
RESTful Hypermedia APIs
Kai Tödter

Who am I?

- Principal Key Expert
  at Siemens Building Technologies
- Web Technology Fan
- Open Source Lover
- E-mail: kai@toedter.com
- Twitter: twitter.com/kaitoedter
- Blog: toedter.com/blog

Show Hands!

Outline

- REST Basics
- HATEOAS
- Hypermedia Examples
- API Documentation
- Demos & Live Coding
- Conclusion

What is REST?

- Stands for Representational State Transfer
- Is a Software Architecture Style
- was introduced and defined in 2000 by Roy T. Fielding in his doctoral dissertation

- REST != CRUD via HTTP

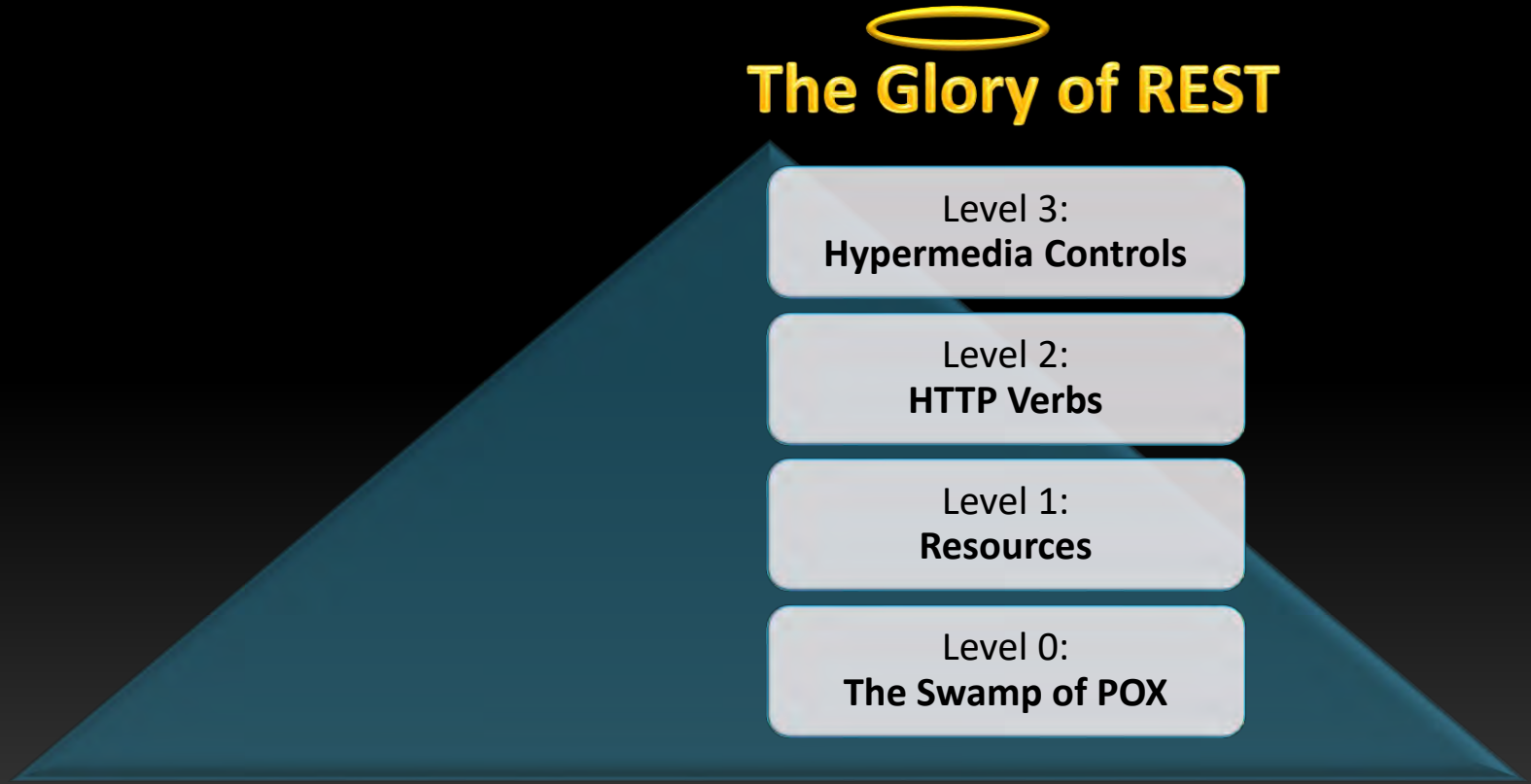
REST Architectural Constraints

- Client-Server
- Stateless
- Cacheable
- Layered system
- Code on demand (optional)
- Uniform interface (see next slide)

