

[database]

username=me

password\*=mypassword

then the first time the file is loaded by the process, the value is immediately encoded and the configuration file is updated, as follows:

[database]

username=me

password\*={xor} OS1sdjE7MyY=

[0035] The characters that follow the “{xor}” represent the encoded string representing the password. Encoding is a two-way process. If the algorithm used to encode the value is known, then it can also be decoded to get the original value. Therefore, if an encoded value is read from a configuration file, the original value can be determined.

[0036] The process proceeds as follows. First, an asterisk is added to the end of the property name of the property to be encoded. Next, when the file gets loaded into memory (see step 106 in FIG. 1), each property name is checked to see if any of them ends in the “\*” character. For each property with an “\*” character, if the property value has been previously encoded (i.e., the value starts with “{xor}”) then the value is decoded as it is loaded into memory (the configuration file is unchanged). If the property value has not been previously encoded (i.e., the value does not start with “{xor}”), then the property value in the configuration file is immediately overwritten with an encoded string. Any algorithm can be used to encode the string.

[0037] The above-described steps can be implemented using standard well-known programming techniques. The novelty of the above-described embodiment lies not in the specific programming techniques but in the use of the steps described to achieve the described results. Software programming code which embodies the present invention is typically stored in permanent storage of some type, such as permanent storage of a computer system. In a client/server environment, such software programming code may be stored with storage associated with a server. The software programming code may be embodied on any of a variety of known media for use with a data processing system, such as a diskette, or hard drive, or CD-ROM. The code may be distributed on such media, or may be distributed to users from the memory or storage of one computer system over a network of some type to other computer systems for use by users of such other systems. The techniques and methods for embodying software program code on physical media and/or distributing software code via networks are well known and will not be further discussed herein.

[0038] It will be understood that each element of the illustrations, and combinations of elements in the illustrations, can be implemented by general and/or special purpose hardware-based systems that perform the specified functions or steps, or by combinations of general and/or special-purpose hardware and computer instructions.

[0039] These program instructions may be provided to a processor to produce a machine, such that the instructions that execute on the processor create means for implementing the functions specified in the illustrations. The computer program instructions may be executed by a processor to

cause a series of operational steps to be performed by the processor to produce a computer-implemented process such that the instructions that execute on the processor provide steps for implementing the functions specified in the illustrations. Accordingly, the figures support combinations of means for performing the specified functions, combinations of steps for performing the specified functions, and program instruction means for performing the specified functions.

[0040] Although the present invention has been described with respect to a specific preferred embodiment thereof, various changes and modifications may be suggested to one skilled in the art and it is intended that the present invention encompass such changes and modifications as fall within the scope of the appended claims.

We claim:

1. A method of returning dynamic results from the processing of a configuration file by a processor, said processor including a memory, comprising:

loading into memory a configuration file comprising configuration properties having one or more variables;

resolving the variables of the configuration properties; and

replacing each variable with the results of its respective resolving step.

2. The method of claim 1, wherein said resolving step includes, for each configuration property:

detecting each variable in said configuration property;

resolving any recursive variables in said configuration property before resolving a primary variable in said configuration property; and

resolving said primary variable.

3. The method of claim 2, wherein said resolving step is executed when an external program attempts to access any configuration properties containing variables.

4. The method of claim 2, wherein said resolving step is performed using a variable resolver.

5. The method of claim 1, wherein said variables being resolved comprise variables of the format “file.section.property” which identify additional configuration properties.

6. The method of claim 1, wherein said variables being resolved comprise boolean expressions.

7. The method of claim 1, wherein said variables being resolved comprise integer math expressions.

8. The method of claim 1, wherein said variables being resolved comprise float math expressions.

9. The method of claim 1, wherein said variables being resolved comprise Java resource bundle properties.

10. A system of returning dynamic results from the processing of a configuration file by a processor, said processor including a memory, comprising:

means for loading into memory a configuration file comprising configuration properties having one or more variables;

means for resolving the variables of the configuration properties; and

means for replacing each variable with the results of its respective resolving step.

11. The system of claim 10, wherein said means for resolving includes, for each configuration property: