

RESTful Web Applications with Spring 3.0



Arjen Poutsma
Senior Software Engineer
SpringSource

Speaker's qualifications

- Fifteen years of experience in Enterprise Software Development
- Six years of Web service experience
- Development lead of Spring Web Services
- Working on Spring 3.0
- Contributor to various Open Source frameworks: (XFire, Axis2, NEO, ...)

Overview

- RESTful URLs
- URI templates
- Content negotiation
- HTTP method conversion
- ETag support

RESTful URLs

Resources

- URLs are unique identifiers for Resources
- Typically nouns
 - Customer
 - Orders
 - Shopping cart



URLs

[scheme:][//authority][path][?query][#fragment]

<http://www.springsource.com>

<https://mybank.com>

[http://www.google.com/search?q=arjen
%20poutsma](http://www.google.com/search?q=arjen%20poutsma)

[http://java.sun.com/j2se/1.4.2/docs/api/java/lang/
String.html#indexOf\(int\)](http://java.sun.com/j2se/1.4.2/docs/api/java/lang/String.html#indexOf(int))

Paths

- Represents hierarchy
- Represents value for consumers
- Collections on higher levels

Example

Path	Description
<code>/hotels</code>	List of all hotels
<code>/hotels/westindiplomat</code>	Details of Westin Diplomat
<code>/hotels/westindiplomat/ bookings</code>	List of bookings of Westin Diplomat
<code>/hotels/westindiplomat/ bookings/42584</code>	Individual booking

No Hierarchy?

Path	Description
<code>maps/24.9195,17.821</code>	Commas
<code>maps/24.9195;17.821</code>	Semicolons

Query Variables

- Input for algorithms
- Get ignored by proxies
- Often abused
 - `http://api.flickr.com/services/rest/?method=flickr.photos.search&tags=penguin`
 - `hotels?hotelId=westindiplomat`

Do's

- Hierarchies
- Collections
- Separate URLs for useful resources

Dont's

- Queries abuse
- RPC in disguise
- Verbs in URLs

URI Templates

URI Templates

- URI-like string, containing one or more variable names
- Variables can be substituted for values to become a URI
- Helps to create nice, “RESTful” URLs

Examples

URI Template	Request	Variables
<code>/hotels/{hotelId}</code>	<code>/hotels/ westindiplomat</code>	<code>hotelId= westindiplomat</code>
<code>/hotels/{hotelId}/ bookings</code>	<code>/hotels/ westindiplomat/ bookings</code>	<code>hotelId= westindiplomat</code>
<code>/hotels/{hotelId}/ bookings/ {bookingId}</code>	<code>/hotels/ westindiplomat/ bookings/21</code>	<code>hotelId= westindiplomat bookingId=21</code>

@PathVariable

- Spring 3.0 M1 introduced the @PathVariable annotation
- Allows you to use URI Templates in @MVC

Example

```
@Controller
@RequestMapping("/hotels/{hotel}/**")
public class HotelsController {

    @RequestMapping
    public void handleHotel(@PathVariable("hotel") String hotel) {
        // ...
    }

    @RequestMapping("bookings/{booking}")
    public void handleBooking(@PathVariable("hotel") String hotel,
        @PathVariable int booking) {
        // ...
    }
}
```


Demo

URI Templates

Content Negotiation

Representations

- Access resource through representations
- More than one representation possible
- Desired representation in Accept header
 - Or file extension
- Delivered representation show in Content-Type



