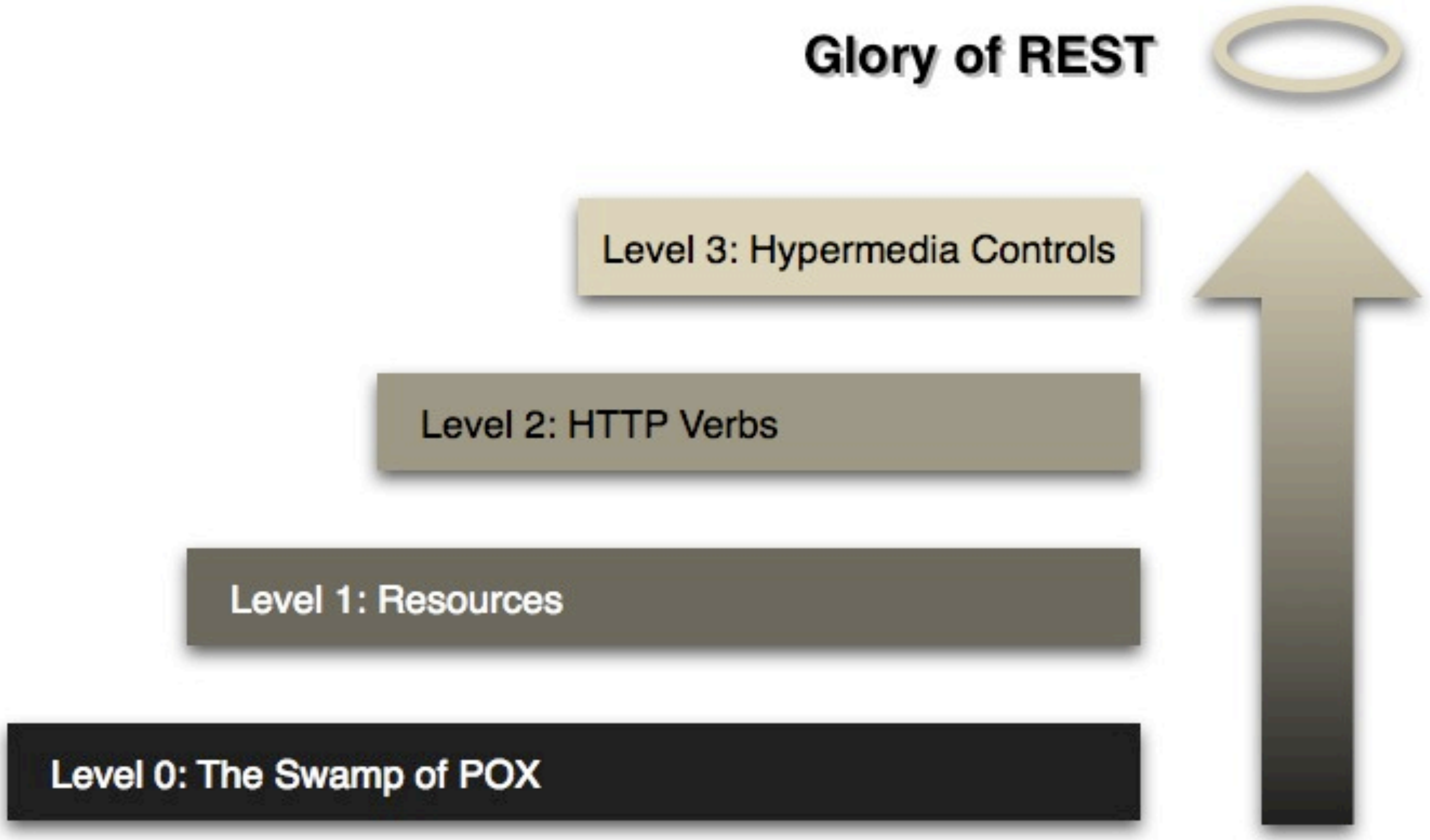
RESTful Web Services

With Spring MVC

@DigitalSonic

~~What?~~ Why? How?

Richardson Maturity Model



Agenda

- How to identify Resources
- How to use HTTP Verbs
- How to design Representations
- Miscellaneous
    - Cache / Version / HATEOAS

Resources

- Domain Nouns
- Collections & Composites
- Functions
- Controllers

Samples

- Article & Comment
  - `http://domain/articles`
  - `http://domain/articles/1`
  - `http://domain/articles/1/comments`
- Web Page
  - a composite of article and comments
- Magazine
  - a collection of articles

