Not only do we now have the ability to use whatever terms we would like to, we can add new terms and relationships at any point in the future without affecting the existing relationships. This schemaless approach is tremendously appealing to anyone who has ever modified an XML or RDBMS schema. It also represents a data model that not only survives in the face of inevitable social, procedural, and technological changes, but also embraces them.

This RDF would be stored in a triplestore or other database, where it could be queried through SPARQL or a similar language. Most semantically enabled containers support storing and querying RDF in this way now. Examples include the Mulgara Semantic Store,[§] the Sesame Engine,[‖] the Talis Platform,[#] and even Oracle 10g and beyond. Nodes in the graph can be selected based on pattern-matching criteria, so we could ask questions of our resources such as "Who created this URL?", "Show me everything that Brian has created," or "Identify any Creative Commons–licensed material produced in the last six months." The terms that mean "created by," "has license," etc. are expressed in the relevant vocabularies, but are easily translated into our stated goals. The flexibility of the data model coupled with the expressiveness of the query language makes describing, finding, and invoking RESTful services reasonably straightforward. It is certainly more pleasant than trying to find and invoke services through lobotomized and high-impedance technologies such as UDDI.

With the ability to address and resolve arbitrary resources, the ability to retrieve them in different forms and the ability to describe them in Open World and mixed-vocabulary ways, we are now ready to apply these ideas in the Enterprise. We will describe an information-driven architecture that supports "surfing" webs of data like you might "surf" the Web of documents.

Resource-Oriented Architectures

The resource-oriented style is marked by a process of issuing logical requests for named resources. These requests are interpreted by some kind of engine and turned into a physical representation of the resource (e.g., HTML page, XML form, JSON object, etc.). See Figure 5-4.

The basic interaction style in a resource-oriented architecture (ROA) is demonstrated in this figure. A logical request is named, resolved, and transferred back to the requestor in some form by a resource-oriented engine. The named resource is likely to resolve to a database query or some bit of functionality that manages information (e.g., a RESTful service). What responds to the request, which is largely irrelevant to the person interested in the information, is potentially a servlet, a Restlet,[*] a NetKernel module, or some other bit of addressable

[§] *http://mulgara.org*

[‖] *http://openrdf.org*

[#] *http://talis.com*

[*] *http://restlet.org*