Implementation of MML tool engine based on MOC

Yan Li, Wang Liping

Information and Computer Engineering of PingXiang University, Ping Xiang, 337000, China

Keywords: MML; MOC; Tool Engine

Abstract. In this paper, based on the description of the abstract resource model, the unified description and modeling of MOC for configuration management are presented. Tool engine based on this unified resource model achieves the automation of construction for all MOC configuration file and the code, and further realizes the automation of MML complete construction, so as to flexibly and fastly generate export for MML.

Introduction

The traditional development process of MML (Man-Machine Language) is very complex. Firstly the design of a human-computer interaction command use CliBuilder to complete the allocation of command code, command name definition, definition of the input parameters and output parameters. And then provide the help document of command context and register the command callback function, further complete the MML command from the command message body to the interface message conversion, finally debugging whether command operation is successful[1]. In this process, developers work mechanically to do repetitive work, time-consuming and error-prone. In this case, it takes several hours of development to follow a traditional development process for each new command. Follow the original development process of MML for the database management system to provide a complete set of MML collections such as add, delete, change, check and other operations, that will consume the developer's great energy and time.

The full name of MOC is Meta-Object Compiler. The definition of MOC resource model uses DSL description language, and the MOC attributes are described in a uniform format[2]. The grammar is organized in the form of trees, with simple grammar, simple structure and clear level. The user can edit the MOC using any text editor. MOC provides compilers, performs syntax parsing, and completes the uniform loading of the model. Figure 1 shows the MOC syntax parsing tree. Based on the tool engine of the MOC implementation to achieve MML multi-tasking plug-in deployment, to facilitate the loading and unloading of the task actuator. Each task actuators are registered with the tool engine in the form of a service, and the task is scheduled by the tool engine. If the MOC definition is incorrect, the line number and message of error is presented to the user by the compiler, until the syntax is checked by the compiler.

Practice

MML generation engine. Similar to other task executable, MML Executor also registers services with Executor Manager in the form of a service, which is dispatched by the tool engine, and Executor Manager is the only way for the tool engine to get the service. Through the Executor Manager hub, the tool engine and the Executors is separated, to achieve the flexible loading and unloading of each Executor task (Figure 1).
For example, service registration can register the `mediator.tools.executors` attribute with a configuration file, so that the Executor Manager can use `System.getSystemProperty("mediator.executors")` to get services which are currently registered to the tool engine. Figure 2 shows registration method UML diagram of the toolkit and its MML generation engine. The tool engine scheduling center mainly maintains a thread pool for parallel processing of the task, that improves the efficiency of the tool. Secondly, it also includes a MOC parser, that completes the MOC syntax and semantic checking, and loads the MOC model [3, 4].

![Figure 1. The MOC syntax parsing tree](image)

MML Executor is responsible for the automatic export of MML and is registered with the tool engine in the form of a service, and generate all MML profile collections from the tool engine. MML Executor mainly maintains the MML data structure, MOC and field filters, code generation templates. MOC and field filters are mainly used to filter MOC and fields according to specific conditions. For example, filter out all MOCs that do not need to be manipulated; filter out invisible fields; filter out all fields beginning with `resveredBy`; filter out `RDN` fields; filter out non-read fields. MML Executor maintains the template file for the MML configuration file. With the functionality of the template, you can efficiently and quickly generate templates for MML commands. When the requirements change, simply modify the template, enable respond to the demand without modifying the code framework.

![Figure 2. MML generation engine](image)
MML and GUI. In addition to providing abundant GUI components, the database management system must provide a complete set of MML and make every operation on the GUI find a corresponding MML command. GUI and MML development are based on a unified resource model description, GUI and MML is unity naturally. If the model description changes, MML will go through the same process of change with the GUI, and maintain the consistency between them maximize. Simply, GUI and MML operations processes and functions are same as MOC exactly, they achieve the same function in two different ways, and there is no difference in addition.

As long as the completion of the resource model changes, that can achieve the scope of the GUI control constraints, the legitimacy check of the server-side relationship, and consistency improve of the flied help document context. Such as ADD, DEL, SET, GET buttons of GUI, the MML command generates the corresponding commands synchronously.

A frame node is registered in the MOC, used to describe the GUI on the left side of the MOC tree display and response mode; MML reuse such an abstract description, to ensure the GUI operation and MML operation consistency.

