

ADAMS

Advanced **D**ata mining **A**nd **M**achine learning **S**ystem

Module: adams-groovy-rest



REST

Peter Reutemann

January 28, 2020

©2019



THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato



Except where otherwise noted, this work is licensed under
<http://creativecommons.org/licenses/by-sa/4.0/>

Contents

1	Introduction	5
2	Flow	7
2.1	Writing a unparamtrized plugin	8
2.2	Writing a parametrized plugin	9
	Bibliography	11

Chapter 1

Introduction

Developing REST webservices using JAX-RS[2] is rather easy. However, during development, services can go through many iterations. Having to deploy a new build every single time can be time-consuming (and annoying). In order to fill the gap, the *adams-groovy-rest* module allows you to write your REST plugins in Groovy[3], therefore avoiding the recompilation of code, resulting in a faster turn-around time for your services.

Chapter 2

Flow

Use the following *RESTProvider* in your *RESTServer* standalone:

```
adams.flow.rest.GroovyServer
```

This provider allows you to point to an arbitrary number of Groovy scripts that make up your REST service. It automatically sets the flow context if the scripts should implement the *adams.flow.core.FlowContextHandler* interface. It also propagates its own logging level to the scripts, if they should implement the *adams.core.logging.LoggingLevelHandler* interface. There are two types of scripts:

- without parameters
- with parameters (to influence behavior)

2.1 Writing a unparamtrized plugin

Each of the Groovy scripts needs to contain a single class which either implements the *RESTPlugin* interface or is derived from the *AbstractRESTPlugin* or *AbstractRESTPluginWithFlowContext* super classes (the latter should be used if you require access to variables or internal storage in the flow). The rest of the code uses the same JAX-RS annotations as for writing a Java class.

Below is an example of a simple *echo* client, which just sends back the data it received as part of the URL of the query¹:

```
import adams.flow.rest.AbstractRESTPlugin
import javax.ws.rs.GET
import javax.ws.rs.Path
import javax.ws.rs.PathParam
import javax.ws.rs.Produces

class Echo extends AbstractRESTPlugin {

    @Override
    String globalInfo() {
        return "simple echo server with optional uppercasing of the input"
    }

    @GET
    @Path("/echo/{input}")
    @Produces("text/plain")
    public String ping(@PathParam("input") String input) {
        getLogger().info("input: " + input)
        return input
    }
}
```

¹adams-groovy-rest_echo.groovy

2.2 Writing a parametrized plugin

With more complex REST services, it can become necessary to add parameters to the service itself: e.g., what model to load from internal storage to process the data. For such REST plugins, you need to also implement the *adams.flow.core.AdditionalOptionsHandler* interface in your script. To make things easier, just use *adams.flow.rest.AbstractParametrizedGroovyRESTPlugin* to derive your plugin from.

Parametrized scripts can be supplied in the *GroovyServer* standalone via the *parametrizedScripts* option. Each of these wrappers allows you to specify a script file and what options to supply to the script once loaded. These options are simple *key=value* pairs and support variable expansion, e.g., *uppercase=@{some.variable}*.

The example plugin below² is a variation of the previous echo plugin. It supports the *uppercase* parameter for influencing how to return the received input string. For that purpose, it uses the *getAdditionalOptions()* (which will have parsed the options supplied to the script when loading it) to access the boolean parameter *uppercase*:

```
import adams.flow.rest.AbstractParametrizedGroovyRESTPlugin
import javax.ws.rs.GET
import javax.ws.rs.Path
import javax.ws.rs.PathParam
import javax.ws.rs.Produces

class Echo extends AbstractParametrizedGroovyRESTPlugin {

    @Override
    String globalInfo() {
        return "simple echo server"
    }

    @GET
    @Path("/echo2/{input}")
    @Produces("text/plain")
    public String ping(@PathParam("input") String input) {
        getLogger().info("input: " + input)
        if (getAdditionalOptions().getBoolean("uppercase"))
            return input.toUpperCase()
        else
            return input
    }
}
```

²adams-groovy-rest.echo2.groovy

Bibliography

- [1] *ADAMS* – Advanced Data mining and Machine learning System
<https://adams.cms.waikato.ac.nz/>
- [2] *JAX-RS* – Java API for RESTful Web Services
https://en.wikipedia.org/wiki/Java_API_for_RESTful_Web_Services
- [3] *Groovy* – An agile dynamic language for the Java Platform
<http://www.groovy-lang.org/>