
TEMPO
REST services

Table of Contents

Introduction

REST (REpresentational State Transfer) is an architectural style, and an approach to communications that is often used in the development of *Web services*. The use of REST is often preferred over the more heavyweight *SOAP* (Simple Object Access Protocol) style because REST does not leverage as much bandwidth, which makes it a better fit for use over the Internet. The SOAP approach requires writing or using a provided server program (to serve data) and a client program (to request data).

REST'S decoupled architecture, and lighter weight communications between producer and consumer, make REST a popular building style for cloud-based APIs, such as those provided by Amazon, Microsoft, and Google. When Web services use REST architecture, they are calledRESTful APIs (Application Programming Interfaces) or REST APIs.

REST architecture involves reading a designated Web page that contains an XML file. The XML file describes and includes the desired content. Once dynamically defined, consumers may access the interface.

REST, which typically runs over HTTP (Hypertext Transfer Protocol), has several architectural constraints:

I. Decouples consumers from producers

II. Stateless existence

III. Able to leverage a cache

IV. Leverages a layered system

V. Leverages a uniform interface

REST is often used in mobile applications, social networking Web sites, mashup tools, and automated business processes. The REST style emphasizes that interactions between clients and services is enhanced by having a limited number of operations (verbs). Flexibility is provided by assigning resources (nouns) their own unique Universal Resource Identifiers (URIs). Because each verb has a specific meaning (GET, POST, PUT and DELETE), REST avoids ambiguity.

There are some downsides. In the world of REST, there is no direct support for generating a client from server-side-generated metadata. SOAP is able to support this with Web Service Description Language (WSDL).

REST provides the following advantages, specifically advantages over leveraging SOAP:

- RESTful Web services are easy to leverage by most tools, including those that are free and inexpensive. REST is becoming the dial tone for systems interaction, including the use of RESTful Web services, which are, for the most part, the waycloud providers externalize their cloud services.
- SOAP services are much harder to scale than RESTful services. Thus, REST is often chosen as the architecture for services that are exposed via the Internet (likeFacebook, MySpace, Twitter, and most public cloud providers).
- The learning curve seems to be reduced. Developers are able to make use of REST from within applications faster than they can with SOAP. This saves time, which saves money.
- REST uses a smaller message format than SOAP. SOAP uses XML for all messages, which makes the message size much larger, and thus less efficient. This means REST provides better performance, as well as lowers costs over time. Moreover, there is no intensive processing required, thus it's much faster than traditional SOAP.
- REST is designed for use over the Open Internet/Web. This is a better choice for Web scale applications, and certainly for cloud-based platforms.

Moving forward, REST is likely to continue its growth as enterprises seek to provide open and well-defined interfaces for application and infrastructure services. The growth of public and private cloud computing is driving much of this demand, and will continue to drive growth into the future.

Pre-requisite softwares to use REST services

- Web browser (Internet Explorer, Google Chrome, Mozilla Firefox)
- SoapUI or Advanced REST client
- MySQL Workbench

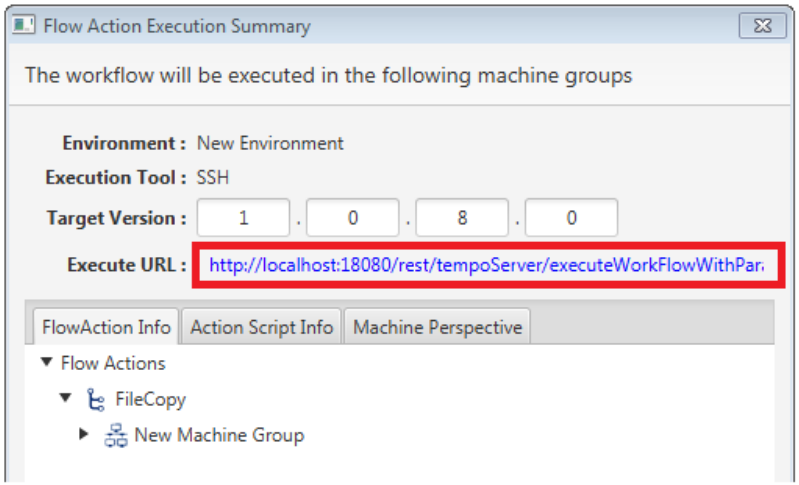
Methods to use the REST services

- HTTP - *Only GET services can be executed*
- SoapUI - *GET and POST methods can be executed*

HTTP method

Only GET services can be run through a web browser and before using it, you have to create workflows through TEMPO Workstation. Then you can identify id number by opening the table from MySQL Workbench.