Views

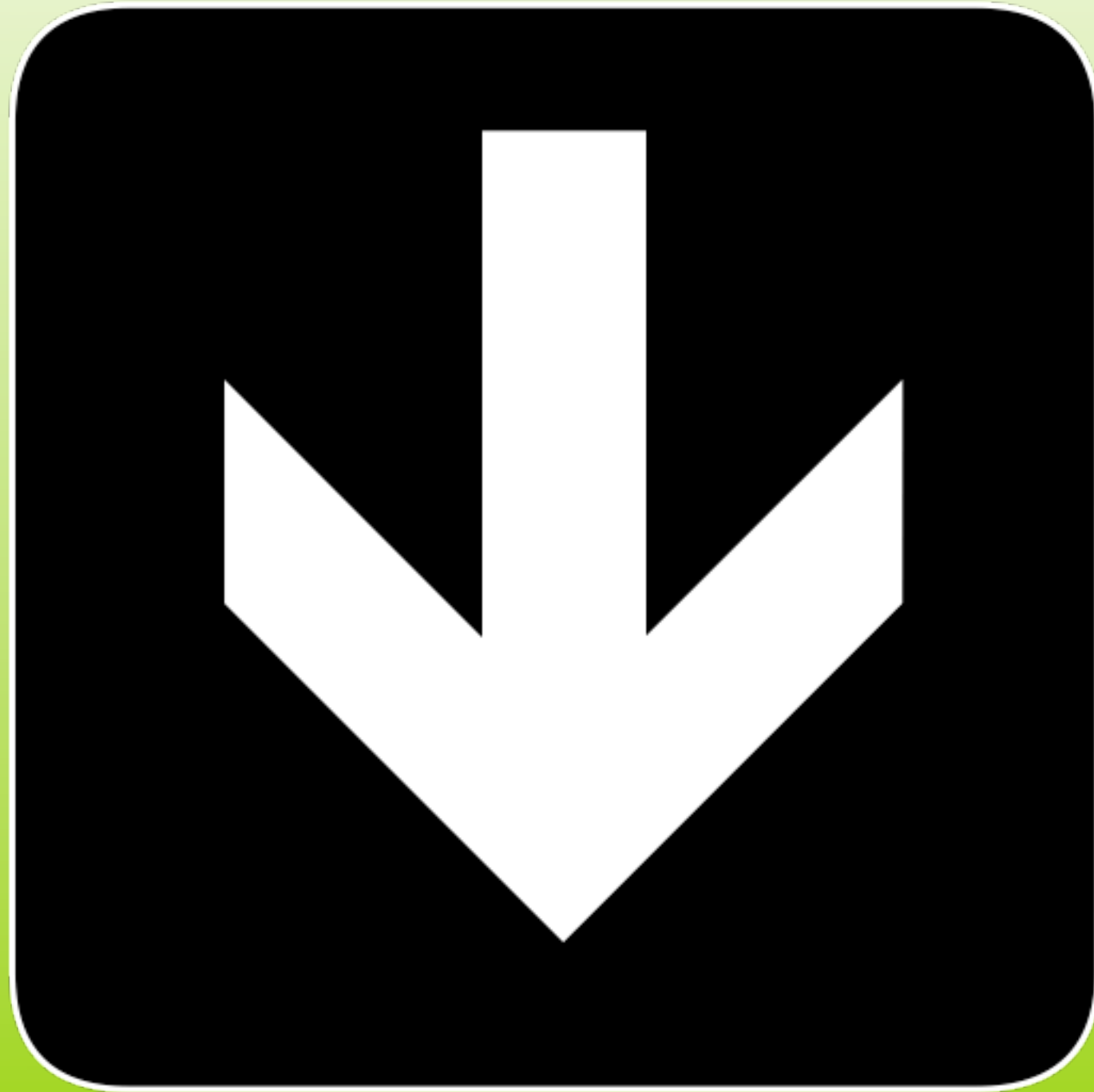
Mime Type	View	When
application/xml	MarshallingView	3.0 M2/SWS 1.5
application/atom+xml	AtomFeedView	3.0 MI
application/rss+xml	RssFeedView	3.0 MI
application/json	JsonView	Spring-JS

No Demo

Coming in Spring 3.0 M2!