For example, frame node registration mode: frame (operations: "get"), where operations is the optional attribute, if no register, then the default value is operations: "add, del, set, get"; At this time it only allows users the query operation. The corresponding create, delete, and set buttons on GUI become stateless and can not be edited, and MML does not generate the corresponding add, del, and set commands.

Uniform frame tree. The frame node that defined by MOC is used to describe the abstract description of the tree diagram, manage the display of MML trees centrally, improve the user's perception and its friendliness. The method is to define a real parent-child relationship tree between the MOCs for organizing the DN sequence. A virtual parent-child relationship tree is defined at the same time, for organizing the frame tree and managing the MML tree centrally.

The frame view is optional and contains the following four sub-attributes:

1) virtualParent, optional attribute, used to describe the MOC in the tree on the father node, the default value is same as parent;
2) isPureVirtual, optional attribute, used to describe whether it is the dummy MOC only for the organization of the tree diagram, the default is false;
3) isVisible, optional attribute, used to describe whether the MOC displayed on the tree, the default value is true;
4) order, optional attribute, used to describe all the brother node order displayed on the tree diagram of the same father, if the order is empty, that means the order between the brothers is as the name dictionary of MOC, the default value is empty;
5) operations, optional attributes, used to describe the available status of GUI's add, delete, set, get button and MML command generation, the default value is "add, del, set, get".

Such as frame settings, frame(virtualParent: "Transmission", order: "2")

Where Transmission is a fake parent, it is the second subsequence for TransportNetwork. The isVisible is optional attribute, its defaults value is true, indicating that it needs to be displayed in the frame; isPureVirtual defaults value is false, indicating that it is the actual MOC in the model. On the contrary, isPureVirtual property of transmission is true, that means the model does not exist in the MOC, only for the organizing of various tree, which is as follows:

Builder.moc (name: "Transmission")
{
  Frame (virtualParent: "TransportNetwork", isPureVirtual: "true", order: "2")
}

Using above implementation, the organization of MML display and GUI frame tree is same exactly, ensure that MOC resource model changes and the MML terminal tree and GUI frame tree keeping the changes synchronously.

Command parameter maintenance. The MML command maintenance operations are divided into two categories, one is the node operation described above, the other is parameter maintenance
include the add parameters, delete parameters and the value modification. The MML command performs automatic maintenance of parameters by the following steps:

1) Add parameters
   The tool engine will list all the command that match the command name and fill in the value of the node in the expected result. The result of the operation is to add a new parameter value to the selected command as the new result of the expected result.

2) Delete parameters
   The tool engine will list all the command that match the command name, parameter name, and parameter value (or no fill). The result of the operation is to delete the selected command parameter from the message.

3) modification of parameter values
   When the parameter type is data, using the data offset operation fill the expected results with the corresponding offset, then the operation for the parameters of the new value is based on the original parameters value plus (or minus) the offset.
   When the parameter type is string, replacing by a regular expression, fill in the expected results with the expression:
   \[\text{Regexp} = \{\ \star\} \text{replace} = \{.\}\]
   Fill the matching value of the regular expression and replace the string in the braces, the operation will be replaced selected original value with the new value directly.

4) Modification of parameter description
   (1) When the command parameter has changed, but the command parameter description has not changed, the new version of the command parameters described as a keyword, automatically update the command parameters for the new version of the parameters;
   (2) When the command parameter description has changed, but the command parameter has not changed, the new version of the command parameters for the keyword, automatic update command parameter description for the new version of the description;
   (3) When the command parameter description and command parameters are both changed: update the command parameters of the man-machine command model, modify the command parameter description to the old version, you can convert to step (1) update method.

Summary

Compared to the traditional MML development process, achievement MML tool engine has flexible, fast features. Provides a unified resource model description, provides export tool to achieve a flexible way to deploy, manage data resources, and its front-end and back-end message interface, achieve automation of the client, server configuration files and code, including MML automation.

MOC uses a special deployment method for MML, after GUI implementation, as development of above methods, developers only need to maintain a set of resource model definition file, the workload modifying configuration parameter is reduced greatly, the effect maintaining information consistency is well; Secondly, achieve unified between GUI and MML one-click automatic export technology of MML achieves the MML zero cost development.

References


