Added by Alex Boisvert, last edited by Alex Boisvert on Nov 25, 2008

Deployment Service

Assembly Model

An assembly is a composition of one or more components for the purpose of deployment. An assembly may be a complete application, or part of an application.

Components are basic units of software construction. A component may be a web form, or a business process (BPEL), an XPath function, a web-service, etc.

```
+-----------------+
| Assembly        |
+-----------------+
| Component1      |
| Component2      |
| Component3      |
| ...             |
+-----------------+
```

Deployment

Deploying an assembly implies the deployment of all its components, and similarly for undeployment. Assemblies are managed through the Deployment Service that is responsible for the deployment of components into their respective containers.

Components within an assembly tend to be closely-coupled, which is the reason they are bundled and deployed together to follow the same lifecycle. Distinct assemblies, on the other hand, may be related but are generally more loosely-coupled and do not share the same lifecycle. In other words, they may be deployed, upgraded and undeployed independently, where applicable.

At this time, the assembly model does not attempt to model and enforce static or dynamic runtime relations between assemblies or components other than those of deployment composition between an assembly and its components.

```
+----------+    deployed     +--------------------+      deploys             +------------+
| Assembly |  ------------>  | Deployment Service |  ---------------> (many)  | Container  |
+----------+                 +--------------------+    components            +------------+
```

Assembly Structure

An assembly can be either an archive file (.zip), or an "exploded" directory. The assembly has a set of directories, with each directory corresponding to a component. As a convention, the deployment service uses top-level directory extensions to determine the component type.

Here is an example assembly composed of a single component,

```
MyAssembly\
  MyComponent.someType\
     someFile.ext
```

the same assembly in .zip form would have the following structure:

```
MyAssembly.zip
  MyComponent.someType\
     someFile.ext
```

A more realistic assembly might look like,

```
CustomerSatisfaction.zip                        (this is the assembly)
  OrderSatisfaction.process\                    (1st component of type "process")
    OrderSatisfaction.bpel
    OrderSatisfaction.wsdl
    OrderSatisfaction.xsd
    deploy.xml
  CustomerSatisfaction.webreport\               (2nd component of type "webreport"
    CustomerSatisfactionDashboard.webreport
  CustomerSatisfaction.jdbc\                    (3rd component of type "jdbc")
    OrderSatisfaction.sql
    OrderSatisfaction.properties
  CollectOrderSatisfaction.pipa\                (4th component of type "pipa")
    OrderSatisfaction.pipa
  CollectOrderSatisfaction.xform\               (5th component of type "xform")
    CollectOrderSatisfaction.xsd
    CollectOrderSatisfaction.xform
```

Versioning Model

Each deployed assembly is assigned a version number upon deployment as well as other properties such as the deployment time and who deployed the assembly. This metadata helps identify and distinguish between deployed assembly instances.

Only one version of a given assembly may be active at once. The components of the active assembly are used by default whenever new requests come in. For example, if a new version of an assembly is deployed, older processes are retired and the new processes become active.

If it is desired to have two coexisting variants of the same assembly, each must have its own name and developers are responsible to ensure there are no conflicts between them; the deployment service cannot prevent these conflicts as it is not aware of the inner workings of components. Containers may prevent or warn against such conflicts at deployment or run time.

Example

The deployment service uses naming conventions to provide automatic versioning of assemblies. Deploying an assembly with the same name as a previously deployed assembly results in the undeployment of the previous version, and incrementing the version number of the new assembly.

Let's illustrate this with an example,

1) Client deploys an assembly named A with components 1, and 2.

Result: Assembly A deployed, components 1 and 2 are activated.

2) Client deploys an assembly named A, with and updated component named 1 (but not component 2).

Result: Previous assembly A is deactivated, meaning that both of its components are deactivated. The new assembly is automatically renamed to A-2 and deployed, and the new component 1 is activated.

3) Client deploys an assembly named B with components 1 and 2.

Result: Assembly B is deployed and its components are activated. The new components are activated side-by-side with previously deployed components from assembly A-2. Components from assembly A remains deactivated.

Deployment API and Code References

The Deployment Service API presents the Java interfaces and classes of the deployment service.

Containers must implement the service-provider interface (SPI).

The Deployment Service implementation can be browsed for debugging and better understanding.

The Tempo WDS module contains an example ComponentManager implementation.

Comments