Petals Service Platform

User Manual
Draft

Adrien LOUIS
08 / 16 / 2005

Summary

1 Petals configuration

The Petals container core configuration

The Petals container has to be parameterized in a petals-config.xml file.
This file defines the global container description, such as its name, host and the administration port:

```
<container  host="ebm-04"
            identification="container"
            admin-port="9999"
/>
```

The Petals container is build upon multiple elements:

- the Petals core component, which represents the master piece of the Petals container,
- system services components, which are plugged into the Petals container:
    - the Directory component,
    - the Transporter element,
    - the ItineraryResolver,
    - the AddressResolver.

Those system services are implemented by objects whose class name are defined in the petals-config.xml file :

```
<system-services>

    <directory>
        <class-name>DirectoryImpl</class-name>
        <parameters param1="" param2=""/>
    </directory>

    <transporter>
        <class-name>TransporterImpl</class-name>
        <parameters param1="" param2=""/>
    </transporter>

    <itinerary-resolver>
        <class-name>ItineraryResolverImpl</class-name>
        <parameters param1="" param2=""/>
    </itinerary-resolver>

    <address-resolver>
        <class-name>AddressResolverImpl</class-name>
        <parameters param1="" param2=""/>
    </address-resolver>

</system-services>
```

Then, all JBI components that are hosted by the Petals container have to be defined in the petals-config.xml file:

```
<components>

    <component type="engine" identification="HelloServiceComponent">
        <component-class-name>hello.HelloService</component-class-name>
        <component-class-path>hello.jar</component-class-path>
        <bootstrap-class-name>hello.Bootstrap</bootstrap-class-name>
        <bootstrap-class-path>hello.jar</bootstrap-class-path>
    </component>

    <component type="binding" identification="ClientComponent">
        <component-class-name>client.Client</component-class-name>
        <component-class-path>client.jar</component-class-path>
        <bootstrap-class-name></bootstrap-class-name>
        <bootstrap-class-path></bootstrap-class-path>
    </component>

</components>
```

The Petals system services configuration

Petals is delivered with already defined system services.
The Directory component and the Transporter components are built upon JORAM-based objects. JORAM is an open source MOM hosted by the ObjectWeb consortium.

JORAM has to be configured for allowing the Petals container to work.

JORAM is based on agents, specially a JMS agent and a JNDI agent. Both agents have to be configured in an A3servers.xml.
For enabling a distributed environment, this file has to define ALL the JORAM agents that are present on the network: the container's ones and all the others container's ones that might be present on the network.

For the distributed JNDI directory, a master agent has to be defined, and the other JNDI agents have to reference the agent id in their configuration.

For example, for a Petals container hosted on Zeus, in a network containing two others Petals containers, hosted on Poseidon and Athena, the a3servers.xml file looks like this:

```
<config>
  <domain name="EBM"/>

  <property name="Transaction" value="fr.dyade.aaa.util.NullTransaction"/>

   <server id="0" name="joram0" hostname="zeus">
    <network domain="EBM" port="16200"/>
    <service class="org.objectweb.joram.mom.proxies.ConnectionManager"
          args="root root"/>
    <service class="org.objectweb.joram.mom.proxies.tcp.TcpProxyService"
          args="16300"/>
     <service class="fr.dyade.aaa.jndi2.distributed.DistributedJndiServer" args="16400"/>
  </server>

<server id="1" name="joram1" hostname="poseidon">
    <network domain="EBM" port="16201"/>
    <service class="org.objectweb.joram.mom.proxies.ConnectionManager"
          args="root root"/>
    <service class="org.objectweb.joram.mom.proxies.tcp.TcpProxyService"
          args="16301"/>
     <service class="fr.dyade.aaa.jndi2.distributed.DistributedJndiServer" args="16401 0"/>
  </server>

  <server id="2" name="joram2" hostname="athena">
    <network domain="EBM" port="16202"/>
    <service class="org.objectweb.joram.mom.proxies.ConnectionManager"
          args="root root"/>
    <service class="org.objectweb.joram.mom.proxies.tcp.TcpProxyService"
          args="16302"/>
     <service class="fr.dyade.aaa.jndi2.distributed.DistributedJndiServer" args="16402 0"/>
  </server>

…
</server>
```

For the internal work of the Directory component, a jndi.properties file has to be set:

```
java.naming.factory.host localhost
java.naming.factory.port 16400
java.naming.factory.initial fr.dyade.aaa.jndi2.client.NamingContextFactory
```

As the a3servers.xml file contains all present agents, a supplementary file has to be set, in order to allow the system components to know which JORAM agent is the local one. A joram.properties file looks like this:

```
id = 0
host = zeus
tcp = 16300
user = root
pwd = root
```

Note: Depending on JORAM internal work, the Petals container, whose JNDI agent is the master of the whole distributed JORAM configuration, has to be started first.

2 JBI Components

Create a JBI Component

????

Adding a JBI Component to a Petals container

????

Communication between a JBI Component and a Petals container

????