HTTP Method conversion

Uniform Interface



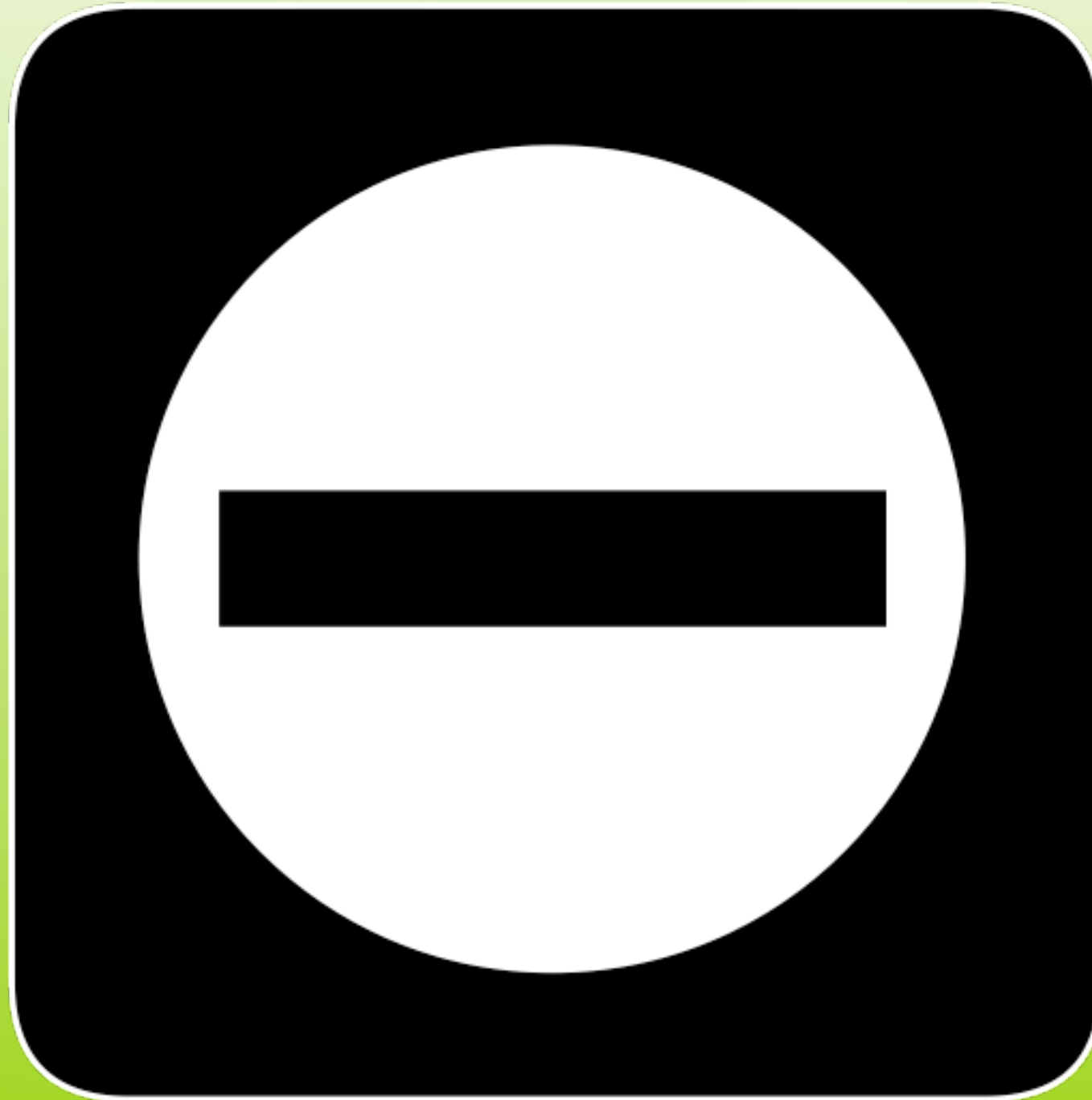
Uniform Interface



Uniform Interface

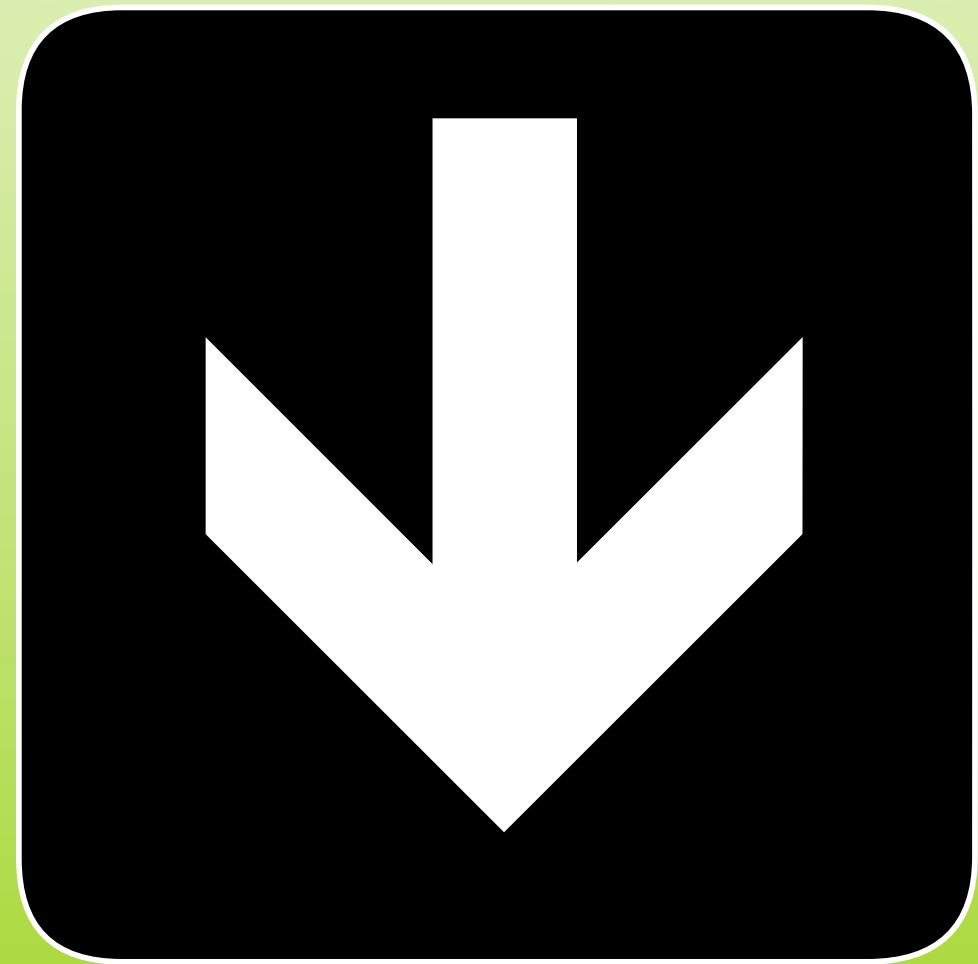


Uniform Interface



GET

- Retrieves Representation of Resource
- Safe operation



GET Example

```
