Understanding HATEOAS

HATEOAS (Hypermedia as the Engine of Application State) is a constraint of the REST application architecture.

A hypermedia-driven site provides information to navigate the site's REST interfaces dynamically by including hypermedia links with the responses. This capability differs from that of SOA-based systems and WSDL-driven interfaces. With SOA, servers and clients usually must access a fixed specification that might be staged somewhere else on the website, on another website, or perhaps distributed by email.

Note: Pronunciations of HATEOAS vary. Some people pronounce it as "hate-ee-os," similar to "hideous," or as "hate O-A-S". People also refer to it as a hypermedia-driven system.

Examples

The following code represents a `Customer` object.

```
class Customer {
    String name;
}
```

A simple JSON presentation is traditionally rendered as:

```
{
    "name" : "Alice"
}
```

The customer data is there, but the data contains nothing about its relevant links.

A HATEOAS-based response would look like this:

```
{
    "name": "Alice",
    "links": [ {
        "rel": "self",
        "href": "http://localhost:8080/customer/
    } ]
}
```

This response not only has the person's name, but includes the self-linking URL where that person is located.

- **rel** means relationship. In this case, it's a self-referencing hyperlink. More complex systems might include other relationships. For example, an order might have a `"rel":"customer"` relationship, linking the order to its customer.
- **href** is a complete URL that uniquely defines the resource.

Note: Although these responses are shown in JSON, XML is also accepted as a standard response format. HATEOAS doesn't impose the requirement of either format. The hypermedia links are the focus of HATEOAS.

It's possible to build more complex relationships. With HATEOAS, the output makes it easy to glean how to interact with the service without looking up a specification or other external document.

Look at the following catalog, courtesy of the sample application shown in the Spring Data Book:

```
{
    "content": [ {
        "price": 499.00,
        "description": "Apple tablet device",
        "name": "iPad",
        "links": [ {
            "rel": "self",
            "href": "http://localhost:8080/produ
        } ],
        "attributes": {
            "connector": "socket"
        }
    }, {
        "price": 49.00,
        "description": "Dock for iPhone/iPad",
        "name": "Dock",
        "links": [ {
            "rel": "self",
            "href": "http://localhost:8080/produ
        } ],
        "attributes": {
            "connector": "plug"
        }
    } ],
    "links": [ {
        "rel": "product.search",
        "href": "http://localhost:8080/product/s
    } ]
}
```

Not only are the items and their prices shown, but the URL for each resource is shown, providing clear information on how to access these resources. According to the Richardson Maturity Model, HATEOAS is considered the final level of REST. This means that each link is presumed to implement the standard REST verbs of GET, POST, PUT, and DELETE (or a subset). Thus providing the links as shown above gives the client the information they need to navigate the service.