Note: Once you click the execute button from workstation it will create a link as below, by using this link also you can identify table ids (Ex:- projecteId, workflowId)



Execute workflows by using GET services

- You can execute by pasting below links in the browser address/URL bar and have to change "{}" according to relevant ids (Ex:-projectId, workflowId) before press enter.

Ex:- *http://localhost:18080/rest/tempoServer/executeWorkFlow/3/2/4/SSH/5/*

projectId — *versionId*

GET Service for execute the workflow -

http://localhost:18080/rest/tempoServer/executeWorkFlow/{projectId}/{workFlowId}/{environmentId}/{toolType}/{versionId}

GET Service for execute the workflow with workflow parameters that need to replace in workflow properties -

http://localhost:18080/rest/tempoServer/executeWorkFlowWithParam/{projectId}/{workFlowId}/{environmentId}/{toolType}/{versionId}/{param}

GET Service for execute the workflow with machines that need to be execute workflow -

http://localhost:18080/rest/tempoServer/executeWorkFlowWithMachine/{projectId}/{workFlowId}/{environmentId}/{toolType}/{versionId}/{machine}

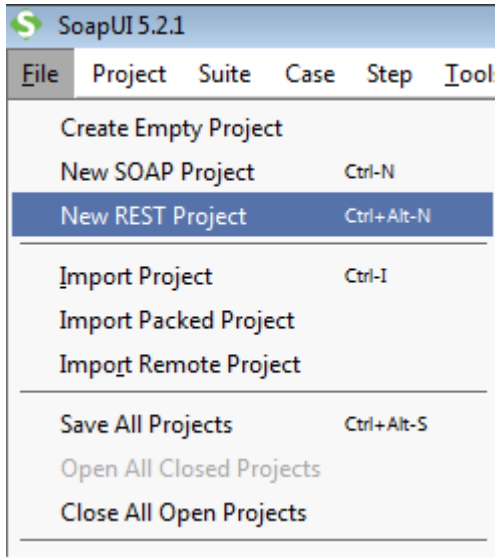
GET Service for execute the workflow with workflow parameters and execute machines -

http://localhost:18080/rest/tempoServer/executeWorkFlowWithParamAndMachine/{projectId}/{workFlowId}/{environmentId}/{toolType}/{versionId}/{param}/{machine}