Samples

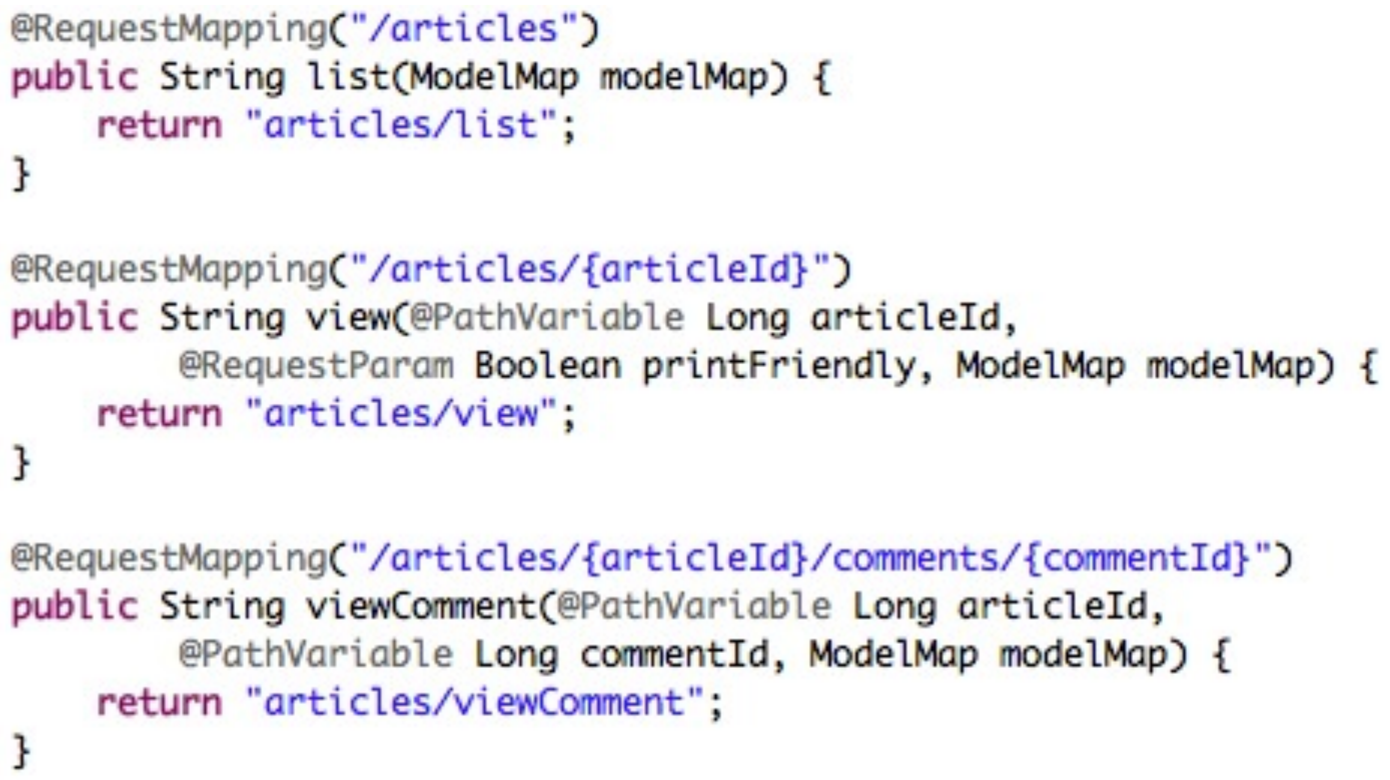
- Functions
  - `http://domain/direction?from=shanghai&to=hangzhou`
- Controllers
  - `http://domain/users/smith/address_merge`

Resource Granularity

- Network Efficiency
- Size of Representations
- Client Convenience

URL Mapping in Spring MVC

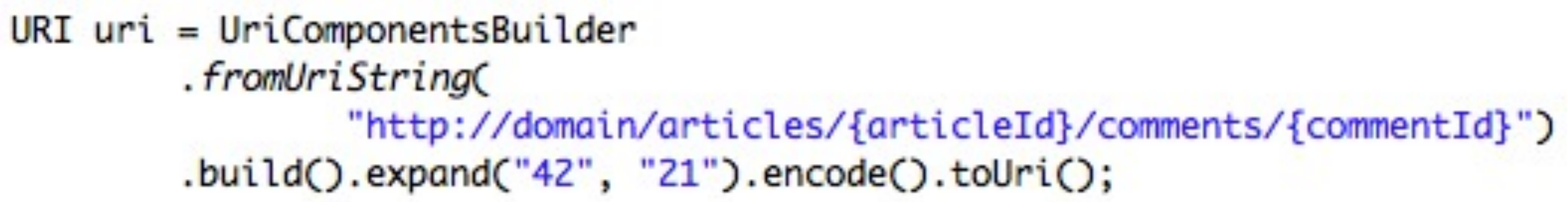
- `<mvc:annotation-driven/>`
- `@RequestMapping`
- `@PathVariable`
- `@RequestParam`
- `@MatrixVariable`



```java
@RequestMapping("/articles")
public String list(ModelMap modelMap) {
    return "articles/list";
}

@RequestMapping("/articles/{articleId}")
public String view(@PathVariable Long articleId,
        @RequestParam Boolean printFriendly, ModelMap modelMap) {
    return "articles/view";
}

@RequestMapping("/articles/{articleId}/comments/{commentId}")
public String viewComment(@PathVariable Long articleId,
        @PathVariable Long commentId, ModelMap modelMap) {
    return "articles/viewComment";
}
```

Building URI

- `UriComponents`
- `UriComponentsBuilder`
- `ServletUriComponentsBuilder`



```
URI uri = UriComponentsBuilder
        .fromUriString(
                "http://domain/articles/{articleId}/comments/{commentId}")
        .build().expand("42", "21").encode().toUri();
```

Better URI

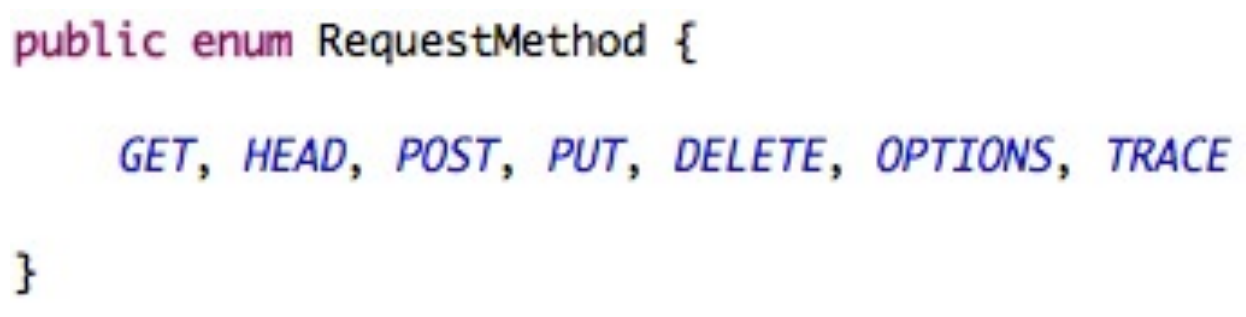
- use domains and subdomains to logically group or partition resources
- use / to indicate a hierarchical relationship
- use , and ; to indicate nonhierarchical elements
- use - and _ to improve the readability
- use & to separate parameters
- avoid including file extensions

