Table of Contents

Every application has its own application zone with one configuration file. The name of the file is config.xml. This file should contain the whole configuration for your application.

Sometimes your configuration has external dependencies or you have a different configuration for your test and production system. In that case, it's better to use an external configuration which contains all system dependent properties. Such an external configuration will reduce your deploment/build effort because your config.xml can be the same for every system/server.

Usage

The config.xml can't be moved to an external location but it's possible to include external properties, e.g.:

config.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<application>
  <include>/home/tom/configuration.properties</include>
  <securitymanager>
<accesscontroller>com.sibvisions.security.MyAccessController</accesscon
troller>
  </securitymanager>
  <property name="password.rule" value="${pwdrule}"/>
</application>
```

The configuration.properties file:

configuration.properties

```
securitymanager.class = com.sibvisions.security.MySecurityManager
pwdrule = strict
```

The result, if you create a config.xml without external properties:

config.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<application>
  <securitymanager>
<accesscontroller>com.sibvisions.security.MyAccessController</accesscon
troller>
    <class>com.sibvisions.security.MySecurityManager</class>
  </securitymanager>
  <property name="password.rule" value="strict"/>
  <pwdrule>strict</pwdrule>

```

```
</application>
```

As you can see in above example, it's possible to use placeholders for values. The placeholders will be replaced automatically with values only if you use an external properties file. If you don't use the include tag, the placeholder replacement won't work!

It's also possible to use system and environment parameters as values with following prefixes:

```
${sys:databasePassword}
```

This will use the value of the system property databasePassword.

```
${env:databaseName}
```

This will use the value of the environment property databaseName.

If you don't use a custom prefix, e.g.

```
${databaseName}
```

the value search order will be:

1. imported property
2. system property
3. environment property

It's also possible to use only imported properties, with following prefix:

```
${imp:databaseName}
```

(an imported property is a value, which was found in an external properties file)

The value replacement syntax also supports partial values, e.g.

```
http://${url}:${port}/${context}/web/ui/
```

Attention

Be careful if you try to modify/save an application configuration that contains include tags. It won't work because of external references. It will be possible if you modify the file directly via XmlWorker but not via Zone or UpToDateConfigFile. In that case, an exception with following message will be thrown:

Couldn't save configuration file! The content was modified by <include> declarations.