Uniform Interface

- Identification of resources
- Manipulation of resources through their representations
  - Create => HTTP POST
  - Read => HTTP GET
  - Update => HTTP PUT, HTTP PATCH
  - Delete => HTTP DELETE
- Self-descriptive messages
- Hypermedia as the engine of application state (HATEOAS)

Richardson Maturity Model



See http://martinfowler.com/articles/richardsonMaturityModel.html

Hypermedia APIs

for Services

are like

Web Pages with Links

for Humans

Existing Hypermedia Representations

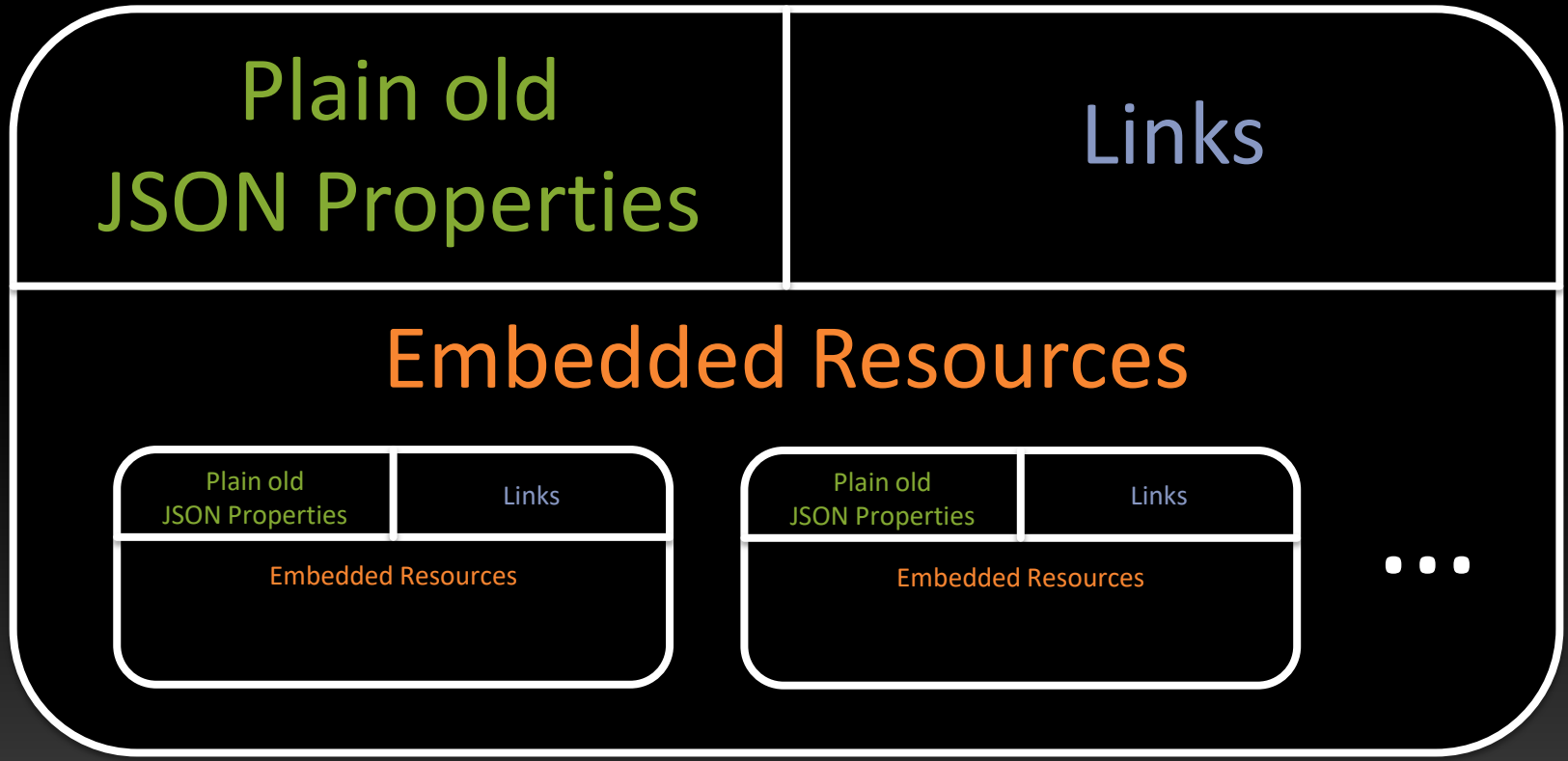
Hypermedia Representations

- HAL - http://stateless.co/hal_specification.html
- Siren - https://github.com/kevinswiber/siren
- Collection+JSON - http://amundsen.com/media-types/collection/format/
- UBER - https://rawgit.com/mamund/media-types/master/uber-hypermedia.html
- ALPS - http://alps.io/
- Hydra - http://www.markus-lanthaler.com/hydra/
- JSON-LD - http://json-ld.org/
- json:api - http://jsonapi.org/
- Mason - https://github.com/JornWildt/Mason

HAL

- Is for Hypertext Application Language
- Was created by Mike Kelly
- Representations for both JSON and XML
- Very popular

HAL Structure



HAL Example

```
{
   "id":1,
   "text":"hello all!",
   "_links": {
      "self": {
         "href":"http://localhost:8080/chatty/api/messages/1"
       }
   },
   "_embedded": {
      "author": {
          "id":"toedter_k"
       }
    }
}
```

Siren

- Developed by Kevin Swiber
- Similar to HAL
- Provides more mechanisms
  - E.g. Actions
- A bit more complex

Siren Example

```
{
   "class": [ "message" ],
   "properties": {
        "id": 1,
        "text": "hello all!",
   },
   "entities": [
      {
         "class": [ "author" ],
         "properties": {
               "id": "toedter_k"
         }
      }
    ],
    "links": [
        { "rel": [ "self" ], "href": "http://localhost:8080/chatty/api/messages/1" },
     ]
}
```

Siren Action Example

