Taco Documentation

Release 0.0.0

Graham Bell

January 25, 2015

Contents

Taco is a system for bridging between scripting languages. Its goal is to allow you to call routines written for one language from another. It does this by running the second language interpreter in a sub-process, and passing messages about actions to be performed inside that interpreter.

In principle, to interface scripting languages it might be preferable to embed the interpreter for one as an extension of the other. However this might not be convenient or possible, and would need to be repeated for each combination of languages. Instead Taco only requires a “client” module and “server” script for each language, which should be straightforward to install, and its messages are designed to be generic so that they can be used between any combination of languages.

This documentation describes the Taco system. For a list of language implementations, please see the Taco Homepage.

CHAPTER 1

Contents

1.1 Command Actions

Command actions are sent by the client to instruct the server to perform various tasks. After sending a command, the client should wait to recieve one of the *Response Actions* from the server.

1.1.1 Call class method

call_class_method

Instructs the server to call the given class method.

The `context` attribute is described on the `call_function` page.

```
{
    "action": "call_class_method",
    "name": "NAME",
    "class": "CLASS",
    "args": [],
    "kwargs": {},
    "context": null
}
```

1.1.2 Call function

call_function

Instructs the server to call the given function.

```
{
    "action": "call_function",
    "name": "NAME",
    "args": [],
    "kwargs": {},
    "context": null
}
```

Context

This action (and the related actions `call_class_method` and `call_method`) include a `context` attribute in order to support context-sensitive languages. Clients for such languages may attempt to set this attribute automatically, and servers for languages which are not sensitive to context are free to ignore it.

It can take the following values:

- `"scalar"`
- `"list"`
- `"map"`
- `"void"`
- null

1.1.3 Call method

`call_method`

Calls a method of an object. The object must already be in the server's object cache, and is referred to by the `number` attribute. This number will have been provided in a `_Taco_Object_` reference sent by the server.

The `context` attribute is described on the `call_function` page.

```
{
    "action": "call_method",
    "name": "NAME",
    "number": 1,
    "args": [],
    "kwargs": {},
    "context": null
}
```

1.1.4 Construct object

`construct_object`

Instructs the server to call the constructor for the given class. If successfull, an `_Taco_Object_` reference should be returned in the `result` message.

```
{
    "action": "construct_object",
    "class": "NAME",
    "args": [],
    "kwargs": {}
}
```

1.1.5 Destroy object

`destroy_object`

This action allows the server to remove the corresponding object from its object cache. The object is identified by the `number` attribute. The client should not attempt to refer to the object again after permitting its destruction.

```
{
    "action": "destroy_object",
    "number": 1
}
```

1.1.6 Get attribute

get_attribute

Requests the value of an attribute of one of the objects in the server's object cache. The object is identified by the number attribute.

```
{
    "action": "get_attribute",
    "name": "NAME",
    "number": 1
}
```

1.1.7 Get class attribute

get_class_attribute

Requests the value of a class attribute.

```
{
    "action": "get_class_attribute",
    "name": "NAME",
    "class": "CLASS"
}
```

1.1.8 Get value

get_value

Requests the value of a variable.

```
{
    "action": "get_value",
    "name": "NAME"
}
```

1.1.9 Import module

import_module

Instructs the server to attempt to load the module specified by the name attribute. The interpretation of the args (arguments) and kwargs (keyword arguments) attributes can differ between different Taco server implementations as is appropriate for different programming languages.

```
{
    "action": "import_module",
    "name": "NAME",
    "args": [],
    "kwargs": {}
}
```

1.1.10 Set attribute

set_attribute

Sets the value of an attribute of an object in the server's object cache. The object is identified by the `number` message attribute.

```
{
    "action": "set_attribute",
    "name": "NAME",
    "number": 1,
    "value": "VALUE"
}
```

1.1.11 Set class attribute

set_class_attribute

Sets the value of a class attribute.

```
{
    "action": "set_class_attribute",
    "name": "NAME",
    "class": "CLASS",
    "value": "VALUE"
}
```

1.1.12 Set value

set_value

Sets the value of a variable.

```
{
    "action": "set_value",
    "name": "NAME",
    "value": "VALUE"
}
```

1.2 Response Actions

This section describes messages which a Taco server can use to respond to *Command Actions*.

1.2.1 Exception

exception

The exception action signifies that an exception or error of some type occurred while a command action was being handled. It can be used by the server to raise errors explicitly, such as when an unrecognised action message is received, as well as for exceptions trapped by the server while attempting to carry out commands.

The message attribute contains a string representation of the error.

```
{
    "action": "exception",
    "message": "MESSAGE"
}
```

1.2.2 Result

result

This message is used when a command has completed successfully. The result attribute holds the return value, if there is one, or null otherwise.

```
{
    "action": "result",
    "result": null
}
```

1.3 Special Objects

This section describes special objects which can appear as part of a Taco message. These JSON objects are embedded in the values of the message's attributes. For example they could appear in the args attribute of the call_function action, or the result attribute of the result action.

1.3.1 Object Reference

_Taco_Object_

An object reference is used to refer to an object which has been constructed on the server side. In other words, it refers to an instance of a class in the server's language, not to JSON objects in Taco messages. When a result action would contain an object, the object should be stored in a cache by the server, and the reference to it replaced with an object reference. This uses an integer to identify the object. Object references can also be used by the client, for example to allow them to be passed as function arguments.

```
{
    "_Taco_Object_": 1
}
```

CHAPTER 2

Indices and tables

- *genindex*
- *search*