Resources

In TomEE resources are mainly "singleton" (understood as defined once per server or application). Technically it can be anything but you will probably meet more Datasources than other type of resources.

Most resources will be created automatically if there is no matching resources - by name and type - when an injection will be found. To avoid that use `openejb.offline` property and set it to `true`. See Server Configuration for more detail.

Definition a resource: how does it work?

Before all let see how properties syntax is equivalent to XML one (system.properties and tomee.xml typically).

Properties syntax uses dot notation to represent setters/properties which are plain properties in XML syntax and a URL syntax with query parameters to define the resource where it is directly the resource and tag attributes in XML. Finally the id is an attribute in XML and the key of the resource definition in properties.

Let see it with a sample, both delcarations are the same:

```
myDataSource = new://Resource?type=DataSource
myDataSource.JdbcUrl = jdbc:hsqldb:mem:site
myDataSource.UserName = sa
```

```
<Resource id="myDataSource" type="DataSource">
  JdbcUrl = jdbc:hsqldb:mem:site
  UserName = sa
</Resource>
```

One started you can get injected any resource using `@Resource`:

```
@Resource(name = "myDataSource")
private DataSource dataSource;
```

Factory syntax

Here are the attributes of a resource:

Name	Optional	Description
id	false	name of the resource, will match `openejb:Resource/id` in JNDI tree.

Name	Optional	Description
provider	true	define a default resource definition using service-jar.xml
class-name	true	specify which class to instantiate
factory-name	true	specify which method to invoke on the class-name when specified to create the resource
properties-provider	true	a class responsible to provide to tomee the properties to use, it can have a property `serviceId` to know which resource it is.
classpath	true	a classpath to use to create the resource. Note: if not implementing an interface the resource will be isolated from the applications.
aliases	true	other names for the resource, allows for instance to share the same pool for a datasource used with multiple names in applications.
post-construct/pre-destroy	true	methods called when creating/destroying the resources.
Lazy	true	for resources set them to be created when first accessed and not when deployed

TomEE supports some implicit properties for resources but sometimes you just want to fully control the resource and not use implicit properties which can be affected to a property which doesn't expect such a value (typically the case if you create a custom Oracle datasource). For such case you can set `SkipImplicitAttributes` property to `true` and your resource will ignore implicit properties.

Implicit properties are:

Name	Description
transactionManager	The JTA transaction manager
ServiceId	the "id" of the resource (its name)

In the same spirit you can skip properties fallback using `SkipPropertiesFallback` and setting it to `true`. It typically avoids to fallback all unset properties (no matching property found) to a `Properties` instance and set it if one matching property is found. In Oracle case for instance it matches the connection properties which can have side effects.

Value ciphering

The propertie values support ciphering using the syntax `cipher:{algorithm}:{cipheredValue}`, for instance `cipher:Static3DES:xMH5uM1V9vQzVUv5LG7YLA==` will be read as `Passw0rd`. Ciphers can be computed using `tomee.sh` script: `${tomee.home}/bin/tomee.sh cipher Passw0rd`.

Common Resources

DataSources

DataSources have defaults for all values and a default datasource can be provided automatically but if you want to configure it here are the common properties:

You can set the boolean `JtaManaged` to false if you don't want your datasource to be using JTA - if you manage transactions yourself.

Then other configurations are linked the pool the datasource is using. By default TomEE uses tomcat-jdbc but we also provide commons-dbcp (2 for TomEE 7.x and 1 for TomEE 1.x). The properties are then the related configurations with these particular entries we try to keep in sync for both:

Name	Description
JdbcDriver	the jdbc driver of the datasource
JdbcUrl	the jdbc url of the datasource
Username	the user to use
Password	the password of the user

Password and ciphering

DataSource were the first resource to support password ciphering. Originally it was another property which is still supported. It is called `PasswordCipher`. Its value is the ciphering algorithm and it affects the password value. However `cipher:xxx` is still supported on `Password` value. Default `PasswordCipher` being `PlainText` it behaves as no ciphering is in place by default.