```
"actions": [
    {
       "name": "add-item",
       "title": "Add Item",
       "method": "POST",
       "href": "http://api.x.io/orders/42/items",
       "type": "application/x-www-form-urlencoded",
       "fields": [
          { "name": "orderNumber", "type": "hidden", "value": "42" },
          { "name": "productCode", "type": "text" },
          { "name": "quantity", "type": "number" }
       ]
    }
]
```

Java Implementations

Java Implementations

- HAL
  - HalBuilder
  - Spring HATEOAS
  - halarious
  - HyperExpress
  - hate
  - SlimPay HAPI client
  - Katharsis
  - …
- Siren
  - Siren4j
  - HalBuilder (Siren Extension)
  - SirenJeeni
  - …

siren4j

```
Link selfLink = LinkBuilder.newInstance()
        .setRelationship(Link.RELATIONSHIP_SELF)
        .setHref("/self/link")
        .build();

Entity result = EntityBuilder.newInstance()
        .setEntityClass("test")
        .addProperty("foo", "hello")
        .addProperty("number", 1)
        .addLink(selfLink)
        .build();
```

siren4j Annotations

```
@Siren4JEntity(name = "video", uri = "/videos/{id}")
public class VideoResource extends BaseResource {
    private String id;
    private String name;
    ...
```

HalBuilder

```
@GET
@Produces(RepresentationFactory.HAL_JSON)
public String getApi(@Context UriInfo uriInfo) {

   Representation representation = representationFactory
         .newRepresentation()
         .withNamespace("chatty", "http://docu.chatty.com/{rel}");
         .withLink("chatty:users",
               createUriFromResource(baseURI, UserResource.class))

   return representation.toString(RepresentationFactory.HAL_JSON);
}
```

JSON Result

```
{
   _links: {
       curies: {
           href: "http://docu.chatty.com/{rel}",
           name: "chatty",
           templated: true
       },
       chatty:users:  {
           href: "http://localhost:8080/chatty/api/users"
        }
   }
}
```