HTTP Verbs

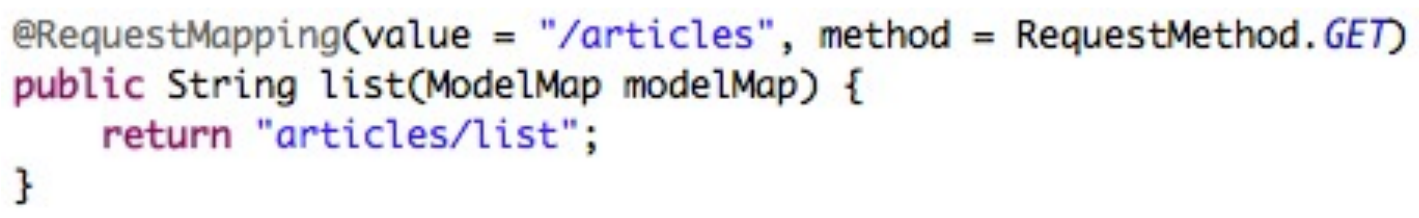
Verbs	Safety & Idempotency	Usage
GET	Y / Y	Use GET for safe and idempotent information retrieval.
POST	N / N	Use this method to let the resource perform a variety of actions on the server side such as creating new resources, updating existing resources, or making a mixture of changes to one or more resources.
DELETE	N / Y	Use this method to let a client delete a resource.
PUT	N / Y	Use this method to completely update or replace an existing resource or to create a new resource with a URI specified by the client.
HEAD	Y / Y	Use this method to retrieve the same headers as that of a GET response but without any body in the response.
OPTIONS	Y / Y	Use this method to find the list of HTTP methods supported by any resource or to ping the server.
TRACE	Y / Y	Use this method to let the server echo back the headers that it received.

HTTP Method in Spring MVC

- `@RequestMapping`
- `RequestMethod` Enum



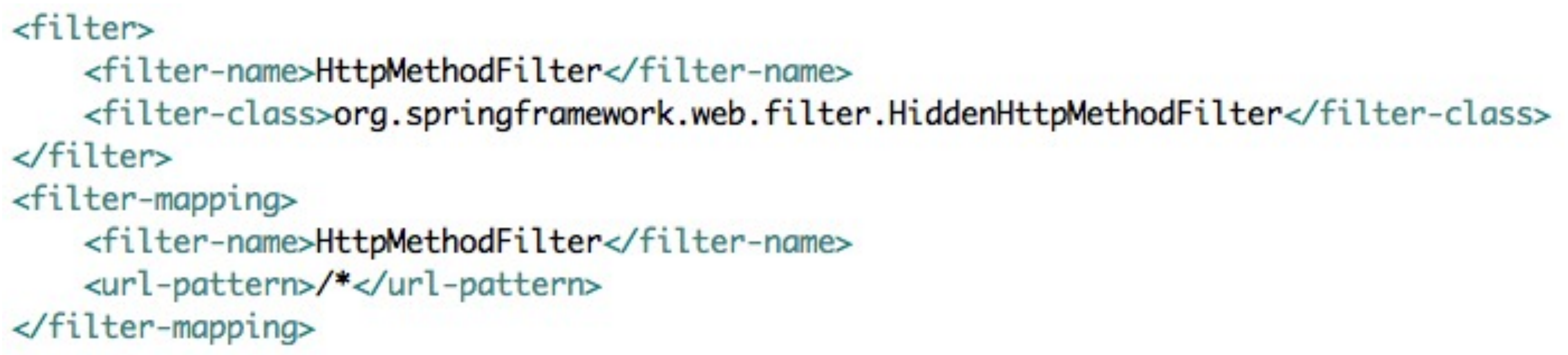
```java
public enum RequestMethod {

    GET, HEAD, POST, PUT, DELETE, OPTIONS, TRACE

}
```



```java
@RequestMapping(value = "/articles", method = RequestMethod.GET)
public String list(ModelMap modelMap) {
    return "articles/list";
}
```

