WIKIPEDIA

HATEOAS

Hypermedia As The Engine Of Application State (**HATEOAS**) is a constraint of the REST application architecture that distinguishes it from other network application architectures.

With HATEOAS, a client interacts with a network application that application servers provide dynamically entirely through hypermedia. A REST client needs no prior knowledge about how to interact with an application or server beyond a generic understanding of hypermedia.

By contrast, clients and servers in some service-oriented architectures (SOA) interact through a fixed interface shared through documentation or an interface description language (IDL).

The way that the HATEOAS constraint decouples client and server enables the server functionality to evolve independently.

Contents

Details

A REST client enters a REST application through a simple fixed URL. All future actions the client may take are discovered within resource representations returned from the server. The media types used for these representations, and the link relations they may contain, are standardized. The client transitions through application states by selecting from the links within a representation or by manipulating the representation in other ways afforded by its media type. In this way, RESTful interaction is driven by hypermedia, rather than out-of-band information.[1]

For example, [2] this GET request fetches an account resource, requesting details in an XML representation:

```
GET /accounts/12345 HTTP/1.1
Host: bank.example.com
Accept: application/xml
...
```

The response is:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: ...

<?xml version="1.0"?>
<account>
```

```
    <account_number>12345</account_number>
    <balance currency="usd">100.00</balance>
    <link rel="deposit" href="https://bank.example.com/accounts/12345?deposit=12345" />
    <link rel="withdraw" href="https://bank.example.com/accounts/12345?withdraw=12345" />
    <link rel="transfer" href="https://bank.example.com/accounts/12345?transfer=2345" />
    <link rel="close" href="https://bank.example.com/accounts/12345?status=close" />
</account>
```

The response contains these possible follow-up links: make a deposit, withdrawal, or transfer, or close the account.

When the account information is retrieved later, the account is overdrawn:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: ...

<?xml version="1.0"?>
<account>
    <account_number>12345</account_number>
    <balance currency="usd">-25.00</balance>
    <link rel="deposit" href="https://bank.example.com/accounts/12345?deposit=12345" />
</account>
```

Now only one link is available: to deposit more money. In its current *state*, the other links are not available. Hence the term *Engine of Application State*. What actions are possible vary as the state of the resource varies.

A client does not need to understand every media type and communication mechanism offered by the server. The ability to understand new media types can be acquired at run-time through "code-on-demand" provided to the client by the server.[3]

Origins

The HATEOAS constraint is an essential part of the "uniform interface" feature of REST, as defined in Roy Fielding's doctoral dissertation.[3] Fielding has further described the concept on his blog.[1]

The purpose of some of the strictness of this and other REST constraints, Fielding explains, is "software design on the scale of decades: every detail is intended to promote software longevity and independent evolution. Many of the constraints are directly opposed to short-term efficiency. Unfortunately, people are fairly good at short-term design, and usually awful at long-term design".[1]

Implementations

- Spring HATEOAS,[4] part of the Spring Framework
- Yii 2 Framework REST API supports HATEOAS,[5] part of the Yii Framework (since version 2.0)
- Jersey API supports HATEOAS[6]
- Tastypie supports HATEOAS[7]
- Eve supports HATEOAS[8]
- Apigility, API builder based on Zend Framework 2 (http://framework.zend.com), supports HATEOAS[9]
- Hateoas PHP library (http://hateoas-php.org), supports HATEOAS REST Web Services implementations.
- API Platform (https://api-platform.com/), PHP framework based on hypermedia and Linked Data support with JSON-LD, Schema.org and Hydra
- AtomGraph Processor (https://github.com/AtomGraph/Processor), a Java backend for building declarative, read-write Linked Data applications, supports HATEOAS Linked Data[10]
- Symfony Framework supports HATEOAS with willdurand/Hateoas (https://github.com/willdurand/Hateoas)