RESTful Web Services

Using the RESTful Web Service plugin business logic can be exposed as a RESTful Web Service. The RESTful Web Service plugin does not contain any client side functions or properties, it is a 100% server side operating plugin.

To create a RESTful Web Service, create a form in a solution and add one or more of the following methods to the form:

- ws_read():Object to **retrieve data**
  By performing an HTTP GET on the url <serverUrl>/servoy-service/rest_ws/solutionName/formName, the ws_read() method will be invoked. **Arguments can be specified** in the url, like <serverUrl>/servoy-service/rest_ws/solutionName/formName/value1/value2. The arguments will be applied to the invokation of ws_read().
  The method **must return** a JavaScript object. The object will be serialized and placed in the body of the HTTP Response. If the return value of the method is null, a NOT_FOUND response (HTTP 404) will be generated
- ws_create(content):Object to **add data**
  By performing an HTTP POST on the url <serverUrl>/servoy-service/rest_ws/solutionName/formName, the ws_create() method will be invoked. **Data has to be supplied** in the body of the HTTP Request. Additional **arguments can be specified** in the url, like <serverUrl>/servoy-service/rest_ws/solutionName/formName/value1/value2. The arguments will be applied to the invocation of ws_delete(content), after the content parameter.
  The method **can return** a JavaScript object. The object will be serialized and placed in the body of the HTTP Response.
- ws_delete():Boolean to **remove data**
  By performing an HTTP DELETE on the url <serverUrl>/servoy-service/rest_ws/solutionName/formName, the ws_delete() method will be invoked. **Arguments can be specified** in the url, like <serverUrl>/servoy-service/rest_ws/solutionName/formName/value1/value2. The arguments will be applied to the invocation of ws_delete().
  The method **has to return** a Boolean value:
  - true: to indicate successful deletion. This result will generate an OK response (HTTP 200)
  - false: to indicate delete failure. This response will generate a NOT_FOUND response (HTTP 404)
- ws_update(content):Boolean to **update data**
  By performing an HTTP PUT on the url <serverUrl>/servoy-service/rest_ws/solutionName/formName, the ws_update() method will be invoked. **Data has to be supplied** in the body of the HTTP request. Additional **arguments can be specified** in the url, like <serverUrl>/servoy-service/rest_ws/solutionName/formName/value1/value2. The arguments will be applied to the invocation of ws_delete(content), after the content parameter.
  The method **has to return** a Boolean value:
  - true: to indicate successful update. This result will generate an OK response (HTTP 200)
  - false: to indicate update failure. This response will generate a NOT_FOUND response (HTTP 404)

In case the matching method for the specific HTTP operation (GET, POST, DELETE or PUT) does not exists on the form, an INTERNAL_SERVER_ERROR response (HTTP 500) will be generated.

Stateless

RESTful Web Services are to be stateless. As subsequent requests to the RESTful Web Service API might be handled by different headless clients in the pool of clients configured for the plugin, do not use any state in between calls to the API. This means at least the following:

- Do not use global or form variables
- Do not use the solution model API
- Do not alter the state of the a form's UI
- Do save or rollback any unsaved changes before the end of the method

HTTP Request

For the POST and PUT operation (resp. ws_create() and ws_update() methods), data has to be supplied in the body of the HTTP Request. If the content in the body of the HTTP Request is missing, a NO_CONTENT response will be generates (HTTP 204).

The supported Content-Types are JSON (application/json) and XML (application/xml). The Content-Type can be explicitly set in the header of the HTTP Request:

```
Content-Type: application/json; charset=utf-8
```

```
Content-Type: application/xml; charset=utf-8
```

Note: the charset is optional. If not specified, utf-8 will be assumed.