Your web form doesn’t support so much verbs?

```
HiddenHttpMethodFilter
<input type="hidden" name="_method" value="put"/>
```



```
<filter>
    <filter-name>HttpMethodFilter</filter-name>
    <filter-class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>HttpMethodFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

Representations

- Different view technologies in Spring
  - JSP/JSTL (`InternalResourceViewResolver`)
  - Tiles (`TilesViewResolver`)
  - XML Marshalling (`MarhsallingView`)
  - JSON Mapping (`MappingJacksonJsonView`)
  - etc.
- `@ResponseBody`

HTML

Should I say anything about this?

You know what I mean.

JSON

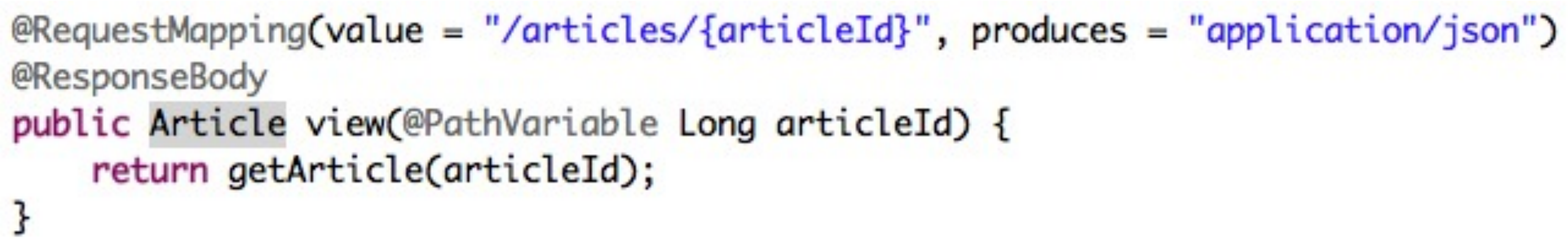
- Jackson
  - `MappingJacksonHttpMessageConverter`
- Jackson 2
  - `MappingJackson2HttpMessageConverter`

JSON

- `@RequestBody`
- `@ResponseBody`
- `@RequestMapping`
  - `consumes="application/json"`
  - `produces="application/json"`

JSON

- Demo

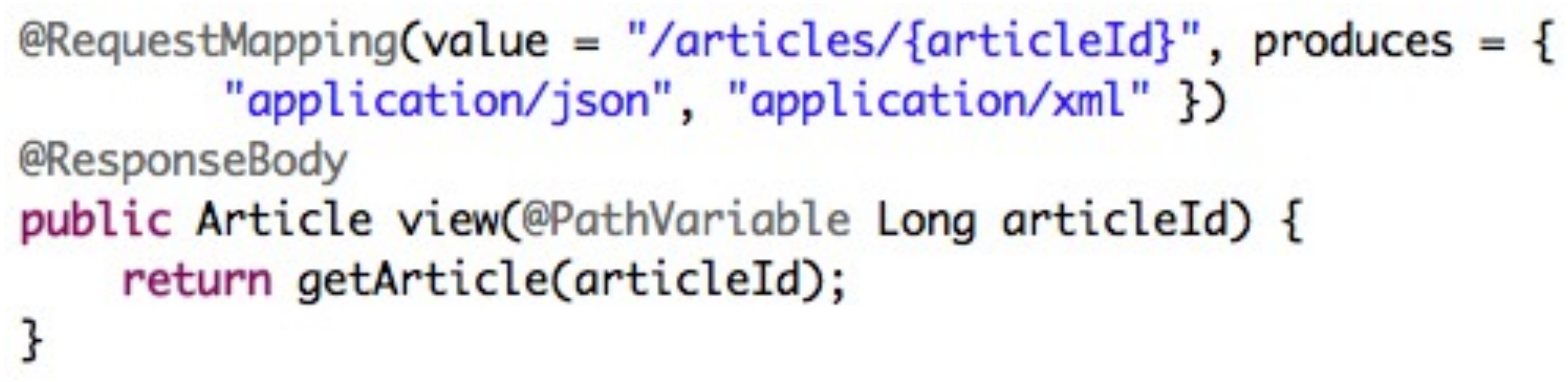


```
@RequestMapping(value = "/articles/{articleId}", produces = "application/json")
@ResponseBody
public Article view(@PathVariable Long articleId) {
    return getArticle(articleId);
}
```

- Request
    - curl -H "Accept: application/json" http://localhost:8080/rest-demo/articles/10

XML

- JAXB 2
  - `Jaxb2RootElementHttpMessageConverter`



```java
@RequestMapping(value = "/articles/{articleId}", produces = {
        "application/json", "application/xml" })
@ResponseBody
public Article view(@PathVariable Long articleId) {
    return getArticle(articleId);
}
```

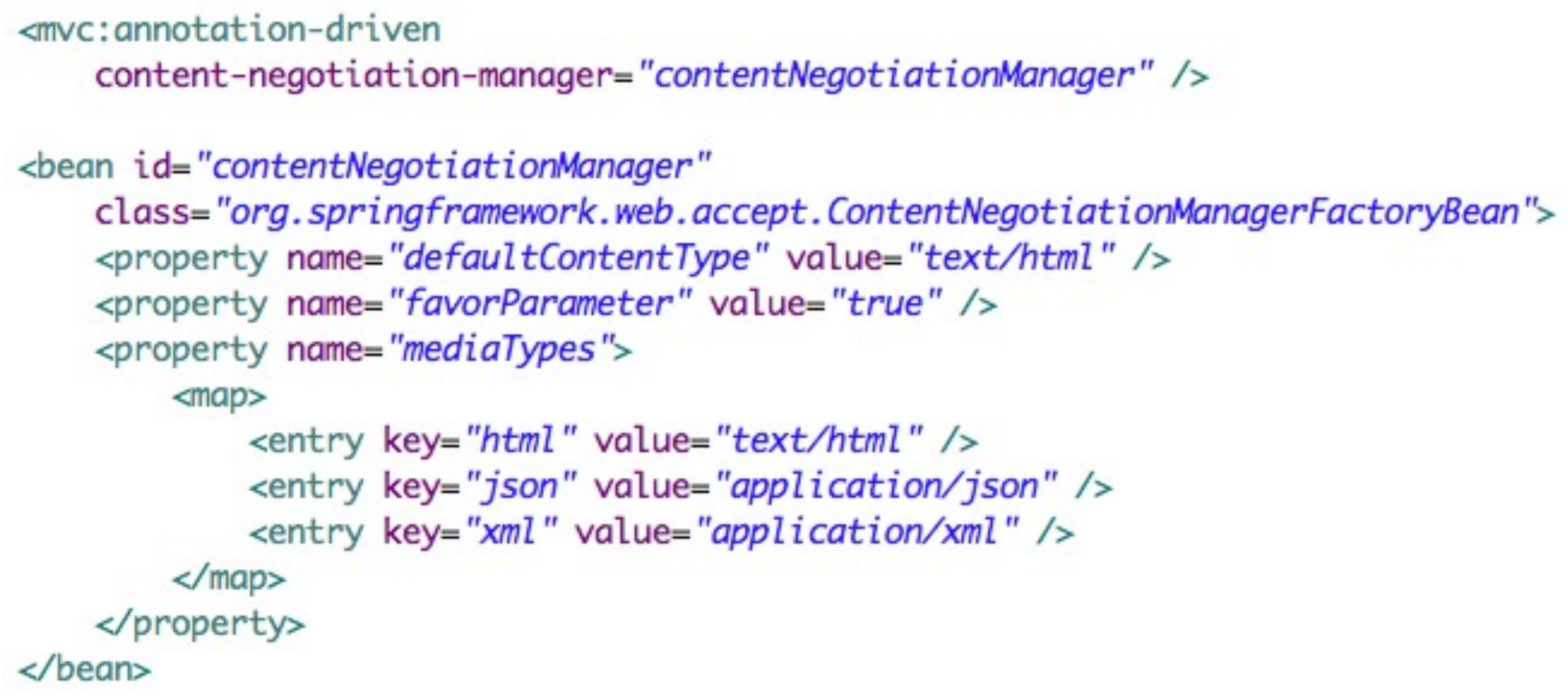
Links

- Link
  - href/rel
- Atom link Ref
  - self/alternate/related/edit
  - first/last/previous/next

Content Negotiation

- serval `@RequestMapping` with different `produces`
- `contentNegotiationManager`
  - file extension
  - `format` parameter
  - `Accept` header
- `ContentNegotiatingViewResolver`

Content Negotiation



