CXF Configuration - JAX-RS (RESTful Services) and JAX-WS (Web Services)

TomEE relies on Apache CXF for JAX-RS (RESTful Services) and JAX-WS (Web Services). It does not provide all CXF modules, but the most common ones for both specifications (JAX-RS is part of all distributions but JAX-WS is only part of plus one).

Configuration

CXF API is reusable but you can also configure the interceptors through `openejb-jar.xml` (located in WEB-INF).

If you want to configure JAX-RS you will use the prefix `cxf.jaxrs` and if you configure JAX-WS you use `cxf.jaxws` prefix.

TIP	to configure directly the bus use `org.apache.openejb.cxf.bus.` prefix and configure it in `conf/system.properties`.

To configure JAX-RS you need to add in `openejb-jar.xml` a `pojo-deployment`:

```
<?xml version="1.0" encoding="UTF-8"?>
<openejb-jar>
 <pojo-deployment class-name="jaxrs-application">
   <properties>
     # here will go the config
   </properties>
 </pojo-deployment>
</openejb-jar>
```

For JAX-WS you will use a `pojo-deployment` matching the webservice class name for POJO webservices or an `ejb-deployment` instead of a `pojo-deployment` for EJB webservices:

```
<?xml version="1.0" encoding="UTF-8"?>
<openejb-jar>
 <ejb-deployment ejb-name="MyEJBWebService">
   <properties>
     # here will go the config
   </properties>
 </ejb-deployment>
</openejb-jar>
```

Then once you selected your prefix and know where to write the config just use the following entries:

- properties: server factory properties
- features: CXF features
- in-interceptors: CXF in interceptors

- out-interceptors: CXF out interceptors
- in-fault-interceptors: CXF in interceptors for fault handling
- out-fault-interceptors: CXF out interceptors for fault handling
- databinding: server databinding
- providers (only for JAX-RS endpoint): list of JAX-RS providers
- skip-provider-scanning (only for JAX-RS): is provider scanning on or not (default true)

For features and interceptors the rule is the same: value is a list comma separated. Each value of the list is either a qualified class name or an id of a service in resources.xml.

Databinding is simply either a qualified name or a service id in resources.xml (located in WEB-INF).

Sample for JAX-WS

To configure WSS4J on the EJB `CalculatorBean` for instance add in openejb-jar.xml:

```
<openejb-jar xmlns="http://www.openejb.org/openejb-jar/1.1">
  <ejb-deployment ejb-name="CalculatorBean">
    <properties>
      cxf.jaxws.in-interceptors = wss4j
    </properties>
  </ejb-deployment>
</openejb-jar>
```

With associated resources.xml which will define precisely the `wss4j` configuration:

```
<resources>
  <Service id="wss4j" class-name=
"org.apache.openejb.server.cxf.config.WSS4JInInterceptorFactory" factory-name="create
">
    action = UsernameToken
    passwordType = PasswordText
    passwordCallbackClass = org.superbiz.ws.security.PasswordCallbackHandler
  </Service>
</resources>
```

Sample for JAX-RS

JAX-RS JSON page shows a sample dedicated to JAX-RS.