If no valid Content-Type is set, the plugin will try to establish the type of the content based on the first character of the content:

- '\Accept: application/json

```
By default, the response will be encoded with the UTF-8 charset. If the HTTP Request specified a
different encoding, that will be used instead. If the encoding of the response needs to be different
than the request encoding, this can be specified in the HTTP Request by setting the charset value in the
Accept header:
```

Accept: application/json; charset=UTF-16

```
h4.Authentication/Authorisation
The RESTful Web service plugin can be configured to check authentication/authorisation.
The plugins server property rest_ws_plugin_authorized_groups can be set with a comma separated list of
groups defined in the built-in security system of Servoy.
When the property is filled with usergroups, HTTP Basic authentication is enabled for all webservice
requests. The username/password supplied in the HTTP Request is validated against the users registered
in Servoy's built-in security system and additionally against group membership. Access is denied if the
user does not exists or the supplied password is incorrect, or the user doesn't belong to one of the
specified groups.

When access is denied, an UNAUTHORIZED response is generated (HTTP 401).

h4.JSONP support
The plugin supports so-called JSONP: a variation of JSON that allows cross domain data retrieval. The
JSONP variation can be invoked by added a "callback" parameter to the HTTP Request URL:
```

http://domain:port/servoy-service/rest_ws/\{solutionName}/{formName}?callback={callbackFunctionName}

```
When invoked with the value "myCallback" for the "callback" parameter, the returned JSON value will be
wrapped in a function call to the "myCallback" function:
```

myCallback(
Unknown macro: { "hello" }
)

Pool of Clients

To service the requests to the RESTful Web service API, the plugin creates a pool of (headless) clients. The maximum number of clients allowed can be set using the "rest_ws_plugin_client_pool_size" property of the plugin (default value = 5).

When there are more concurrent requests than the number of clients in the pool, by default the requests will wait until a client becomes available in the pool. This behavior can be altered by setting the "rest_ws_plugin_client_pool_exhausted_action" property of the plugin. The following values are supported for this property:

- block (default): requests will wait untill a client becomes available
- fail: the request will fail. The API will generate a SERVICE_UNAVAILABLE response (HTTP 503)
- grow: allows the pool to temporarily grow, by starting additional clients. These will be automatically removed when not required anymore.

Servoy Cluster

The RESTful Web service plugin uses a pool of headless clients to service the requests. When operated within a Servoy Cluster, note that poolsize is set per Servoy Application Server.

Samples

A sample solution is included in the Servoy distribution (servoy_sample_rest_ws.servoy), detailing how to retrieve data from the http request and to return a response.

For more information on RESTful Web Services, see:

http://en.wikipedia.org/wiki/Representational_State_Transfer
http://www.infoq.com/articles/rest-introduction
http://www.ibm.com/developerworks/webservices/library/ws-restful/
http://home.ccil.org/~cowan/restws.pdf

Unknown macro: {table}

{column:padding=0px|width=80px}{column}{column}{column}

Unknown macro: {th}

Unknown macro: {td}

rest_ws_plugin_authorized_groups

Unknown macro: {td}

rest_ws_plugin_client_pool_exhausted_action

Unknown macro: {td}

rest_ws_plugin_client_pool_size

Unknown macro: {table}

{column:padding=0px|width=100%}{column}

Unknown macro: {th}

Unknown macro: {td}

rest_ws_plugin_authorized_groups

Unknown macro: {td}

Only authenticated users in the listed groups (comma-separated) have access, when left empty unauthorised access is allowed

Unknown macro: {td}

rest_ws_plugin_client_pool_exhausted_action

Unknown macro: {td}

The following values are supported for this property:
block (default): requests will wait untill a client becomes available
fail: the request will fail. The API will generate a SERVICE_UNAVAILABLE response (HTTP 503)
grow: allows the pool to temporarily grow, by starting additional clients. These will be automatically removed when not required anymore.

Unknown macro: {td}

rest_ws_plugin_client_pool_size

Unknown macro: {td}

Max number of clients used (this defines the number of concurrent requests and licences used), default = 5

Unknown macro: {td}