Variant User Guide

Version 0.5 Beta

September 2015

# Core

## Overview

## Experiment Design

### Introduction

…

## Experiment Execution

### System Configuration

The Variant engine is configured with a set of application properties. Typically, they will be packaged into properties files and passed to the engine at either compilation or run time. Properties may also be overridden individually at run time, as explained below.

Compile time configuration involves supplying the names of application properties files to the Variant.bootstrap() call. If supplied, each argument is understood as the file name, as a Java classpath resource. When, at run time, the engine is looking for the value of a particular property, these files will be scanned left to right until the first match is found. If a value wasn’t found in any of the supplied files, or if no files were supplied, the default value is used, as defiled in the /variant-default.props file found inside the core jar file.

You may override compile time configuration in the following ways:

* Place the /variant.props file on the class path. The Variant engine will always look for this file and, if found, consult it first, before any of the files supplied at compilation time.
* Pass the –Dvariant.props.resource=... command line parameter to the JVM. The supplied value is understood to be a classpath resource name, which must exist.
* Pass the –Dvariant.props.file=... command line parameter to the JVM. The supplied value is understood to be an OS file, which must exist.

It is an error to supply both ‑Dvariant.props.resource and ‑Dvariant.props.resource.

## Experiment Analysis

## Core API Reference

### Public Packages

|  |  |
| --- | --- |
| Package | Description |
| com.variant.core |  |
| com.variant.core.annotations |  |
| com.variant.core.ext |  |
| com.variant.core.schema |  |
|  |  |
|  |  |

### Class Variant

static void bootstrap(String...propsResourceName) : Variant

Bootstrap the Variant engine with an optional list of application properties files as resource names. If more than one resource file is passed, they are scanned left to right until the first match is found. If a property was not found in any of the supplied properties files (or none was supplied), a default value is used, as specified by the /variant-default.props file found inside the core jar file.

Each argument is interpreted as the name of a classpath resource. If one of argument names a resource file that cannot be located on the classpath, the bootstrap() call fails.

Note that at run time, these values may be overridden from the command line, as explained in TODO.

### Class …

# Web API

## Path Matching

Web API has the following implementation of the path matcher:

* Symbol ‘/’ always stands for the path separator. Path must start with ‘/’.
* Any sequence of symbols between two consequtive symbols '/' is taken to be a literal, unless the first letter of the sequence is a tilde '~' in which case the string immediately following the tilde and until, but not including, the next unescaped ‘/’ is considered a regular expression. Complete regular expression syntax is supported[[1]](#footnote--1).
* Although symbol ‘/’ does not have a special meaning in the regular expression grammar, it does for Variant: this is how Variant decides where a regular expression ends. Therefore, if ‘/’ must be included in the regular expression, it must be escaped with the ‘\’ symbol, like any other special character. By including ‘/’ symbols in the regular expression, it is possible to match variable sections in the middle of a path.
* Symbol ‘//’ can be used anywhere, where ‘/’ can be used, and is a shortcut for ‘/~.\*/’. On other words, ‘//’ will match any string. Note that ‘///’ is legal but superfluous because it will be expanded to ‘/~.\*/~.\*/’.
* The very last ‘/’ of the pattern is not significant, i.e. Variant will remove it, if present, from both the pattern and the path after all ‘//’ are expanded. This enables easy prefix match: ‘/user//’ will match any path that starts with ‘/user/’.

Examples:

|  |  |  |
| --- | --- | --- |
| Path | Will Match | Will Not Match |
| /user | /user  /user/ | /user/new |
| /user// | /user  /user/  /user/new | /service/user/ |
| /user//.html | /user/new/error.html | /user/error |

1. See, e.g. http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html. [↑](#footnote-ref--1)