```xml
<mvc:annotation-driven
    content-negotiation-manager="contentNegotiationManager" />

<bean id="contentNegotiationManager"
    class="org.springframework.web.accept.ContentNegotiationManagerFactoryBean">
    <property name="defaultContentType" value="text/html" />
    <property name="favorParameter" value="true" />
    <property name="mediaTypes">
        <map>
            <entry key="html" value="text/html" />
            <entry key="json" value="application/json" />
            <entry key="xml" value="application/xml" />
        </map>
    </property>
</bean>
```

HTTP Status Codes

- Use the RIGHT HTTP Status Code!
- `@ResponseStatus`

- http://www.ietf.org/rfc/rfc2616.txt

HTTP Status Codes

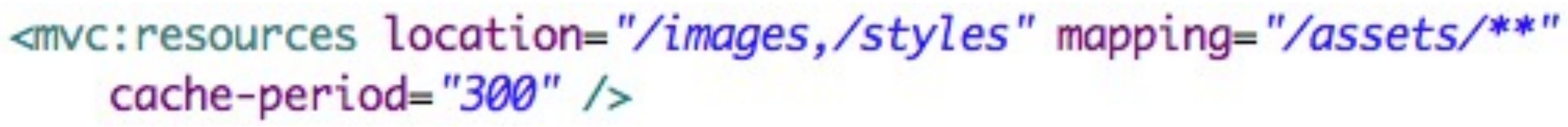
Code	Description	Code	Description
200	OK	400	Bad Request
201	Created	401	Unauthorized
202	Accepted	403	Forbidden
301	Moved Permanently	404	Not Found
303	See Other	410	Gone
304	Not Modified	500	Internal Server Error
307	Temporary Redirect	503	Service Unavailable

Miscellaneous

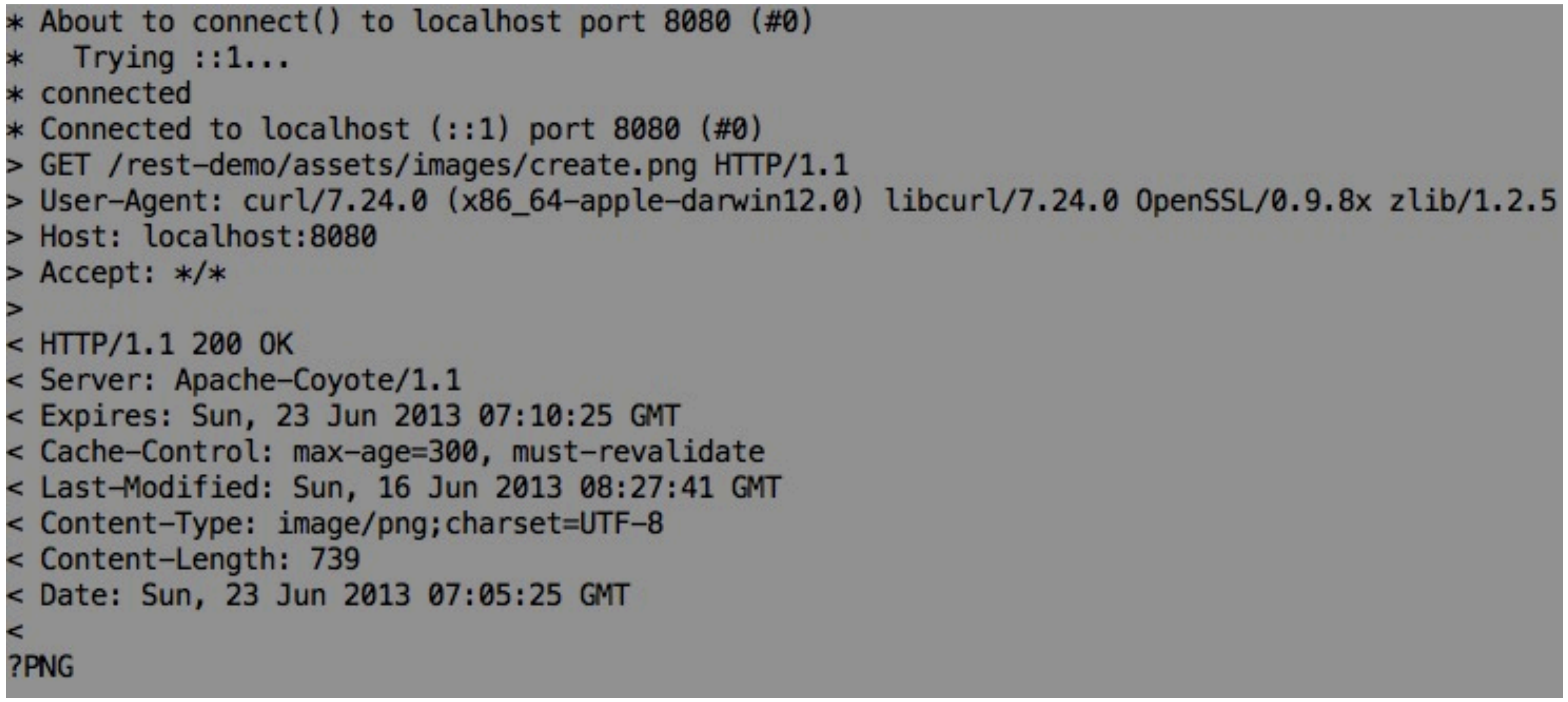
Cache

- Use the RIGHT Cache HTTP Headers
    - `Cache-Control`
    - `Expires`
    - `Date`
    - `Last-Modified`

Cache for Static Resources

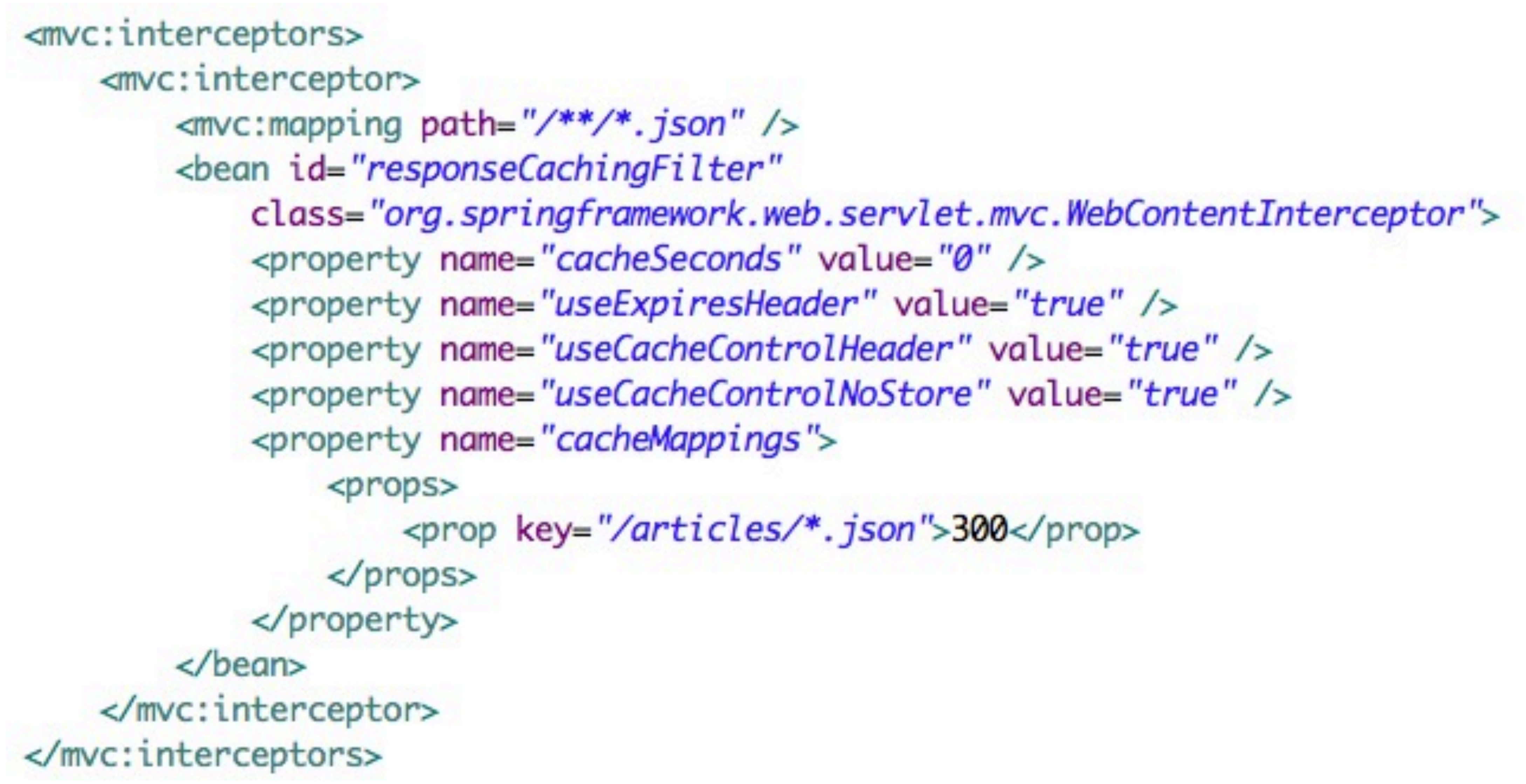


```
<mvc:resources location="/images,/styles" mapping="/assets/**"
    cache-period="300" />
```



