RESTful Web Services

{scrollbar}

This article takes you through some of basic concepts about RESTful Web Services.

2listpipe

What is REST?

The acronym REST stands for **REpresentational State Transfer** which basically means that each unique URI is a representation of some object and an action executed upon it.

URI & URL
URI (Uniform Resource Identifier) is used throughout this tutorial. If it makes you feel better, cross it out and use URL instead.

In the Web Services world, REpresentational State Transfer (REST) is a key design idiom that embraces a stateless client-server architecture in which the web services are viewed as resources and can be identified by their URIs. Web service clients that want to use these resources can get a content of using HTTP GET, modify the content using HTTP POST or HTTP UPDATE or delete the resource using HTTP DELETE.

REST - An Architectural Style, Not a Standard
REST is not a standard. You will not see the W3C putting out a REST specification. Because REST is just an architectural style. You can design your web services in that style by understanding it.

RESTful Web Services

Systems which follow REST principles are often referred to as RESTful. RESTful Web Services is one among them and simplifies the development of web services when compared to SOAP Web Services. Today, the **Java API for XML Web Services (JAX-WS)** provides basic support for building and deploying RESTful Web Services. However, the upcoming **Java API for RESTful Web Services (JAX-RS)** specification will provide the full support for RESTful Web Services for the Java platform.

XML Message Level

In RESTful Web Services applications commonly work at XML message level which unlocks full potential of utilizing the web service. Although this has a downside by increasing complexity of client applications, compared to the significant ease of developing services its just a very little bump on the road. JAX-WS provides a generic **Provider** and **Dispatch** interface to access the message payload using low level API.

REST Advantages

- The Web Services are completely stateless
- Light Weight
- No security issues with firewalls
- Human Readable results
- Relative ease and reduction in time to develop services.

Security Issues
As firewalls don't understand the meaning of SOAP messages, they never let those messages pass whereas REST messages don't have this problem because they only use what is specified in HTTP standard.

And REST is definitely the trendy way to develop web services if creating web services could ever be trendy.

World Wide Web

The World Wide Web is the key example of RESTful design. Much of it conforms to the REST principles. REST itself draws much of its design from the current design of World Wide Web to be compatible with it.

Tools to develop REST

This is the area where REST is stumbling by a lack of companies to support development of REST. Since REST services are easy to build, theoretically any HTTP-compliant tool could be used to develop them. In Java, the JAX-WS specification provides adequate support for REST Services to develop and consume.

Who are using REST?

RESTful Web Services isn't some obscure thing that is introduced in this tutorial. It has already got plenty of support from many companies. A few listed are:

- Yahoo! - All Yahoo API's Web Services uses REST including Flickr etc
- Google - **Google Adsense Search** and **Google Maps** are two good examples for RESTful services implemented by Google
- Amazon
- EBay
- del.ico.us

Many other companies admire REST Services and the list will be pretty big if each and every company were listed.

When to use REST Services?
If you want to know when to use REST services and when to use SOAP ones, you can refer to the following link for a better understanding http://java.sun.com/developer/technicalArticles/WebServices/restful/

Future

RESTful Web Services have recently been introduced when compared to SOAP Web Services which are sufficiently well established. In addition, the future holds the possibility of describing RESTful web services for tools to consume, which will further simplify the developer's experience.