

Connectors

Introduction

A connector in the REST architecture style is a software element that manages network communication for a component, typically by implementing a network protocol (e.g. HTTP). A client connector initiates communication with a server (of any kind) by creating a request. A server connector listens for connections (from clients of any kind), transmits the request to the component that performs the request processing, creates the response and sends it to the client.

All connectors are provided as extensions of the Noelios Restlet Engine, the reference implementation of the Restlet API.

This document will describe how to add a connector to your application, how to configure it and will give you the list of available server and client connectors.

Add a connector to your application

All connectors and their dependencies are shipped with the Restlet distribution by the way of jar files. Adding a connector to your application is as simple as adding the archives of the chosen connector and its dependencies to the classpath.

You can also have a look to the [FAQ #4](#)¹ and [FAQ #5](#)² which completes this subject.

Configuration

Each connector looks for its configuration from its context. The latter provides a list of modifiable parameters, which is the right place to set up the connector's configuration. Some parameters are defined by the NRE engine and thus are shared by all clients (in the ClientHelper hierarchy) and server connectors (in the ServerHelper hierarchy), and most of them by the connector's ClientHelper or ServerHelper subclasses.

The list of all parameters are available in the javadocs. Please refer to the rest of this document for references to these documentation. Here are the [commons parameters](#)³ dedicated to internal connectors.

Server connectors

Here are the [commons parameters](#)⁴ dedicated to non-internal HTTP server connectors.

Here are the [commons parameters](#)⁵ dedicated to internal HTTP server connectors.

Here is a sample code showing how to set such a parameter on a component's server connector.

```
// Create the HTTP server and listen on port 8182
Component c = new Component();
Server server = c.getServers().add(Protocol.HTTP, 8182);
server.getContext().getParameters().add("useForwardedForHeader", "true");
c.start();
```

Client connectors

Here are the [commons parameters](#)⁶ dedicated to non-internal HTTP client connectors.

Here are the [commons parameters](#)⁷ dedicated to internal HTTP client connectors.

Here is a sample code showing how to set such a parameter.

```
Client client = new Client(new Context(), Protocol.HTTP);
```

```
client.getContext().getParameters().add("useForwardedForHeader", "false");
```

Here is a sample code showing how to set such a parameter on a component's client connector.

```
// Create the HTTP server and listen on port 8182
Component c = new Component();
Client client = c.getClients().add(Protocol.HTTP);
client.getContext().getParameters().add("useForwardedForHeader", "false");
```

If you want to configure the client connector used by a ClientResource, there are several cases. When your ClientResource instances are created in the context of an application hosted by a Component, the client connector of the component is used for all requests. Thus, just configure the component's client connector as shown just above. If not, just set it:

```
// Instantiate the client connector, and configure it.
Client client = new Client(new Context(), Protocol.HTTP);
client.getContext().getParameters().add("useForwardedForHeader", "false");

// Instantiate the ClientResource, and set it's client connector.
ClientResource cr = new ClientResource("http://www.example.com/");
cr.setNext(client);
```

List of available connectors

Server connectors

Extension	Version	Protocols	Asynchronous	Comment
Internal ⁸	2.0	HTTP, RIAP	Yes	Recommended for development
Grizzly ⁹	1.9	HTTP, HTTPS	No	Experimental integration
Jetty ¹⁰	7.0	HTTP, HTTPS, AJP	No	Recommended for robust and scalable deployments
Netty ¹¹	3.1	HTTP, HTTPS	No	Experimental integration
Simple ¹²	4.1	HTTP, HTTPS	No	Recommended for lightweight deployments
Servlet ¹³	2.5	HTTP, HTTPS, AJP	No	Recommended for deployments inside Java EE servers

Client connectors

Extension	Version	Protocols	Asynchronous	Proxy	Comment
Internal ¹⁴	2.0	HTTP, CLAP, FILE, RIAP	Yes	No	Stable but HTTP connectors are recommended for development only

Apache HTTP Client ¹⁵	4.0	HTTP, HTTPS	No	Yes	Recommended for robust and scalable deployments
Lucene Solr ¹⁶	2.9	SOLR	No	No	Stable
Net ¹⁷ (JDK's HttpURLConnection)	1.5	HTTP, HTTPS, FTP	No	Yes	Recommended for lightweight deployments
JavaMail ¹⁸	1.4	SMTP, SMTPS, POP, POPS	No	No	Stable
JDBC ¹⁹	3.0	JDBC	No	No	Stable

1. <http://www.restlet.org/documentation/1.1/faq#04>
2. <http://www.restlet.org/documentation/1.1/faq#05>
3. <http://www.restlet.org/documentation/2.0/jse/engine/org/restlet/engine/http/connector/BaseHelper.html>
4. <http://www.restlet.org/documentation/2.0/jse/engine/org/restlet/engine/http/HttpServerHelper.html>
5. <http://www.restlet.org/documentation/2.0/jse/engine/org/restlet/engine/http/connector/BaseServerHelper.html>
6. <http://www.restlet.org/documentation/2.0/jse/engine/org/restlet/engine/http/HttpClientHelper.html>
7. <http://www.restlet.org/documentation/2.0/jse/engine/org/restlet/engine/http/connector/BaseClientHelper.html>
8. [/docs_2.0/13-restlet/27-restlet/48-restlet/86-restlet.html](#)
9. [/docs_2.0/13-restlet/28-restlet/73-restlet.html](#)
10. [/docs_2.0/13-restlet/28-restlet/78-restlet.html](#)
11. [/docs_2.0/13-restlet/28-restlet/296-restlet.html](#)
12. [/docs_2.0/13-restlet/28-restlet/82-restlet.html](#)
13. [/docs_2.0/13-restlet/28-restlet/81-restlet.html](#)
14. [/docs_2.0/13-restlet/27-restlet/48-restlet/86-restlet.html](#)
15. [/docs_2.0/13-restlet/28-restlet/75-restlet.html](#)
16. [/docs_2.0/13-restlet/28-restlet/229-restlet.html](#)
17. [/docs_2.0/13-restlet/28-restlet/79-restlet.html](#)
18. [/docs_2.0/13-restlet/28-restlet/76-restlet.html](#)
19. [/docs_2.0/13-restlet/28-restlet/77-restlet.html](#)