Persistence and Transactions

Agenda

01 RUNTIME STATE

02 PROCESS DEFINITIONS

03 HISTORY LOG

04 TRANSACTIONS

05 QUIZ

06 CONCLUSION

07 HANDS-ON

Runtime State

Runtime State

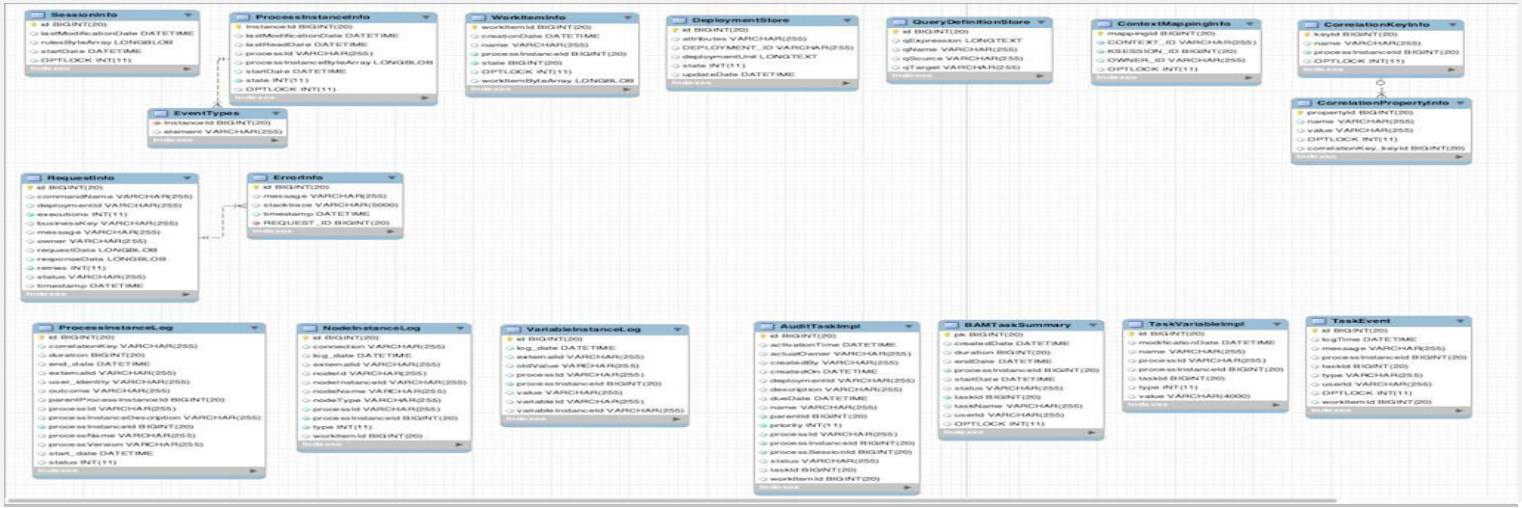
- jBPM allows the persistent storage of certain information.
- Storing the process runtime state is necessary in order to be able to continue execution of a process instance at any point, if something goes wrong.
- Also, the process definitions themselves, and the history information (logs of current and previous process states) can also be persisted.
- Whenever a process is started, a process instance is created, which represents the execution of the process in that specific context.
- For example, when executing a process that specifies how to process a sales order, one process instance is created for each sales request. The process instance represents the current execution state in that specific context, and contains all the information related to that process instance.
- The runtime state of an executing process can be made persistent, for example, in a database.
- This allows to restore the state of execution of all running processes in case of unexpected failure, or to temporarily remove running instances from memory and restore them at some later time.
- jBPM allows you to plug in different persistence strategies. By default, if you do not configure the jBPM engine otherwise, process instances are not made persistent.

Runtime State

- If you configure the engine to use persistence, it will automatically store the runtime state into the database.
- You do not have to trigger persistence yourself, the engine will take care of this when persistence is enabled.
- Whenever you invoke the engine, it will make sure that any changes are stored at the end of that invocation, at so-called safe points.
- Whenever something goes wrong and you restore the engine from the database, you also should not reload the process instances and trigger them manually to resume execution, as process instances will automatically resume execution if they are triggered, like for example by a timer expiring, the completion of a task that was requested by that process instance, or a signal being sent to the process instance.
- The engine will automatically reload process instances on demand.
- Binary Persistence (Marshalling)
- jBPM uses a binary persistence mechanism, otherwise known as marshalling, which converts the state of the process instance into a binary dataset. When you use persistence with jBPM, this mechanism is used to save or retrieve the process instance state from the database. The same mechanism is also applied to the session state and any work item states.

Runtime State

- When the process instance state is persisted, two things happen:
  - First, the process instance information is transformed into a binary blob. For performance reasons, a custom serialization mechanism is used and not normal Java serialization.
  - This blob is then stored, alongside other metadata about this process instance. This metadata includes, among other things, the process instance id, process id, and the process start date.



Process Definitions

History Log

History/Audit Log

- In many cases it will be useful (if not necessary) to store information *about* the execution of process instances, so that this information can be used afterwards. For example, sometimes we want to verify which actions have been executed for a particular process instance, or in general, we want to be able to monitor and analyze the efficiency of a particular process.
- However, storing history information in the runtime database can result in the database rapidly increasing in size, not to mention the fact that monitoring and analysis queries might influence the performance of your runtime engine. This is why process execution history information can be stored separately.
- This history log of execution information is created based on events that the jBPM engine generates during execution. This is possible because the jBPM runtime engine provides a generic mechanism to listen to events. The necessary information can easily be extracted from these events and then persisted to a database. Filters can also be used to limit the scope of the logged information.

Transactions

Transactions

- The jBPM engine supports JTA transactions. It also supports local transactions *only* when using Spring. It does not support pure local transactions at the moment. For more information about using Spring to set up persistence, please see the Spring chapter in the Drools integration guide.
- Whenever you do not provide transaction boundaries inside your application, the engine will automatically execute each method invocation on the engine in a separate transaction. If this behavior is acceptable, you don't need to do anything else. You can, however, also specify the transaction boundaries yourself. This allows you, for example, to combine multiple commands into one transaction.
- You need to register a transaction manager at the environment before using user-defined transactions. The following sample code uses the Narayana JTA transaction manager. Use the Java Transaction API (JTA) to specify transaction boundaries

Quiz

Persistence and Transaction

Quiz

The state of a process instance is stored at so-called

a) restore points

b) data store

c) safe points

d) backup points

Answers

The state of a process instance is stored at so-called

a) restore points

b) data store

c) safe points

d) backup points

Quiz

With Persistence you cannot restore process state in runtime.

a) True

b) False

Answers

With Persistence you cannot restore process state in runtime.

a) True

b) False

Quiz

The jBPM engine supports _____ transactions

a) JTA

b) Java

c) XML

d) Database

Answers

The jBPM engine supports _____ transactions

a) JTA

b) Java

c) XML

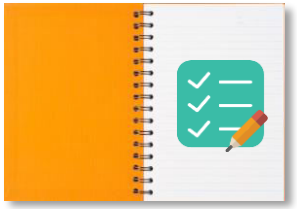
d) Database

Conclusion

Summary

In this session, you have learnt:

- Runtime State
- Process Definitions
- History Log
- Transactions



Hands-On

Practical

Hands-On 1

- Create Leave Approval process with Persistence enabled
- Create various User Task Steps and check if data get stored into tables

India : +91-7847955955

US : 1-800-216-8930 (TOLL FREE)

support@intellipaat.com

24X7 Chat with our Course Advisor