Spring

- Spring Boot
- Spring Data REST
- Spring HATEOAS

Domain Classes with Project Lombok

```
@Data
@Entity
@NoArgsConstructor
public class User {
   @Id
   private String id;
   private String fullName;
   private String email;

   @OneToMany(mappedBy="author", cascade= CascadeType.ALL)
   List<ChatMessage> messages;
}
```

Spring Data REST: Repository

```
@RepositoryRestResource(
        collectionResourceRel = "users",
        path = "users")

interface UserRepository extends
         PagingAndSortingRepository<User, String> {
}
```

Spring Data REST: Repository (2)

```
@RepositoryRestResource( exported = false )

interface UserRepository extends
        PagingAndSortingRepository<User, String> {
}
```

Spring Data Rest: JSON Result


```
{
   _links: {
      self: {
         href: "http://localhost:8080/chatty/api/users{?page,size,sort}",
         templated: true
      }
   },
   _embedded: {
       users: [ {
         fullName: "Jane Doe",
         email: "jane@doe.com",
         _links:  {
            self: {
                href: "http://localhost:8080/chatty/api/users/doe_ja"
            },
…
```

JavaScript Implementations

- HAL
  - backbone.hal
  - gomoob/backbone.hateoas
  - RePoChO/backbone.hateoas
  - halbert
  - halberd
  - JS HAL
  - hateoas-client
  - hyperagent.js
  - Traverson
  - Ember.js Data HAL Adapter
  - HALSON
  - hal-body
  - koa-hal
  - angular-hal
  - angular-hy-res
  - halacious for node/hapi
  - rest.js (from the cujo toolkit)
  - angular-hypermedia
- Siren
  - Node-siren
  - Backbone.Siren
  - AngularHypermedia
  - angular-hy-res
  - …

Robust Clients

- Start from main API
- Find link relations through defined contracts
- Follow Links
  - For navigation
  - For possible “actions”

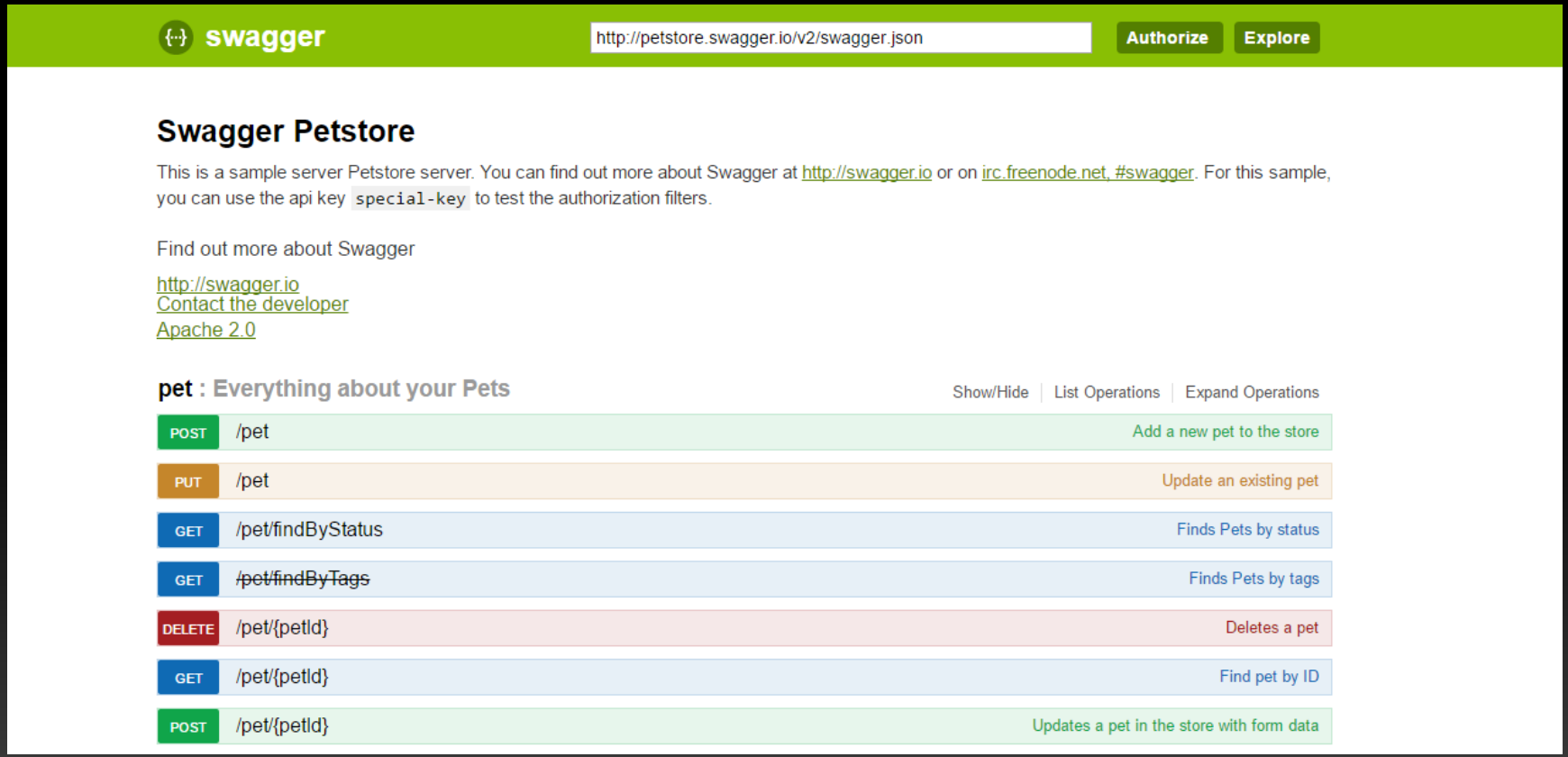
=> Clients are robust regarding changes in link URIs