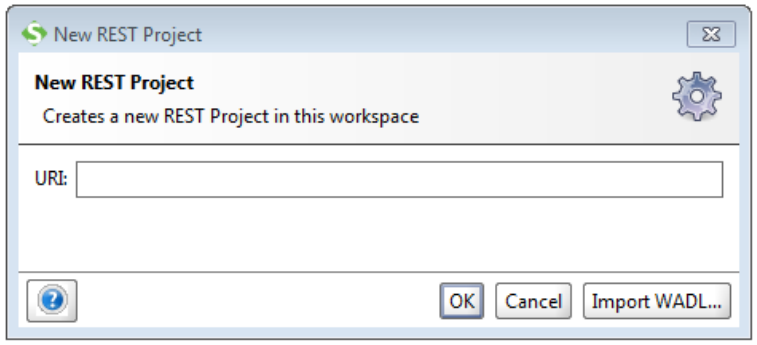
SoapUI method

1. Start by creating a new REST project:

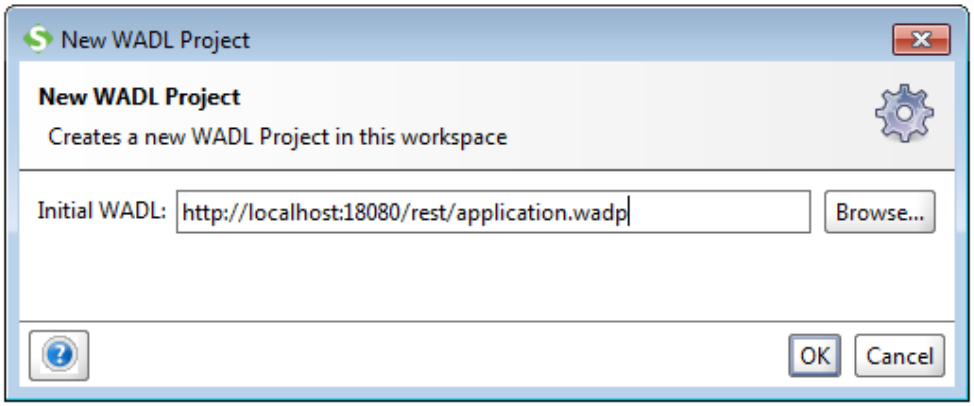
   Select **File | New REST Project** from the main menu:



   Click **Import WADL**



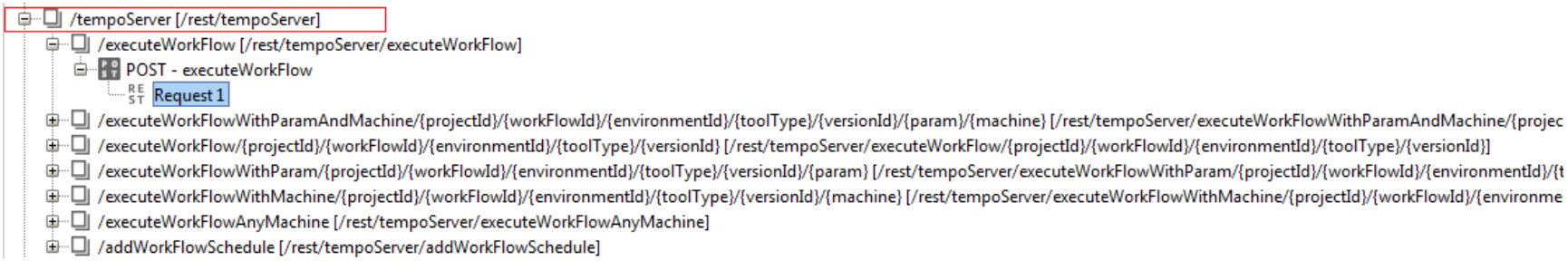
2. Enter "http://localhost:18080/rest/application.wadp" in the Initial WADL field