GET /hotels  
Host: example.com  
...
```



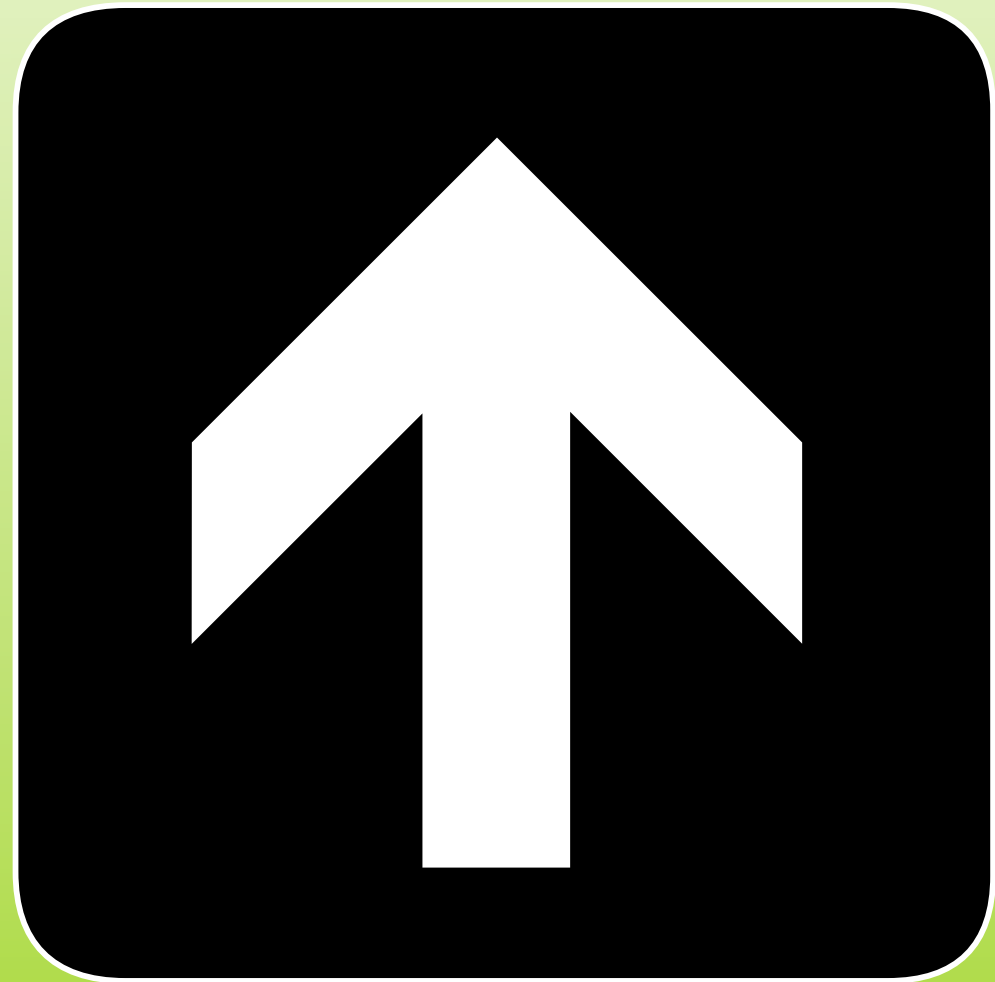
GET Example

```
GET /hotels  
Host: example.com  
...
```

```
HTTP/1.1 200 OK  
Date: ...  
Content-Length: 1456  
Content-Type:  
application/xml  
  
<hotels>  
...  
</hotels>
```

