Jyle Darling

M093437

Configuration Demonstration

AT1.7

Table of Contents

[The Configuration Process 1](#_Toc57380186)

[Why it’s used. 1](#_Toc57380187)

[When it’s used. 1](#_Toc57380188)

[How it’s used. 2](#_Toc57380189)

# The Configuration Process

## Why it’s used.

When an administrator receives a new application or version of the application from the development team, the application is usually configured for a development / testing environment. The application may use specific resource names and performance tuning settings that match the available resources on the target servers used in the development or QA environments where the application was last deployed.

Because development and testing environments can be quite different from the production environment where the application will be used, an administrator must configure the application to use resource names and performance tuning settings that represent what will be used in the production environment. The easiest way to achieve these changes in setting requirements is with the use of a configuration file.

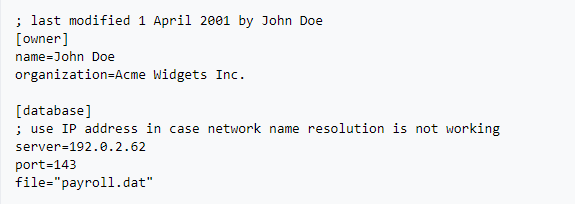
## When it’s used.

Configuration files can be used at different times during a applications development life cycle, but is commonly grouped into four different time frames.

1. Development Configuration – during development, a programmer creates J2EE deployment descriptors for the application or module. If the application is being deployed to a WebLogic Server environment, than a specific WebLogic deployment descriptors will be created.
2. Export Configuration – before releasing an application from development, a designer or programmer may optionally export the application’s deployment configuration to a WebLogic Server deployment plan. Exporting a configuration creates deployment plan variables for all or a subset of the deployment properties already defined by a developer in the application’s WebLogic Server deployment descriptor files.
3. Deployment-time configuration – An administrator configures the application before deploying the application into the target environment.
4. Post-deployment configuration – After an application has been deployed to a target environment, an administrator can reconfigure the application by redeploying with a new deployment plan or by using the Administration Console to update and redeploy an existing deployment plan.

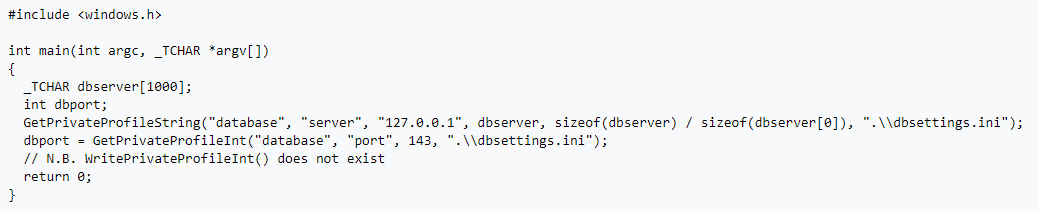
## How it’s used.

The following example of an .ini file, has two sections: one for the owner of the software, and one for a payroll database connection. Comments record the last person who modified the file and the reason for modification.



Under Windows, the *Profile API* is the programming interface used to read and write settings from classic Windows .ini files. For example, the GetPrivateProfileString function retrieves a string from the specified section in an initialization file.

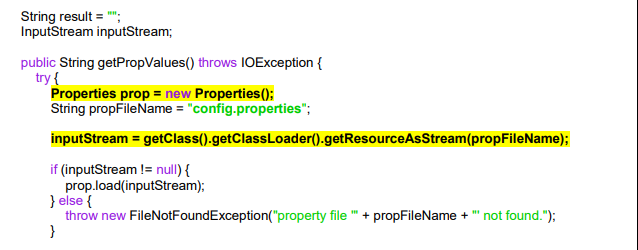
The following sample C program demonstrates reading property values from the above sample INI file (dbsettings.ini).



Using the example above, a string call could be made to fetch the *name* key from the *owner* section from a settings file called, say, *dbsettings.ini*. The returned value should be the string "John Doe":



An example of a Java Program accessing items contained in a configuration file is as follows.

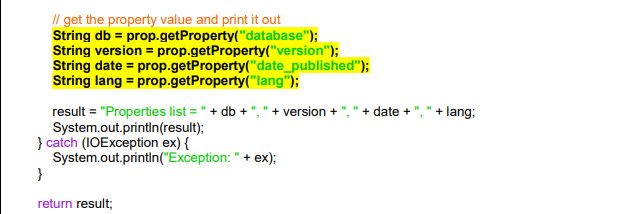


In this section we can see that we are declaring a properties object ‘prop’.

We are declaring the file name for the configuration file ‘config.properties’.

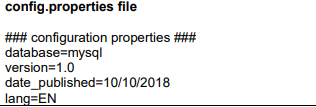
We are assigning a data input stream from the file information for the configuration file.

After checking that the file contains data, we are loading the data input stream into the properties object.



Now we are assigning strings values that we retrieve from the config file. We are scanning for certain phrases or keys in the config file that we can then map to our declared variables.

After this we are displaying all the retrieved values to a single string which is being returned by the method.



The configuration file contains the data in the above image, so when the method to retrieve this information is used with:

**System.out.println(result);**

The console application would display:

**Properties list = mysql, 1.0, 10/10/2018, EN**

Altering the information contained within the config.properties file would alter the output displayed to the console. So any other methods or tests using that information could easily be changed, without changing the application code itself.