```
* About to connect() to localhost port 8080 (#0)
*   Trying ::1...
* connected
* Connected to localhost (::1) port 8080 (#0)
> GET /rest-demo/assets/images/create.png HTTP/1.1
> User-Agent: curl/7.24.0 (x86_64-apple-darwin12.0) libcurl/7.24.0 OpenSSL/0.9.8x zlib/1.2.5
> Host: localhost:8080
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Expires: Sun, 23 Jun 2013 07:10:25 GMT
< Cache-Control: max-age=300, must-revalidate
< Last-Modified: Sun, 16 Jun 2013 08:27:41 GMT
< Content-Type: image/png;charset=UTF-8
< Content-Length: 739
< Date: Sun, 23 Jun 2013 07:05:25 GMT
<
?PNG
```

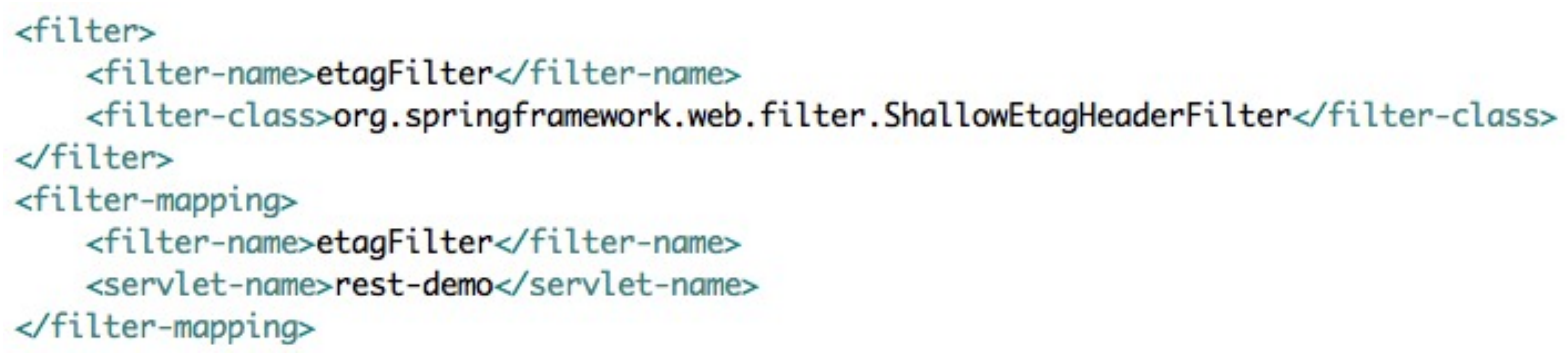
Cache for Dynamic Resources



```
<mvc:interceptors>
    <mvc:interceptor>
        <mvc:mapping path="/**/*.json" />
        <bean id="responseCachingFilter"
            class="org.springframework.web.servlet.mvc.WebContentInterceptor">
            <property name="cacheSeconds" value="0" />
            <property name="useExpiresHeader" value="true" />
            <property name="useCacheControlHeader" value="true" />
            <property name="useCacheControlNoStore" value="true" />
            <property name="cacheMappings">
                <props>
                    <prop key="/articles/*.json">300</prop>
                </props>
            </property>
        </bean>
    </mvc:interceptor>
</mvc:interceptors>
```

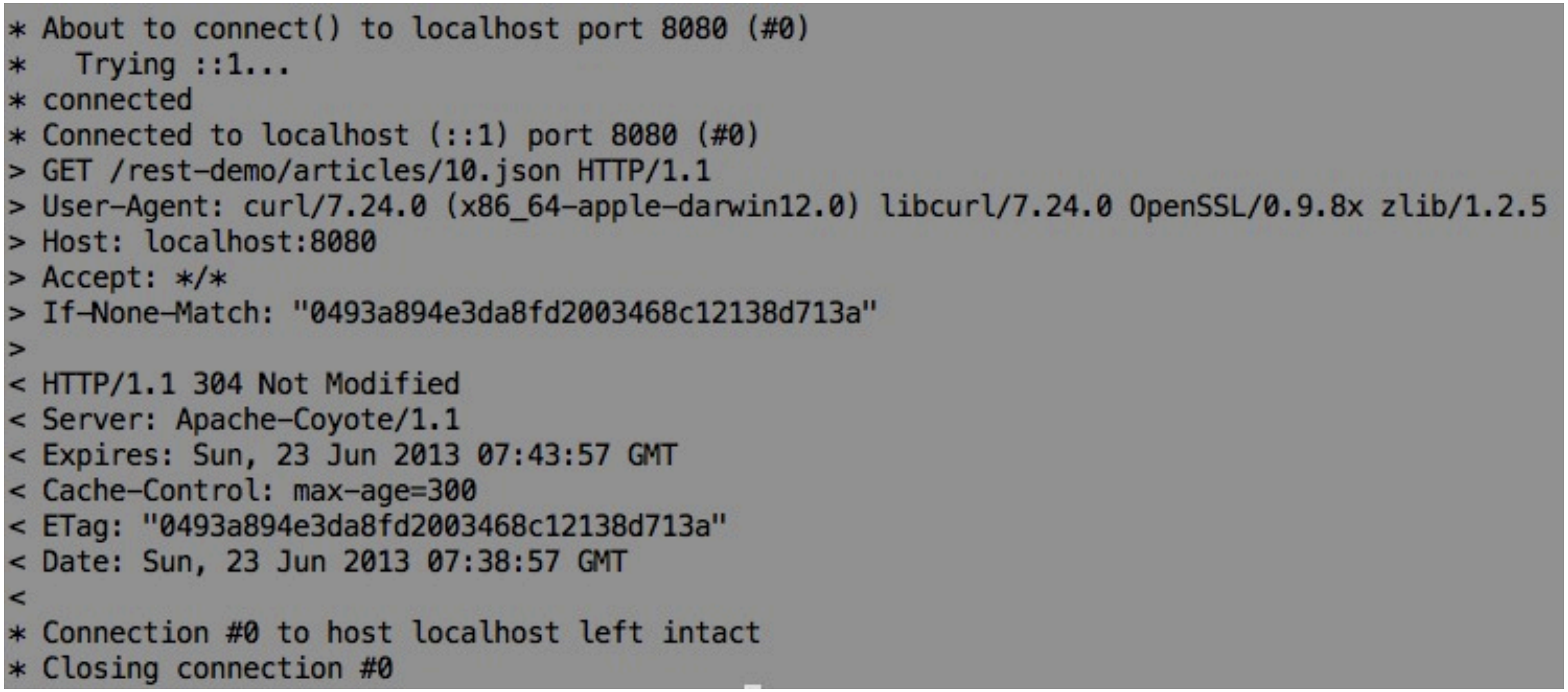
Support for Etags

- `ShallowEtagHeaderFilter`