PUT

- Updates resource
- Creates resource, when the destination URI is known
- Idempotent



PUT Example

```
PUT /hotels/2  
Host: example.com
```

```
<hotel>  
...  
</hotel>
```



PUT Example

```
PUT /hotels/2  
Host: example.com
```

```
<hotel>  
...  
</hotel>
```

```
HTTP/1.1 201 Created  
Date: ...  
Content-Length: 0
```

```
...
```


POST

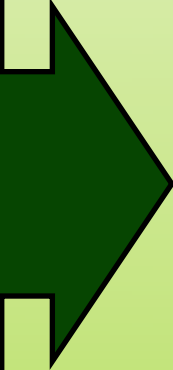
- Creates new Resource
- Child of other Resource
- Response Location header is used to indicate URI of child



POST Example

```
POST /hotels/1/  
  bookings  
Host: example.com
```

```
<booking>  
...  
</booking>
```



POST Example

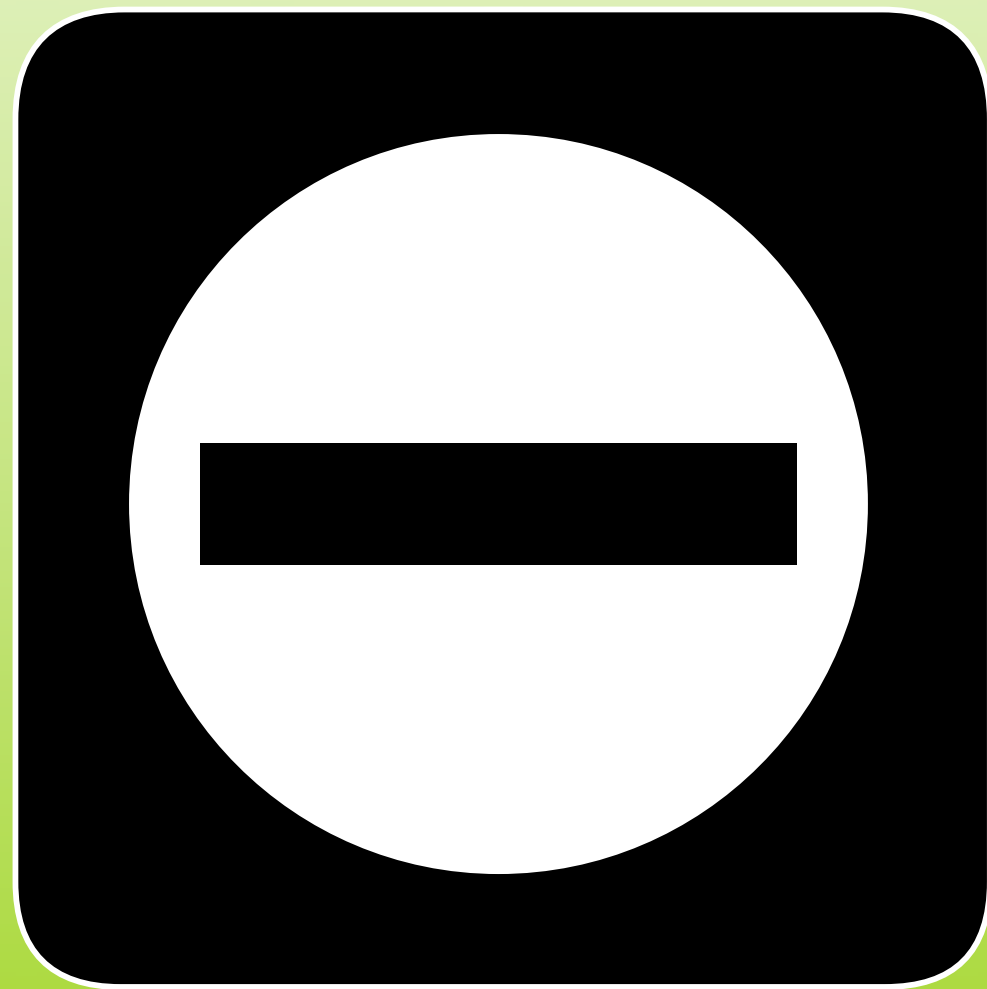
```
POST /hotels/1/  
  bookings  
Host: example.com
```

```
<booking>  
...  
</booking>
```

```
HTTP/1.1 201 Created  
Date: ...  
Content-Length: 0  
Location:  
http://example.com/  
hotels/1/bookings/50  
...
```


