JSC Engineering Orbital Dynamics NamedItem Model

Simulation and Graphics Branch (ER7)
Software, Robotics, and Simulation Division
Engineering Directorate

Package Release JEOD v5.0

Document Revision 1.0
July 2022

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

JSC Engineering Orbital Dynamics
NamedItem Model

Document Revision 1.0
July 2022

Dr. Robert Shelton

Simulation and Graphics Branch (ER7)
Software, Robotics, and Simulation Division
Engineering Directorate

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

Abstract

The NamedItem Model is really not a model in any normal sense of the word. It is simply a class which contains several static methods for concatenating 1 or more character strings into a new string in which the original strings are separated by a dot, e.g. construct_name("foo", "bar") returns "foo.bar".

Contents

Chapter 1

Introduction

1.1 Model Description

The NamedItem Model is basically a wrapper class for a set of static methods which construct a name as a dot-conjoined string. These functions return a newly-allocated string, and the caller is responsible for freeing the allocated memory via JEOD_DELETE.

There is also a static method which, given a prefix and a dot-conjoined name, will find the part of the name that follows the prefix. If the name is of the form "prefix.suffix", the suffix function returns a pointer to "suffix". If the name does not start with "prefix." the function returns the input name.

Most of these methods are used throughout the JEOD code. Creating dot-conjoined strings in their own allocated memory is an extremely common task best handled by a central utility such as the NamedItem class. There are two additional public static methods – vconstruct_name and va_construct_name which are only used presently by the construct_name methods of the NamedItem class.

1.2 Document History

Author	Date	Revision	Description
Robert O. Shelton	November, 2009	1.0	JEOD 2.0 Release
NO REVISIONS			

1.3 Document Organization

This document is formatted in accordance with the NASA Software Engineering Requirements Standard [1] and is organized into the following chapters:

Chapter 1: Introduction - This introduction contains three sections: description of model, document history, and organization. The first section provides the introduction to the NamedItem

Model and its reason for existence. The second section displays the history of this document which includes author, date, and reason for each revision. The final section contains a description of how the document is organized.

Chapter 2: Product Requirements - Describes requirements for the NamedItem Model.

Chapter 3: Product Specification - Describes the underlying theory, architecture, and design of the NamedItem Model in detail. It is organized in three sections: Conceptual Design, Mathematical Formulations, and Detailed Design.

Chapter 4: User Guide - Describes how to use the NamedItem Model in a Trick simulation. It is broken into three sections to represent the JEOD defined user types: Analysts or users of simulations (Analysis), Integrators or developers of simulations (Integration), and Model Extenders (Extension).

Chapter 5: Verification and Validation - Contains NamedItem Model verification and validation procedures and results.

Chapter 2

Product Requirements

This model shall meet the JEOD project requirements specified in the JEOD top-level document.

Requirement NamedItem_1: String Construction

Requirement:
The NamedItem Model shall offer static functions capable of accepting one or more character strings, computing space requirements for a dot-conjoined concatenation of the input strings, allocating the required space (including the null character terminating the string), and returning the newly created dot-conjoined string.

Rationale:
This is the function of the construct_name methods

Verification:
Inspection

Requirement NamedItem_2: Suffix Determination

Requirement:
The NamedItem Model shall offer a static method capable of finding the part of a dot-conjoined name following a given prefix. For names of the form "prefix.suffix", this function shall return a pointer to "suffix". The function shall return the input name if the name does not start with "prefix.".

Rationale:
Determination of a suffix in this manner supports JEOD naming and message conventions.

Verification:
Inspection

Chapter 3

Product Specification

3.1 Conceptual Design

The NamedItem Model is a single class with no instance fields or methods. It's content consists of 7 construct_name static methods which construct dot-conjoined names from 1 to 7 input string arguments; two generic methods for constructing dot-conjoined strings from a variable number of arguments (null terminated); and a method for determining a suffix of a dot-conjoined string given an input prefix. These methods are basic C++ functions which serve their intended purpose and play no role in the object design of JEOD.

3.2 File Inventory

Files Comprising the NamedItem Model
named_item:
docs
include
src

named_item/docs:
NamedItem.pdf
tex

named_item/docs/tex:
makefile
NamedItemAbstract.tex
NamedItem.bib
NamedItemChapters.tex
NamedItemReq.tex
NamedItemSpec.tex
NamedItem.sty
Continued on next page

Continued from previous page
NamedItem.tex

named_item/include:
named_item.hh

named_item/src:
named_item.cc
Table 3.1: File Inventory

Chapter 4

User Guide

The NamedItem Model is never referenced in any JEOD simulations, thus the Analyst and Integrator sections of this chapter are omitted. Readers wishing to use the NamedItem Model in their extensions of JEOD classes are directed to the API reference.

Chapter 5

Verification and Validation

5.1 Verification

Verification of requirements NamedItem_1 and NamedItem_2 is performed by code inspection.

5.2 Validation

There are no validation cases specifically designed for the NamedItem Model; however, the functionality is thoroughly exercised throughout the JEOD software package. This usage constitutes an implicit validation of nominal performance.

Bibliography

[1] NASA. NASA Software Engineering Requirements. Technical Report NPR-7150.2, NASA, NASA Headquarters, Washington, D.C., September 2004.