The Atom Format

In this lesson, we'll look at the Atom format.

WE'LL COVER THE FOLLOWING

Introduction

Atom is a data format originally developed to **make blogs accessible to readers**. An Atom document contains blog posts that subscribers can read in chronological order. Besides blogs, Atom can also be used for **podcasts**.

Because the protocol is so flexible, it is **also suitable for other types of data**.

- For example, a microservice can offer information about **new orders** to other microservices as an **Atom feed**. This corresponds to the REST approach, which uses established web protocols such as HTTP for the integration of applications.

> Atom is not a protocol, but **a data format**.

A GET request to a new URL such as http://innoq.com/order/feed can return a document with orders in the Atom format. This document can contain links to the details of the orders.

MIME T

MIME Type

HTTP-based communication indicates the type of content with the help of **MIME types (Multipurpose Internet Mail Extensions)**.

> The MIME type for Atom is `application/atom+xml` .

Thanks to content negotiation, other data representation can be offered in addition to Atom for the same URL.

Content negotiation is built into HTTP. The HTTP client signals via an accept header what data formats it can process. The server then sends a response in a suitable format. Thus, accept headers enable a client to request all orders as JSON or the last changes as an Atom feed under the same URL.

Feed

```
<?xml version="1.0" encoding="UTF 8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title>Order</title>
  <link rel="self" href="http://localhost:8080/feed" />
  <author>
    <name>Big Money Online Commerce Inc.</name>
  </author>
  <subtitle>List of all orders</subtitle>
  <id>tag:ewolff.com/microservice-atom/order</id>
  <updated>2017 04 20T15:28:50Z</updated>
  <entry>
  ...
  </entry>
</feed>
```

> A document in Atom format is called an **Atom feed**.

An Atom feed contains different components.

Meta data

First, the feed defines **meta data**. The Atom standard defines that this meta data has to comprise a number of elements.

- `id` is a globally unambiguous and a permanent URI (Uniform Resource

Identifier). It has to identify the feed. In the listing, this is done via a tag URI.

- `title` is the title of a feed in a human-readable form.
- `updated` contains the point in time when a feed has been last changed. This information is important for finding out whether new data exists.
- There also has to be an `author` including `name`, `email`, and `uri`. This is very helpful for blogs.
    - However, it **does not seem to make sense for Atom feeds**, which only transport data. However, it can be useful for indicating a contact person in case of problems.
- Multiple `link` elements are recommended.
    - Each element has an attribute called `rel` for indicating the relationship between the feed and the link.
    - The attribute `href` contains the actual link.
    - A `link` can be used to provide a link to a HTML representation of the data.
    - In addition, the feed can use a `self` link to indicate at which URL it is available.

Optional meta data

Additional optional meta data includes:

- `category` narrows down the topic area of the feed, for example, to a field such as sports.
- Analogous to `author`, `contributor` contains information about people who contribute to the feed.
- `generator` indicates the software that produced the feed.
- `icon` is a small icon, whereas `logo` represents a larger icon. This makes it possible to represent a blog or podcast on the desktop. It is not that useful for a data feed.
- `rights` define things, for example, the copyright. This element is likewise

mostly interesting for blogs or podcasts.

- Finally, `subtitle` is a human-readable description of the feed.

As the listing illustrates, the fields `id`, `title`, and `updated` are enough for an Atom feed with data. Having a `subtitle` for documentation is helpful. There should also be a `link` for documenting the URL of the feed. All other elements are not really needed.

Entry

In the previous example, the most important thing is missing, the feed entries. Such an entry adheres to the following format:

```
<entry>
  <title>Order 1</title>
  <id>tag:ewolff.com/microservice-atom/order/1</id>
  <updated>2017 04 20T15:27:58Z</updated>
  <content type="application/json"
    src="http://localhost:8080/order/1" />
  <summary>This is the order 1</summary>
</entry>
```

An entry contained in the feed must consist of the following data according to the Atom standard:

- `id` is the globally unambiguous ID as URI. Thus, there cannot be a second entry with the id `tag:ewolff.com/microservice-atom/order/1` in this feed. This URI is a tag URI meant for globally unambiguous identifiers.

- `title` is a human-readable title for the entry.

- `updated` is the timepoint when the entry has last been changed. It has to be set so that the client can decide whether it already knows the last state of a certain entry.

In addition, the following elements are recommended:

- As described for the feed, `author` names the author of the entry.

- `link` can contain links, for example, *alternate* as a link to the actual entry.

- `summary` is a summary of the content. It has to be provided if the content

- `summary` is a summary of the content. It has to be provided if the content is only accessible via a link or if it does not have an XML media type. This is the case in the example.
- `content` can be the complete content of the entries or a link to the actual content. To enable access to the data of the entry, either `content` has to be offered, or a `link` with `alternate`.
- In addition, `category` and `contributor` are optional, analogous to the respective elements of the feed. `published` can indicate the first date of publication, the element rights can state the rights, and source can name the original source if the entry was a copy from another feed.

In the example, the elements `id`, `title`, `summary`, and `updated` are filled. Access to the data is possible via a link in `content`. The data could also directly be contained in the `content` element in the document. However, in that case, the document would be very large.

Thanks to the links the document remains small. Each client has to access the additional data via links, and can limit itself to only the relevant data for a respective client.

QUIZ

1 Atom is ______.

COMPLETED 0%

1 of 3

In the next lesson, we'll look at Atom caching.