Click **OK**

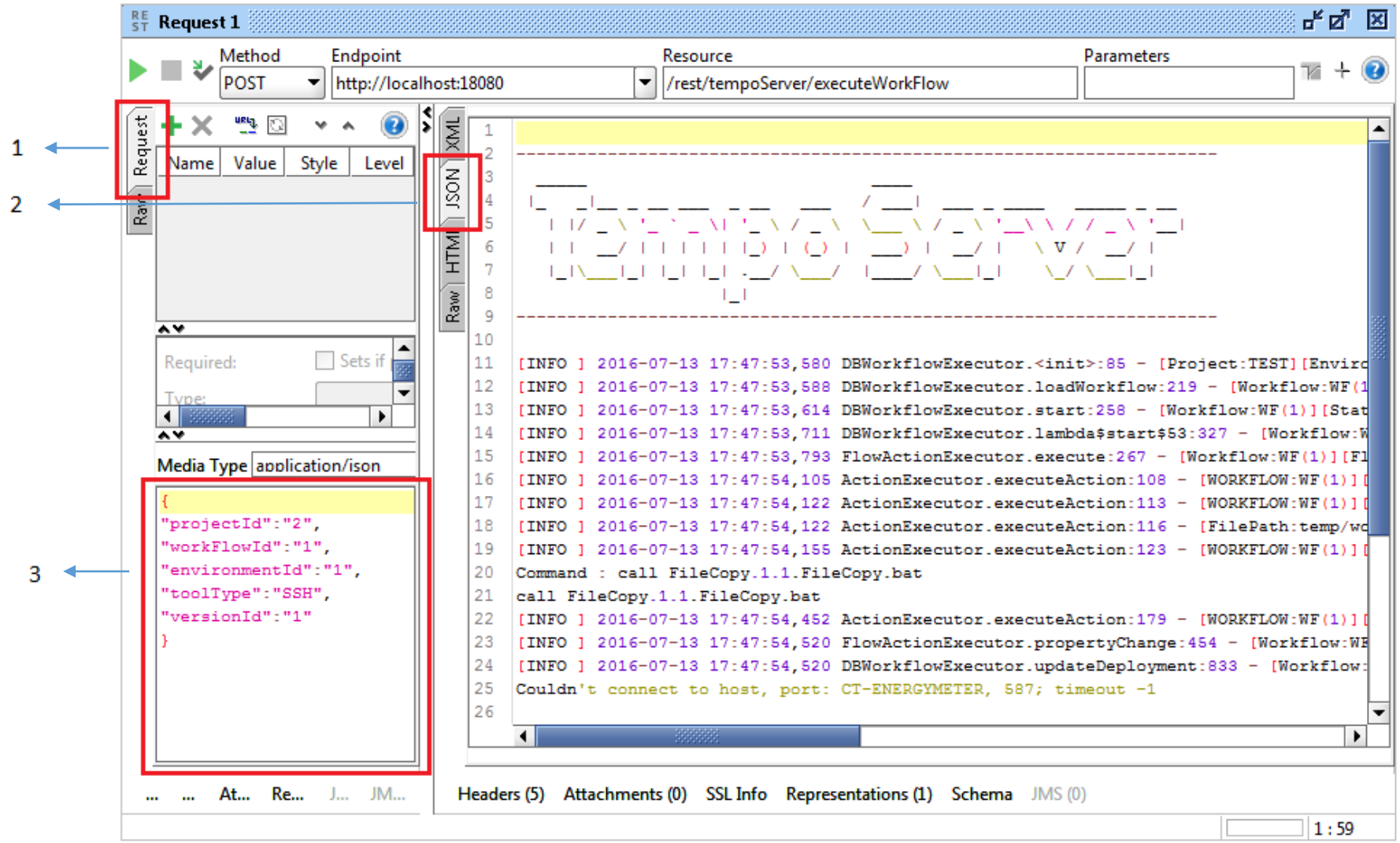
- ✓ *SoapUI creates the specified project and imports the WADL, resulting in the following object hierarchy in the navigator*

3. Select *" /tempoServer [/rest/tempoServer]"* from the imported actions



- ✓ Below /tempoServer it will load several actions and you have to select the correct action according to the requirement
- ✓ Clicking on **Request** will open execution window as below

Execute a workflow through SoapUI using JSON



Click **Request** (1), then it will open another panel below (3)

a. Here (3) you can type JSON as according to what you have chosen below the "*/tempoServer [/rest/tempoServer]"*

POST Service for execute workflow

http://localhost:18080/rest/tempoServer/executeWorkFlow

JSON OBJECT - >

```
{
"projectId":"1",
"workFlowId":"1",
"environmentId":"1",
"toolType":"SSH",
"versionId":"1",
"param":"para1=value1&para2=value2",
"machine":"1,2"
}
```

- ✓ In post method, need to pass JSON object with above format
- ✓ In JSON object, "projectID" ,"workflowID" ,"evnID" ,"VersionID" and "toolType" must pass.
- ✓ "Param" and "machine" are optional.

NOTE: Examples for {param} and {machine} parameters.

- {param} = param1=value1&param2=value2&param3=value3
  (List of param and value pairs separate with & sign.)
- {machine} = 1,2,3,4,5
  (List of machine ids separate with comma)

POST Service for add a work flow schedule

http://localhost:18080/rest/tempoServer/addWorkFlowSchedule

- JSON Object for POST Rest Service :

```
{
"projectId":23,
"workFlowId":28,
"name":"TEST123",
"startTime":"2016-07-06 11:23:43",
"envName":"QA",
"version":"1.0.8.0",
"toolType":"SSH",
"param":"w1=P1&w2=P2",
"machines":"M1,M3",
"job":"Execute_Workflow"
}
```

✓ NOTE: "**param"** is option
✓ POST service output will be json object like following,

```
{  "status": "true",
  "description": "Workflow Schedule is added Successfully"
}
```