API Documentation

API Documentation Tools

- Swagger (OpenAPI)
- Spring REST Docs
- RAML (RESTful API Modeling Language)
- …

Swagger



Swagger + Hypermedia

- Pros
  - High Adoption
  - Good tooling
  - Active community
- Cons
  - It’s URI-Centric
  - Not suitable for hypermedia and links
    - OpenAPI v3 introduced links, but they are not sufficient for Hypermedia doc
  - It’s has a pretty big footprint (many libs)

⇒ Not recommended for the documentation of Level 3 REST APIs

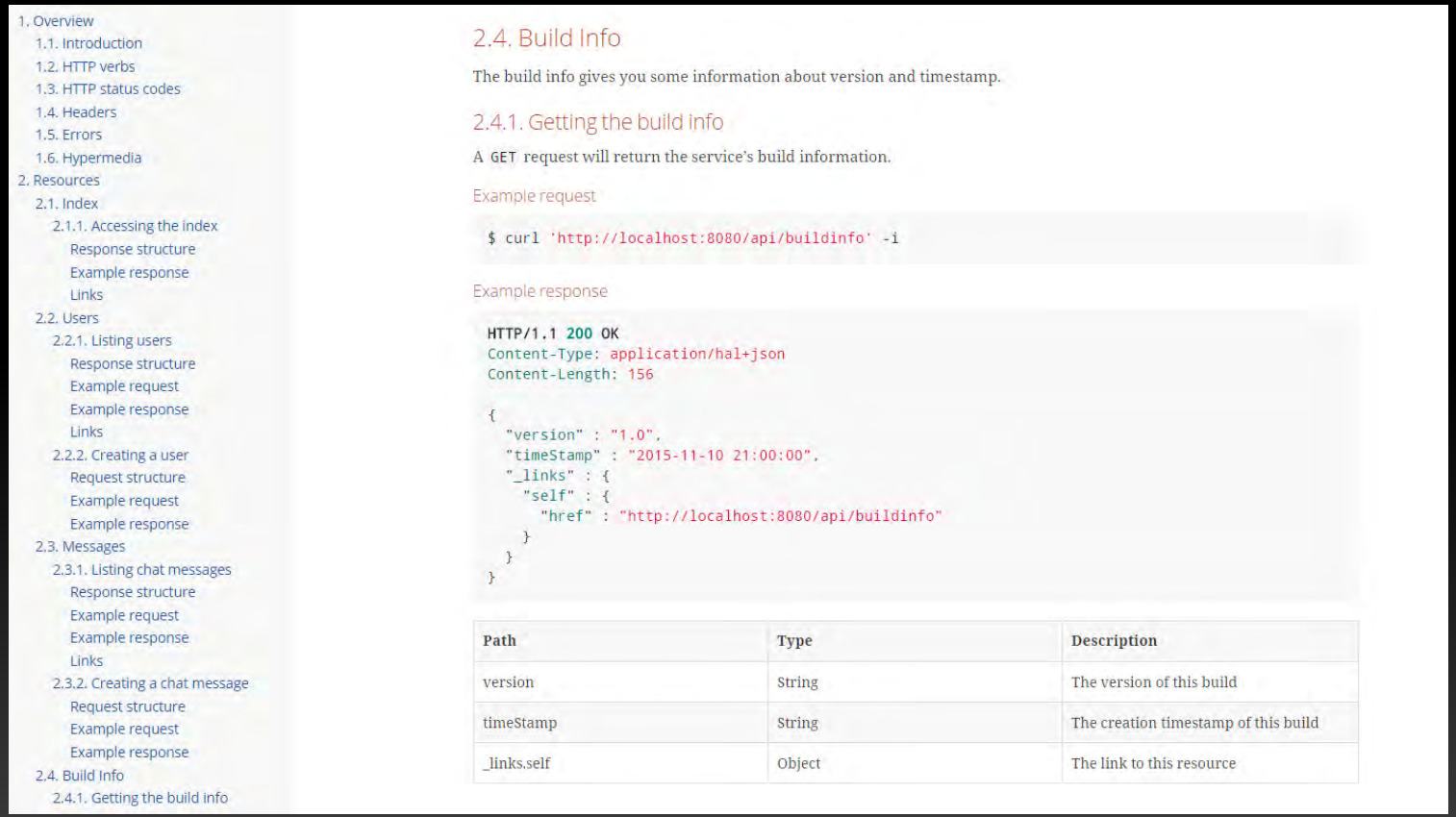
Spring REST Docs

- Write Tests for the API documentation
  - Tests will fail if real behavior and documented behavior are out of sync!
- Include generated AsciiDoc snippets in manually written AsciiDoc API documentation
- Supports Hypermedia well

Spring REST Docs Example Test

```
@Test
public void shouldDocumentBuildInfo() throws Exception {
   this.mockMvc.perform(get("/api/buildinfo"))
         .andExpect(status().isOk())
         .andDo(document("build-info-get-example",
               responseFields(
                     fieldWithPath("version").description("The version of this build"),
                     fieldWithPath("timeStamp").description("The creation timestamp"),
                     fieldWithPath("_links.self").description("The link to this resource")
               )));
}
```

Spring REST Docs Example Result



2.4. Build Info

The build info gives you some information about version and timestamp.

