

## RESTful Web Service

### What Are RESTful Web Services?

**RESTful web services** are built to work best on the Web. Representational State Transfer (REST) is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability, and modifiability, that enable services to work best on the Web. In the REST architectural style, data and functionality are considered resources and are accessed using **Uniform Resource Identifiers (URIs)**, typically links on the Web. The resources are acted upon by using a set of simple, well-defined operations. The REST architectural style constrains an architecture to a client/server architecture and is designed to use a stateless communication protocol, typically HTTP. In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol.

The following principles encourage RESTful applications to be simple, lightweight, and fast:

**Resource identification through URI:** A RESTful web service exposes a set of resources that identify the targets of the interaction with its clients. Resources are identified by URIs, which provide a global addressing space for resource and service discovery. See [The @PathAnnotation and URI Path Templates](#) for more information.

**Uniform interface:** Resources are manipulated using a fixed set of four create, read, update, delete operations: PUT, GET, POST, and DELETE. PUT creates a new resource, which can be then deleted by using DELETE. GET retrieves the current state of a resource in some representation. POST transfers a new state onto a resource. See [Responding to HTTP Methods and Requests](#) for more information.

**Self-descriptive messages:** Resources are decoupled from their representation so that their content can be accessed in a variety of formats, such as HTML, XML, plain text, PDF, JPEG, JSON, and others. Metadata about the resource is available and used, for example, to control caching, detect transmission errors, negotiate the appropriate representation format, and perform authentication or access control. See [Responding to HTTP Methods and Requests](#) and [Using Entity Providers to Map HTTP Response and Request Entity Bodies](#) for more information.

**Stateful interactions through hyperlinks:** Every interaction with a resource is stateless; that is, request messages are self-contained. Stateful interactions are based on the concept of explicit state transfer. Several techniques exist to exchange state, such as URI rewriting, cookies, and hidden form fields. State can be embedded in response messages to point to valid future states of the interaction. See [Using Entity Providers to Map HTTP Response and Request Entity Bodies](#) and [?Building URIs?](#) in the JAX-RS Overview document for more information.

Reference :<http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html>