W3C Workshop on Data and Services Integration

Position Paper
9 September 2011
Cornelia Davis
Senior Technologist
EMC Corporation

EMC is interested in participating in the upcoming W3C Workshop on Data and Service Integration on 20-21 October in Bedford, Massachusetts. I work in the Architecture Group of the Office of the CTO at EMC and, along with a handful of colleagues, I am responsible for developing architecture and guidance for integration of the myriad of product groups across our company. EMC is a large company made up of business units assembled via numerous acquisitions, our products run on a diverse set of platforms, are built using a varied set of technologies and project many different types of interfaces. When we provide a service to our customers that draws on these varying systems we are challenged to achieve an integration in a manner that is easy to construct, maintain and extend into the future. In short, we have all of the challenges described in the workshop's call for participation in spades, even when we are only addressing internal product integrations. Of course, our customers require further integration with non-EMC technologies as well.

In the CTO Office, we are developing our guidance in a manner that leverages standards and aligns with industry best practices. EMC actively participates in standards development efforts in many forums including the W3C, OASIS, SNIA, IETF and NIST. Our interest in this workshop is in this spirit; we are keen to share our experiences, learn from others and identify further opportunities for collaboration across the industry.

Our Integration Architecture

At the core of our integration architecture are RESTful services and we use the term RESTful SOA to refer to this approach. We chose a RESTful style for several different reasons:

1. Simplicity: We have all had experience with a variety of integration and services paradigms; in fact, several of my colleagues were intimately involved in the development of several of the SOAP-based standards. It comes as no surprise that when standards are too complex, they are difficult to apply correctly and uniformly. If we keep our interfaces and architectures simple, as we can do with REST, our product architects can follow the guidance with greater success.
2. Scale and distribution: Our products are increasingly made available in highly distributed, very large deployments (cloud) and must therefore be constructed to account for the characteristics of such deployments (see [1]). Given that the REST architectural style is that of the World Wide Web, following that approach is wise.

3. Customer request: Our customers are asking for RESTful interfaces, and our products groups are building them. We aim to assist those groups in correctly applying the principals of REST.
4. Resource orientation: EMC is an information management company and making information available as resources is natural.

While we cite simplicity as one of the reasons for taking a REST-oriented approach to integration, and indeed, we find that RESTful interfaces are generally quite easy to work with, our experience shows that creation of good REST interfaces is usually much harder. One of the reasons for this is that the core principals of REST, and the value that they bring, is often not well understood. Those core tenets are:

- Identification and addressability of resources: All interesting bits of information are identified with URIs and are usually accessed via URL.
- The uniform interface: Interaction with resources is through a standardized set of operations, with well understood semantics (usually HTTP).
- Resource representations: Manipulation of resources is through representations. Resources are not objects and representations are not serializations of data structures.
- Hypermedia constraint: Resource representations must carry hyperlinks that allow the application flow to progress. Links may be to other resources or may be to actions that drive the application.

Of these tenets, *resource orientation* and *uniform interface* are often reasonably well applied (though there are a few common pitfalls even there) and frameworks for the development of RESTful services address them. When it comes to the final two principals, however, there are fewer tools to help the developer and hence, mistakes are quite often made.

We have invested significantly in educating EMC technologists in the principals of REST, in part through traditional means such as presentations and project engagements. Additionally, we have provided a framework that provides tooling to address all four tenets of the REST architectural style. Because many data sources are already available in XML form (even stored that way in databases or on disk), our focus has been on XML and we have found that there are limited tools and standards that allow the XML-developer to "do REST." The developer must have a way to create services that fulfill the core REST tenets:

- There must be a way to define the resource model of the RESTful interface.
- There must be a way to define the HTTP interactions for those resources.
- There must be a way to transform native/storage XML format into the RESTful service resource representations.
- There must be a way to make it easy to insert hyperlinks into resource representations.

There are various XML-centric tools such as XProc, XQuery and XSLT that serve the need for the latter two very well, however, we have found no widely adopted XML-centric tools the former two. As such, we have used Java and the Spring Framework. While the use of Java has the disadvantage of forcing the

developer out of the XML tools stack, it does have the advantage that additional concerns, such as security, can be readily addressed.

We are interested in sharing our experiences from the last two years of making REST the core of our integration architecture and prompting a dialog on opportunities for standards or tools development.

References

[1] Fallacies of Distributed Computing, http://en.wikipedia.org/wiki/Fallacies_of_Distributed_Computing.