JNDI

TomEE has several JNDI client intended for multiple usages.

Default one

In a standalone instance you generally don't need (or want) to specify anything to do a lookup. Doing so you will inherit from the contextual environment:

```
final Context ctx = new InitialContext();
ctx.lookup("java:....");
```

LocalInitialContextFactory

This is the legacy context factory used by OpenEJB. It is still useful to fallback on the "default" one in embedded mode where sometimes classloaders or libraries can mess up the automatic conextual context.

Usage:

```
Properties properties = new Properties();
properties.setProperty(Context.INITIAL_CONTEXT_FACTORY,
"org.apache.openejb.core.LocalInitialContextFactory");
final Context ctx = new InitialContext(properties);
ctx.lookup("java:....");
```

This context factory supports few more options when **you boot the container** creating a context:

Name	Description
openejb.embedded.remotable	true/false: starts embedded services
Context.SECURITY_PRINCIPAL/Context.SECURITY_CREDENTIALS	the **global** security identity for the whole container

IMPORTANT `Context.SECURITY_*` shouldn't be used for runtime lookups with `LocalInitialContextFactory`, it would leak a security identity and make the runtime no more thread safe. This factory was deprecated starting with 7.0.2 in favor of `org.apache.openejb.core.OpenEJBInitialContextFactory`.

OpenEJBInitialContextFactory

This factory allows you to access local EJB and container resources.

```
Properties properties = new Properties();
properties.setProperty(Context.INITIAL_CONTEXT_FACTORY,
"org.apache.openejb.core.OpenEJBInitialContextFactory");
final Context ctx = new InitialContext(properties);
ctx.lookup("java:....");
```

RemoteInitialContextFactory

Intended to be used to contact a remote server, the `org.apache.openejb.client.RemoteInitialContextFactory` relies on the provider url to contact a tomee instance:

```
Properties p = new Properties();
p.put(Context.INITIAL_CONTEXT_FACTORY,
"org.apache.openejb.client.RemoteInitialContextFactory");
p.put(Context.PROVIDER_URL,
"failover:ejbd://192.168.1.20:4201,ejbd://192.168.1.30:4201");

final InitialContext remoteContext = new InitialContext(p);
ctx.lookup("java:....");
```

Contrarly to local one, the remote factory supports `Context.SECURITY_*` options in a thread safe manner and you can do lookups at runtime using them.

See Cluster page for more details on the options.

Security

The context configuration can take additional configuration to handle EJB security:

```
p.put("openejb.authentication.realmName", "my-realm"); // optional
p.put(Context.SECURITY_PRINCIPAL, "alfred");
p.put(Context.SECURITY_CREDENTIALS, "bat");
```

The realm will be used by JAAS to get the right LoginModules and principal/credentials to do the actual authentication.

HTTP case

Often HTTP layer is secured and in this case you need to authenticate before the EJBd (remote EJB TomEE protocol) layer. Thanks to TomEE/Tomcat integration login there will propagate to the EJBd context.

This can be done passing the token you need to set as `Authorization` header in the `PROVIDER_URL`:

```
// tomee/openejb principal/credentials
p.put(Context.PROVIDER_URL,
"http://localhost:8080/tomee/ejb?authorization=Basic%20dG9tZWU6b3BlbmVqYg==");
```

The token passed as `authorization` query parameter is the header value URL encoded. It can be any token like a basic one, a custom one, an OAuth2 one (in this case you need to renew it programmatically and change your client instance when renewing) etc...

TIP	basic being very common there is a shortcut with two alternate query parameter replacing `authorization` one: `basic.password` and `basic.username`.

Finally if you don't use `Authorization` header you can change the used header setting `authorizationHeader` query parameter.

NOTE	`authorization`, `authorizationHeader`, `basic.username`, and `basic.password` are removed from the URL before opening the connection and therefore not logged in the remote server access log since version 7.0.3.