ARTICLES

RESTful Web Services Part II: Development and Testing

By: Jeremy Deane, Technical Architect, Collaborative Consulting
Friday, November 7, 2008

In my previous article, RESTful Web Services Part I, I introduced the key concepts behind REST and outlined an approach for designing RESTful web services. This article will cover implementing web services using Restlet and testing those web services using Apache JMeter.

The example web services presented in this article are part of a web application that, along with the source code and tests, can be downloaded here. The web services expose transactions for managing facilities. For example, web services exist for creating, retrieving, updating, and deleting buildings.

The web application consists of two architectural layers. The service layer provides access to a domain model persisted in-memory and the web layer provides an API for clients (or consumers). The Spring Framework wires together the application components via dependency injection and Apache Maven is used to build and package the web application.

The web layer is built using the Restlet Framework. Although several options exist for integrating Restlet into the web layer, most organizations will use the Restlet Servlet Converter. The Converter class allows Restlet to hook into an existing web application already implementing a framework such as Struts or Tapestry.

Within the web.xml file a servlet is defined and mapped to a URI pattern that does not conflict with an existing framework (e.g. not "*.do"). In our case, all requests with /resource/ in the URI are mapped to the servlet. The servlet initializes a secured Router class and delegate all requests to the Converter. The Converter in turn transforms requests into a format that can be processed by Restlet and then forwards them onto the Router.

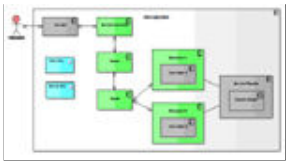

Figure 1: Web Application Deployment Diagram (Click image to view PDF)

During initialization of the Router, resources are mapped to URI patterns. This is similar to how the Struts framework maps URI patterns to actions.

```
router.attach("/building/{buildingID}", BuildingResource.class);
router.attach("/building/{buildingID}/room/{roomID}", RoomResource.class);
```

Within the URI patterns above, templates[1] capture data points that are then available to resources during processing as named value pairs (e.g. {roomID}). For each request the Router maps the URI and then instantiates a Resource class which in turn handles the request. In addition, the Router can be secured using HTTP Basic Authentication. A Guard class intercepts requests to the Router and authenticates the consumer.

A Resource class is essentially a proxy to a conceptual resource. It is responsible for the following:

- returning a representation of the conceptual resource
- accepting a representation that either creates or updates a resource
- deleting a resource

A Representation class captures the state of a resource and is also associated to a Media Type class such as XML, HTML, and JSON.

Within the example web application, the BuildingResource and RoomResource classes both extend an abstract class, FacilitiesResource. The abstract class handles default configuration by overriding methods in the super class, Resource and also provides methods for success and failure response creation. The BuildingResource and RoomResource classes override and implement methods for handling HTTP PUT, GET, POST, and DELETE requests.

Unfortunately, there is no way for a Router class to delegate creation of a Resource class to the

SOA Watch: When Considering Services...

Read this column.

Services are the building blocks of SOA, and like building blocks of a house or a building, the quality will define the value of the finished product. In this case, the SOA itself. Thus, spending...

EXPERTS WANTED

Would you like to:

- Submit an article
- Lead a Round Table
- Speak at a Conference

Contact us today!

Spring Framework. Consequently, for each request, a Resource class must look up the services façade, BuildingServices class using the Spring Framework bean factory. In addition, the Restlet framework requires overriding Resource methods such as allowPut() instead of allowing the Resource to be configured declaratively.

In the example web application, a request for a resource representation is first delegated to the Router which determines, based on the URI, the resource class that should be instantiated. The instantiated resource class attempts to retrieve a resource representation from the services façade in the form of a domain object. If a domain object is returned, the resource class first transforms the representation into an XML document and then returns the XML document within the response. And if a domain object cannot be found, then an error XML document is returned.

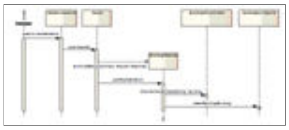

Figure 2: Update Building Sequence Diagram (Click image to view PDF)

A request to create or update a resource is similar to the retrieval process. A consumer submits a POST or PUT of an XML document and the Router, based on the URI, instantiates the appropriate resource class. The resource class transforms the XML document into a domain object and calls the appropriate service layer façade method. A deletion request is simply a pass through from the Router to the resource, to the appropriate service facade method.

Web services tests are created using JMeter's graphical user interface. Because JMeter tests are persisted as XML documents, seeding scripts can dynamically modify the tests before they are run. In addition, JMeter test execution can be automated using a Maven plug-in[2]. Furthermore, Maven projects can be hooked into a continuous integration server such as Continuum or Hudson.

Figure 3: JMeter Web Service Tests (Click image to view PDF)

With slight modifications, JMeter tests can be used for both functional and performance testing. Functional testing ensures that a web service has implemented the functional requirements while performance testing is used to identify bottlenecks and define Service Level Agreements (SLA).

In summary, building and testing RESTful web services is a straight forward process provided you are using a framework supporting the core concepts of REST. The web services referenced in this article were built using Restlet however other frameworks do exist such jersey, the reference implementation for JSR 311 (JAX-RS)[3] . Or you could build your RESTful web services using a resource-oriented computing platform such as NetKernel. Regardless of the implementation technology, a RESTful approach to exposing web services results in interfaces that are simple, scalable and maintainable.

Jeremy Deane is a Technical Architect at Collaborative Consulting and has over 12 years of software engineering experience in leadership positions. His areas of expertise include Resource Oriented Architecture, Performance Engineering and Software Process Improvement.

---

1 URI Template Specification

2 JMeter Maven Plugin

3 The Java API for RESTful Web Services

Back to Articles, including SOA, BPM & BA

BECOME A MEMBER TODAY

SOAInstitute.org offers many benefits to its members. Learn more about becoming a member or join today!

READ MORE ON SOAINSTITUTE.ORG

Featured Presentation:

Making SOA and Enterprise Architecture Efforts More Mutually Supportive

Featuring: Gary Berg-Cross, Principle, Semantic Technology, Engineering, Management and Integration (EM&I)

SOA and EA paradigms compete in setting the agenda to satisfy business goals such as increased business agility. In fact while the 2 paradigms have different perspectives and emphases they also have...

more

Featured White Paper:

SOA and BPM - Taking the Enterprise to the Next Level

Courtesy of: Rick Sweeney, Independent Consultant

It is unfortunate but most companies find themselves always trying to catch up with technology advancements. Like many others they are burdened by a significant investment in legacy systems (and the...

more