

Table of contents

1. [Introduction](#)²
2. [Imported classes](#)³
3. [Declaring the Main class](#)⁴
4. [Main method](#)⁵
5. [Build the component](#)⁶
6. [Redirection application](#)⁷
7. [Conclusion](#)⁸

Introduction

The Restlet web site that you are currently navigating is powered by the Noelios Restlet Engine, the reference implementation of the Restlet API. In order to serve HTTP calls, we rely on a production-ready Simple 3.1 HTTP connector. As you will see below, running basic web sites with Restlets is very simple.

WARNING

This page needs to be updated to use the new Restlet 2.0 API

Imported classes

First, let's declare the imported classes required to support our web component.

```
import org.restlet.Component;
import org.restlet.VirtualHost;
import org.restlet.data.Protocol;
```

Declaring the Main class

Now, we declare the main class, called the WebComponent, extending the org.restlet.Component. This component contains several virtual hosts and associated applications.

```
/**
 * The web component managing the Restlet and Noelios Technologies web
 * servers.
 *
 * @author Jerome Louvel (contact@noelios.com)
 */
public class WebComponent extends Component {
    ...
}
```

Main method

Below, you have the main method that is invoked by our startup scripts. Note that we require a few arguments in order to parameterize several aspects like IP address and port to listen on, or the location of static files.

```
/**
```

```

* Main method.
*
* @param args
*         Program arguments.
*/
public static void main(String[] args) {
    try {
        if ((args == null) || (args.length != 7)) {
            // Display program arguments
            System.err
                .println("Can't launch the web server. List of "
                    + "required arguments:\n"
                    + " 1) IP address to listen on\n"
                    + " 2) Port to listen on\n"
                    + " 3) File URI to the \"www\" directory location.\n"
                    + " 4) File URI to the \"data\" directory location.\n"
                    + " 5) Search redirect URI template."
                    + " 6) Login for protected pages."
                    + " 7) Password for protected pages.");
        } else {
            // Create and start the server
            new WebComponent(args[0], Integer.parseInt(args[1]), args[2],
                args[3], args[4], args[5], args[6]).start();
        }
    } catch (Exception e) {
        System.err
            .println("Can't launch the web server.\nAn unexpected "
                + "exception occurred:");
        e.printStackTrace(System.err);
    }
}

```

Build the component

Now we need to build the component containing our web applications. As we are handling several domain names (www.noelios.com, www.restlet.org, search.restlet.org, etc.) via the same HTTP server connector (with a single IP address and port open), we also need to declare several virtual hosts.

```

/**
* Constructor.
*
* @param ipAddress
*         IP address to listen on.
* @param port
*         Port to listen on.
* @param wwwUri
*         File URI to the "www" directory location.
* @param dataUri
*         File URI to the "data" directory location.
* @param redirectUri
*         The search redirect URI template.
* @param login
*         Login for protected pages.
* @param password
*         Password for protected pages.
*/
public WebComponent(String ipAddress, int port, String wwwUri,
    String dataUri, String redirectUri, String login, String password)
    throws Exception {
    getLogService().setLoggerName("com.noelios.web.WebComponent.www");

    // -----
    // Add the connectors
    // -----
    getServers().add(Protocol.HTTP, ipAddress, port);
    getClients().add(Protocol.FILE);

    // -----
    // www.restlet.org

```

```

// -----
VirtualHost host = new VirtualHost(getContext());
host.setHostDomain("www.restlet.org|81.67.81.67");
host.setHostPort("80|" + Integer.toString(port));
host.attach(new WwwRestletOrg(getContext(), dataUri, wwwUri
    + "/www-restlet-org"));
getHosts().add(host);

// -----
// Redirect to www.restlet.org
// -----
host = new VirtualHost(getContext());
host.setHostDomain("restlet.org|restlet.net|restlet.com|"
    + "www.restlet.net|www.restlet.com");
host.setHostPort("80|" + Integer.toString(port));
host.attach(new RedirectApplication(getContext(),
    "http://www.restlet.org{rr}", true));
getHosts().add(host);

// -----
// wiki.restlet.org
// -----
host = new VirtualHost(getContext());
host.setHostDomain("wiki.restlet.org");
host.setHostPort("80|" + Integer.toString(port));
host.attach("/", new RedirectApplication(getContext(),
    "http://wiki.java.net/bin/view/Javawsxml/Restlet{rr}", false));
getHosts().add(host);

// -----
// search.restlet.org
// -----
host = new VirtualHost(getContext());
host.setHostDomain("search.restlet.org|localhost");
host.setHostPort("80|" + Integer.toString(port));
host.attach(new SearchRestletOrg(getContext(), wwwUri
    + "/search-restlet-org", redirectUri));
getHosts().add(host);

// -----
// www.restlet.net
// -----
host = new VirtualHost(getContext());
host.setHostDomain("www.restlet.net");
host.setHostPort("80|" + Integer.toString(port));
host.attach(new RedirectApplication(getContext(),
    "http://restlet.tigris.org{rr}", false));
host.attach("/fisheye/", new RedirectApplication(getContext(),
    "http://fisheye3.cenqua.com/browse/restlet/{rr}", false));
getHosts().add(host);

// -----
// www.noelios.com
// -----
host = new VirtualHost(getContext());
host.setHostDomain("www.noelios.com");
host.setHostPort("80|" + Integer.toString(port));
host.attach(new WwwNoeliosCom(getContext(), dataUri, wwwUri
    + "/www-noelios-com", login, password));
getHosts().add(host);

// -----
// Redirect to www.noelios.com
// -----
host = new VirtualHost(getContext());
host.setHostDomain("noelios.com|noelios.net|noelios.org|"
    + "www.noelios.net|www.noelios.org");
host.setHostPort("80|" + Integer.toString(port));
host.attach(new RedirectApplication(getContext(),
    "http://www.noelios.com{rr}", true));
getHosts().add(host);
}

```

Redirection Application

In addition to the main WebComponent class, we also rely on four application classes. Let's have a look at the RedirectApplication which is generic and reused several times.

```
/**
 * Application redirecting to a target URI.
 *
 * @author Jerome Louvel (contact@noelios.com)
 */
public class RedirectApplication extends Application {
    /** The target URI template. */
    private String targetUri;

    /** Indicates if the redirection is permanent or temporary. */
    private boolean permanent;

    /**
     * Constructor.
     *
     * @param parentContext
     *         The parent context. Typically the component's context.
     * @param targetUri
     *         The target URI template.
     * @param permanent
     *         Indicates if the redirection is permanent or temporary.
     */
    public RedirectApplication(Context parentContext, String targetUri,
        boolean permanent) {
        super(parentContext);
        this.targetUri = targetUri;
        this.permanent = permanent;
    }

    @Override
    public String getName() {
        return "Redirection application";
    }

    @Override
    public Restlet createRoot() {
        int mode = (this.permanent) ? Redirector.MODE_CLIENT_PERMANENT
            : Redirector.MODE_CLIENT_TEMPORARY;
        return new Redirector(getContext(), this.targetUri, mode);
    }
}
```

Conclusion

In term of coding, that's about all that we use. In addition, we configure standard JDK logging properties in order to write a web log file based on the applications' log services that write to the "com.noelios.web.WebComponent.www" logger. Finally, we also rely on the Java Service Wrapper tool to execute our component as a Linux daemon.

NOTE

Thanks to Michael Mayer for the idea of providing the source code of this web site as a sample application.