DELETE

- Deletes a resource
- Idempotent



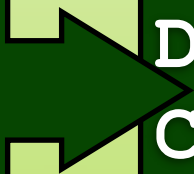
DELETE Example

```
DELETE /hotels/3  
Host: example.com  
...
```



DELETE Example

```
DELETE /hotels/3  
Host: example.com  
...
```



```
HTTP/1.1 204 No Content  
Date: ...  
Content-Length: 0  
...
```

One Problem...

- HTML only supports GET and POST
- Possible workarounds:
 - Javascript
 - POST with hidden method parameter
 - `HiddenHttpMethodFilter`

Demo

HTTP Method Conversion

ETag Support

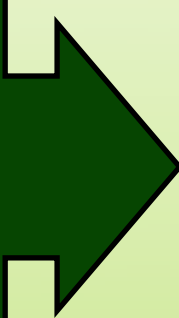
GET is Cacheable

- Servers returns ETag header
- Send on subsequent retrieval
- If not changed, 304 (Not Modified) is returned



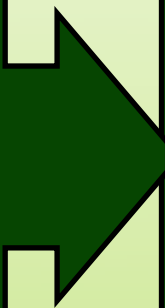
Conditional GET

```
GET /hotels  
Host: example.com  
...
```



Conditional GET

```
GET /hotels  
Host: example.com  
...
```



```
HTTP/1.1 200 OK  
Date: ...  
ETag: "b4bdb3"  
Content-Length: 1456  
...
```

Conditional GET

```
GET /hotels  
Host: example.com  
...
```

```
HTTP/1.1 200 OK  
Date: ...  
ETag: "b4bdb3"  
Content-Length: 1456  
...
```

```
GET /hotels  
If-None-Match: "b4bdb3"  
Host: example.com  
...
```

Conditional GET

```
GET /hotels
Host: example.com
...
```

```
HTTP/1.1 200 OK
Date: ...
ETag: "b4bdb3"
Content-Length: 1456
...
```

```
GET /hotels
If-None-Match: "b4bdb3"
Host: example.com
...
```

```
HTTP/1.1 304 Not
Modified
Date: ...
ETag: "b4bdb3"
Content-Length: 0
```

ShallowEtagHeaderFilter

- Introduced in Spring 3.0 M1
- Creates ETag header based on MD5 of rendered view
- Saves bandwidth only
- Deep ETag support comes in M2
 - Through @RequestHeader

Demo

ETag Support

Summary

- REST uses the Web like it should be
- Spring 3.0 will help you create RESTful Web Applications

Q&A

Why not JAX-RS?

- Does not help the current @MVC users
- Semantics are different
 - No mix-and-match
- Web Service focus

- We do plan to provide JAX-RS integration in the future