```
<filter>
    <filter-name>etagFilter</filter-name>
    <filter-class>org.springframework.web.filter.ShallowEtagHeaderFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>etagFilter</filter-name>
    <servlet-name>rest-demo</servlet-name>
</filter-mapping>
```

Support for Etags



```
* About to connect() to localhost port 8080 (#0)
*   Trying ::1...
* connected
* Connected to localhost (::1) port 8080 (#0)
> GET /rest-demo/articles/10.json HTTP/1.1
> User-Agent: curl/7.24.0 (x86_64-apple-darwin12.0) libcurl/7.24.0 OpenSSL/0.9.8x zlib/1.2.5
> Host: localhost:8080
> Accept: */*
> If-None-Match: "0493a894e3da8fd2003468c12138d713a"
>
< HTTP/1.1 304 Not Modified
< Server: Apache-Coyote/1.1
< Expires: Sun, 23 Jun 2013 07:43:57 GMT
< Cache-Control: max-age=300
< ETag: "0493a894e3da8fd2003468c12138d713a"
< Date: Sun, 23 Jun 2013 07:38:57 GMT
<
* Connection #0 to host localhost left intact
* Closing connection #0
```

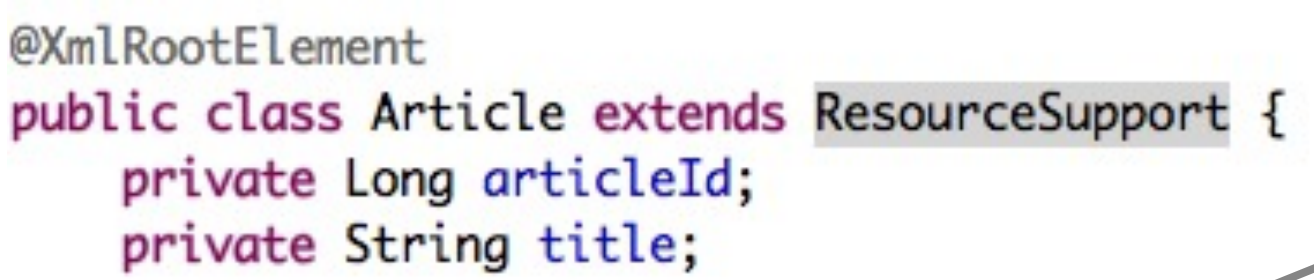
Version

- Try to maintain the compatibility
  - URI
  - Representation
  - Link
- If you can’t, then make a new version
  - subdomain
  - path segments
  - query parameters

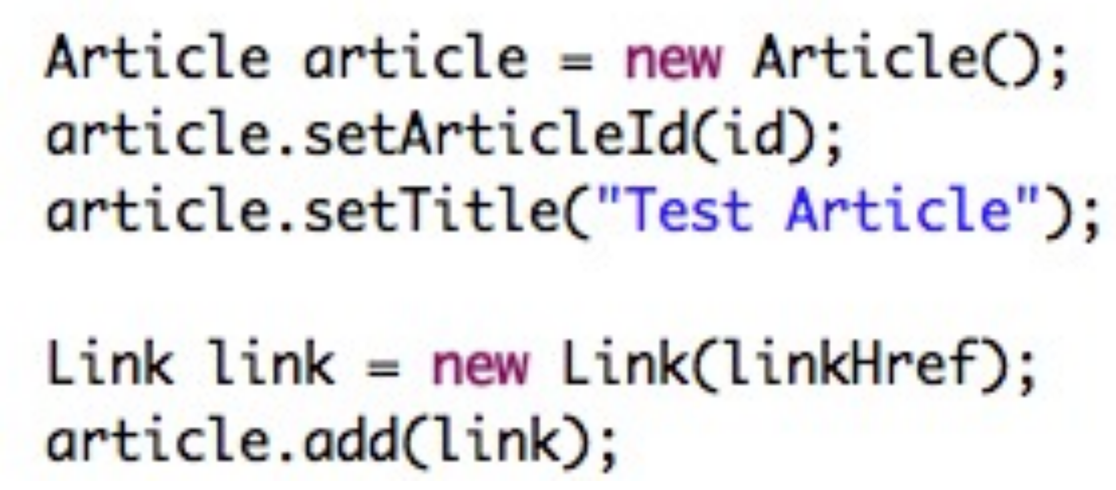
HATEOAS

- Hypermedia as the Engine of Application State
- Spring Hateoas
  - https://github.com/SpringSource/spring-hateoas
  - Better support for Links and Resources

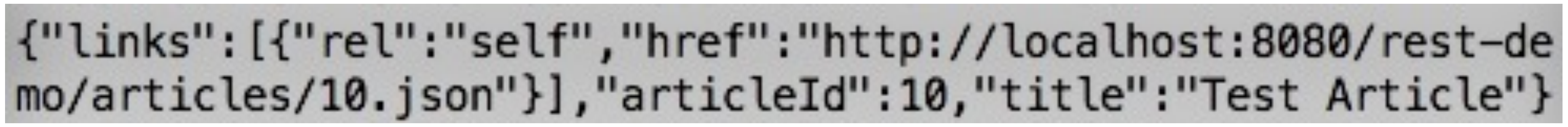
Demo



```java
@XmlRootElement
public class Article extends ResourceSupport {
    private Long articleId;
    private String title;
```



```java
Article article = new Article();
article.setArticleId(id);
article.setTitle("Test Article");

Link link = new Link(linkHref);
article.add(link);
```



```
{"links":[{"rel":"self","href":"http://localhost:8080/rest-de
mo/articles/10.json"}],"articleId":10,"title":"Test Article"}
```

References

- RESTful Web Services Cookbook, O’Reilly
- REST in Practice, O’Reilly
- the official reference of the Spring Framework
- and so many articles on the Internet

Let’s Code!!!