Sample:

```
ds = new://Resource?type=javax.sql.DataSource
# our password is "Passw0rd"
ds.Password = xMH5uM1V9vQzVUv5LG7YLA==
ds.PasswordCipher = Static3DES
```

Advanced DataSource configuration

TomEE also provides few utilities you can add in DataSource properties:

Name	Description
LogSql	Should SQL be logged (using TomEE logger)
LogSqlPackages	if set the logging will show the matching packages (separated by comma) inline when logging the query, allows to know where a query comes from
Flushable	if true the datasource can be casted as a Flushable to recreate the pool
ResetOnError	if a `SQLException` happens the pool is automatically recreated. Configuration is either "true" to do it each time an exception occurs, `x` or `retry(x)` to do it and retry until maximum `x` times
ResetOnErrorMethods	which methods are handled by ResetOnError
TomEEProxyHandler	Custom `InvocationHandler` wrapping the datasource calls
DataSourceCreator	which pool to use, `dbcp`, `tomcat`, `dbcp-alternative` (DBCP and TomEE proxying instead of DBCP JTA integration), `simple` (no pooling)

DataSource and JTA

`JtaManaged` determines wether or not this data source should be JTA managed or user managed. If set to 'true' it will automatically be enrolled in any ongoing transactions. Calling begin/commit/rollback or setAutoCommit on the datasource or connection will not be allowed. If you need to perform these functions yourself, set `JtaManaged` to `false`

DataSource and JPA

In terms of JPA persistence.xml:

- `JtaManaged=true` can be used as a 'jta-data-source'
- `JtaManaged=false` can be used as a 'non-jta-data-source'

ActiveMQResourceAdapter

Declarable in tomee.xml via

```
<Resource id="Foo" type="ActiveMQResourceAdapter">
    BrokerXmlConfig = broker:(tcp://localhost:61616)?useJmx=false
    ServerUrl = vm://localhost?waitForStart=20000&async=true
    DataSource = Default Unmanaged JDBC Database
    StartupTimeout = 10 seconds
</Resource>
```

Declarable in properties via

```
Foo = new://Resource?type=ActiveMQResourceAdapter
Foo.BrokerXmlConfig = broker:(tcp://localhost:61616)?useJmx=false
Foo.ServerUrl = vm://localhost?waitForStart=20000&async=true
Foo.DataSource = Default Unmanaged JDBC Database
Foo.StartupTimeout = 10 seconds
```

Configuration

BrokerXmlConfig

Broker configuration URI as defined by ActiveMQ see http://activemq.apache.org/broker-configuration-uri.html BrokerXmlConfig xbean:file:conf/activemq.xml - Requires xbean-spring.jar and dependencies

ServerUrl

Broker address

DataSource

DataSource for persistence messages

StartupTimeout

How long to wait for broker startup

javax.jms.ConnectionFactory

An ActiveMQ (JMS) connection factory.

Declarable in tomee.xml via

```
<Resource id="Foo" type="javax.jms.ConnectionFactory">
    ResourceAdapter = Default JMS Resource Adapter
    TransactionSupport = xa
    PoolMaxSize = 10
    PoolMinSize = 0
    ConnectionMaxWaitTime = 5 seconds
    ConnectionMaxIdleTime = 15 Minutes
</Resource>
```

Declarable in properties via

```
Foo = new://Resource?type=javax.jms.ConnectionFactory
Foo.ResourceAdapter = Default JMS Resource Adapter
Foo.TransactionSupport = xa
Foo.PoolMaxSize = 10
Foo.PoolMinSize = 0
Foo.ConnectionMaxWaitTime = 5 seconds
Foo.ConnectionMaxIdleTime = 15 Minutes
```

Configuration

ResourceAdapter

An ActiveMQ (JMS) resource adapter.

TransactionSupport

Specifies if the connection is enrolled in global transaction allowed values: `xa`, `local` or `none`. Default to `xa`.

PoolMaxSize

Maximum number of physical connection to the ActiveMQ broker.

PoolMinSize

Minimum number of physical connection to the ActiveMQ broker.

ConnectionMaxWaitTime

Maximum amount of time to wait for a connection.

ConnectionMaxIdleTime

Maximum amount of time a connection can be idle before being reclaimed.

javax.jms.Queue

An ActiveMQ (JMS) queue.

Declarable in tomee.xml via

```
<Resource id="Foo" type="javax.jms.Queue">
    # not set means id
    destination =
</Resource>
```

Declarable in properties via

```
Foo = new://Resource?type=javax.jms.Queue
# not set means id
Foo.destination =
```

Configuration

destination

Specifies the name of the queue

javax.jms.Topic

An ActiveMQ (JMS) topic.

Declarable in tomee.xml via

```
<Resource id="Foo" type="javax.jms.Topic">
    # not set means id
    destination =
</Resource>
```

Declarable in properties via

```
Foo = new://Resource?type=javax.jms.Topic
# not set means id
Foo.destination =
```

Configuration

destination

Specifies the name of the topic

org.omg.CORBA.ORB

NOTE	to use it you need to add an implementation of corba.

Declarable in tomee.xml via

```
<Resource id="Foo" type="org.omg.CORBA.ORB" />
```

Declarable in properties via

```
Foo = new://Resource?type=org.omg.CORBA.ORB
```

javax.mail.Session

A mail session.

