interface d'administration de ActiveBPEL permet l'affichage de l'état d'une orchestration, ainsi que de chaque exécution (et chaque activité). Nous avons donc étudié le source de la page et avons remarqué qu'elle utilisait des JSP.
  Nous avons finalement trouvé des classes très intéressantes qui pourraient nous permettre de suivre l'état des exécutions dans le package **org.activebpel.rt.bpeladmin.war.web.processview :**
  - **AeProcessViewBase** : cette classe possède le détail d'un processus, comme son Id (mProcessId), et ses activités : mProcessObj (qui est un AeBpelProcessObject) est en fait la racine, on verra par la suite comment se déplacer dans l'arbre.
  - **AeProcessViewBean** qui hérite de AeProcessViewBase et qui

permet de faire des setSuspend, setTerminate ... pour modifier l'état de l'activité. De plus il possède mBpelObj qui est un AeBpelObjectBase (à approfondir...)

- **AeBpelProcessObject** : contient getState(), getLinks(), getEventHandlers(), qui peuvent nous informer sur l'état de l'activité et getTargetLinks(), getSourcesLinks() qui permettent de nous déplacer dans l'arbre des activités d'une exécution.

Voici le diagramme de classe correspondant :



A partie de la classe **AeProcessViewBean,** nous pourrions donc surveiller et contrôler l'orchestration.