2.4.1. Getting the build info

A `GET` request will return the service's build information.

Example request

```
$ curl 'http://localhost:8080/api/buildinfo' -i
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 156

{
  "version" : "1.0",
  "timeStamp" : "2015-11-10 21:00:00",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/api/buildinfo"
    }
  }
}
```

Path	Type	Description
version	String	The version of this build
timeStamp	String	The creation timestamp of this build
_links.self	Object	The link to this resource

Live Demos

Sources: https://github.com/toedter/chatty

Demo: https://chatty42.herokuapp.com

HAL Browser: https://chatty42.herokuapp.com/api/

Controversial Discussion

- Are we there yet?
- RESTistential Crises
  - http://www.infoq.com/news/2014/03/rest-at-odds-with-web-apis
- DHH, Getting hyper about hypermedia apis
  - https://signalvnoise.com/posts/3373-getting-hyper-about-hypermedia-apis

Conclusion

RESTful Web Services

\+ Hypermedia

Works very well together

Discussion

Links

- siren4j: https://code.google.com/p/siren4j/
- HalBuilder: https://github.com/HalBuilder
- Spring Data REST: http://projects.spring.io/spring-data-rest/
- Spring REST Docs: http://projects.spring.io/spring-restdocs
- Project Lombok: http://projectlombok.org/

License

- This work is licensed under a Creative Commons Attribution 4.0 International License.
  - See http://creativecommons.org/licenses/by/4.0/