Declarable in tomee.xml via

```
<Resource id="mail/mysession" type="javax.mail.Session">
  mail.transport.protocol = smtp
  mail.smtp.host = smtp.provider.com
  mail.smtp.auth = true
  mail.smtp.starttls.enable = true
  mail.smtp.port = 587
  mail.smtp.user = user@provider.com
  password = abcdefghij
</Resource>
```

Declarable in properties via

```
mail/mysession = new://Resource?type=javax.mail.Session
mail/mysession.mail.transport.protocol = smtp
mail/mysession.mail.smtp.host = smtp.provider.com
mail/mysession.mail.smtp.auth = true
mail/mysession.mail.smtp.starttls.enable = true
mail/mysession.mail.smtp.port = 587
mail/mysession.mail.smtp.user = user@provider.com
mail/mysession.password = abcdefghij
```

The properties are `javax.mail.Session` ones with the addition of `useDefault` which specifies if `getDefaultInstance()` or `getInstance` is used to create the session. `getDefaultInstance()` will ensure that several calls are done with the same configuration and return the same instance. For tomee it is likely better to rely on `getInstance()`(ie keep `useDefault` to false) and use `aliases` option of the resource to define an alias if you need to share the same instance accross multiple names.

ManagedExecutorService

A concurrency utility for EE executor service.

Declarable in tomee.xml via

```
<Resource id="Foo" type="ManagedExecutorService">
    Core = 5
    Max = 25
    KeepAlive = 5 s
    Queue = 15
    ThreadFactory = org.apache.openejb.threads.impl.ManagedThreadFactoryImpl
    Lazy = true
</Resource>
```

Declarable in properties via

```
Foo = new://Resource?type=ManagedExecutorService
Foo.Core = 5
Foo.Max = 25
Foo.KeepAlive = 5 s
Foo.Queue = 15
Foo.ThreadFactory = org.apache.openejb.threads.impl.ManagedThreadFactoryImpl
Foo.Lazy = true
```

Configuration

Core

The pool core size.

Max

The pool max size.

KeepAlive

The thread keep alive time (in duration format)

Queue

The queue type size.

ThreadFactory

The thread factory implementation class.

Lazy

If set to true the pool is created when first accessed otherwise it is created at startup.

ManagedScheduledExecutorService

Inherit from `ManagedExecutorService` and adds scheduling abilities.

Declarable in tomee.xml via

```
<Resource id="Foo" type="ManagedScheduledExecutorService">
    Core = 5
    ThreadFactory = org.apache.openejb.threads.impl.ManagedThreadFactoryImpl
    Lazy = true
</Resource>
```

Declarable in properties via

```
Foo = new://Resource?type=ManagedScheduledExecutorService
Foo.Core = 5
Foo.ThreadFactory = org.apache.openejb.threads.impl.ManagedThreadFactoryImpl
Foo.Lazy = true
```

Configuration

See ManagedExecutorService.

ManagedThreadFactory

A thread factory for a ManagedExecutorService.

Declarable in tomee.xml via

```
<Resource id="Foo" type="ManagedThreadFactory">
    Prefix = openejb-managed-thread-
    Lazy = true
</Resource>
```

Declarable in properties via

```
Foo = new://Resource?type=ManagedThreadFactory
Foo.Prefix = openejb-managed-thread-
Foo.Lazy = true
```

Configuration

Prefix

The thread prefix (suffixed with thread id).

ContextService

A concurrency utilities for JavaEE context service. It allows to create contextual proxies (inheriting from security, classloader...contexts).

Declarable in tomee.xml via

```
<Resource id="Foo" type="ContextService" />
```

Declarable in properties via

```
Foo = new://Resource?type=ContextService
```

JndiProvider: inject remote clients

A thread factory for a `ManagedExecutorService`. Default implementation is `org.apache.openejb.threads.impl.ManagedThreadFactoryImpl`.

Declarable in tomee.xml via

```
<Resource id="Foo" type="ManagedThreadFactory">
    Prefix = openejb-managed-thread-
    Lazy = true
</Resource>
```

Declarable in properties via

```
Foo = new://Resource?type=ManagedThreadFactory
Foo.Prefix = openejb-managed-thread-
Foo.Lazy = true
```

Configuration

Prefix

The thread prefix (suffixed with thread id).

ContextService

A concurrency utilities for JavaEE context service. It allows to create contextual proxies (inheriting from security, classloader...contexts).

Declarable in tomee.xml via

```
<Resource id="Foo" type="ContextService" />
```

Declarable in properties via

```
Foo = new://Resource?type=ContextService
```