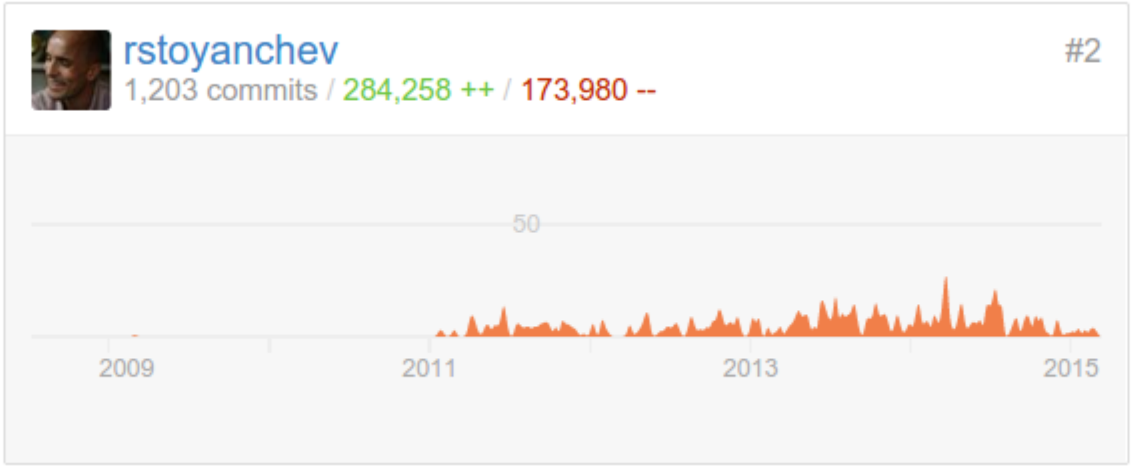
Spring 4 Web Applications

Rossen Stoyanchev
Pivotal Inc

About the speaker

- Spring Framework committer
- Spring MVC
- Spring WebSocket and Messaging


rstoyanchev
1,203 commits / 284,258 ++ / 173,980 --
#2

Spring MVC

- Since 2003 (circa JDK 1.4)
- Before Java annotations, REST, SPAs, ...
- Continued success, evolution
- Most popular status today

Programming Model Evolution

@Controller	2.5 (2007)
REST	3.0 (2009)
Async requests	3.2 (2012)
WebSocket messaging	4.0 (2013)

Keys to Success

- Simple, clean design at the core
- Friendly to extension
- Embraces HTTP and REST
- Community requests

Always Evolving

- One of most actively developed parts of Spring Framework

- Continuous flow of ideas and requests from the community

- Improvements, new features, even modules with each new version

@MVC

@Controller

- @InitBinder
- @ModelAttribute
- **@RequestMapping**
- @ExceptionHandler

@RestController

```
@RestController
public class MyController {

    @RequestMapping ~~@ResponseBody~~
    public Foo handle() { … }

    @RequestMapping ~~@ResponseBody~~
    public Bar handle() { … }

}
```

Beyond Class Hierarchy

@ControllerAdvice

- @InitBinder
- @ModelAttribute
- @ExceptionHandler

Selectors

```
@ControllerAdvice  => “Apply to every @Controller”

@ControllerAdvice(basePackages = "org.app.module")

@ControllerAdvice(annotations = RestController.class)

@ControllerAdvice(assignableTypes =
        {BaseController1.class, BaseController2.class})
```

ResponseEntityExceptionHandler

- Base class for use with @ControllerAdvice
- Handle Spring MVC exceptions
- REST API error details in response body

ResponseBodyAdvice

- Interface for use with @ControllerAdvice
- Customize response before @ResponseBody & ResponseEntity are written
- Built-in usages
  - @JsonView on @RequestMapping methods
  - JSONP

Further Jackson Support

- Use Jackson for both JSON and XML
- ObjectMapper builder
- Highly recommended read:

  https://spring.io/blog/2014/12/02/latest-jackson-integration-improvements-in-spring

@RequestMapping methods

- java.util.Optional (JDK 1.8) support
- ListenableFuture return value
- ResponseEntity/RequestEntity builders
- Links to @MVC methods
- @ModelAttribute method ordering

ResponseEntityBuilder

```
String body = "Hello";
HttpHeaders hdrs = new HttpHeaders()
headers.setLocation(location);
new ResponseEntity<String>(body, hdrs, CREATED);
```

vs

```
ResponseEntity.created(location).body("Hello");
```

RequestEntityBuilder

```
HttpHeaders headers = new HttpHeaders();
headers.setAccept(MediaType.APPLICATION_JSON);
new HttpEntity("Hello", headers);
```

vs

```
RequestEntity.post(uri)
    .accept(MediaType.APPLICATION_JSON)
    .body("Hello");
```

Link to @RequestMapping

- Simulate controller method invocation

```
fromMethodCall(on(MyController.class).getAddress("US"))
     .buildAndExpand(1).toUri();
```

- Uses proxy, similar to testing w/ mocks

- See section on Building URIs

How to link from views?

- Refer to @RequestMapping by name
- Default name assigned to every mapping
    - or use `@RequestMapping(name=”..”)`
- See subsection in Building URIs

@ModelAttribute Ordering

Creates “foo”

<- Call this 1st

```
@ModelAttribute("foo")
public Object getFoo() {
}
```

Uses “foo”

```
@ModelAttribute("bar")
public Object getBar(@ModelAttribute("foo") Object foo) {
}
```

Static Resources

- Key topic for web applications today
  - Optimize .. minify, concatenate
  - Transform .. sass, less
  - HTTP caching .. versioned URLs
  - CDN
  - Prod vs dev

Static Resources in 4.1

- Build on existing ResourceHttpRequestHandler
- Add abstractions to resolve and transform resources in a chain
- Prepare “public” resource URL

URL “Fingerprinting”

- HTTP “cache busting”
- Version URL with content-based hash
- Add aggressive cache headers (e.g. +1 year)

Example URL:

`“/css/font-awesome.min-7fbe76cdac.css”`

Static Resources Continued

See Resource Handling talk on Youtube,

browse the slides,

or check the source code.

Groovy Markup Templating

- DRY markup based on Groovy 2.3
- Like HAML in Ruby on Rails

```
yieldUnescaped '<!DOCTYPE html>'
html(lang:'en') {
    head {
        title('My page')
    }
    body {
        p('This is an example of HTML contents')
    }
}
```

MVC Config

- We now have ViewResolver registry
- ViewController can do more
  - redirects, 404s, etc.
- Patch matching by popular demand
  - suffix patterns, trailing slashes, etc.

Servlet 3 Async Requests

- Since v3.2
  - Long polling, HTTP streaming
- Server can push events to client
  - chat, tweet stream
- Relatively simple, close to what we know
- Not easy for more advanced uses
  - games, finance, collaboration

Web Messaging

- WebSocket protocol
  - bi-directional messaging between client & server
- SockJS fallback
  - WebSocket emulation (IE < 10, proxy issues, etc.)
- STOMP
  - Simple messaging sub-protocol
  - Like HTTP over TCP

Why not just WebSocket?

- Too low level
- Practically a TCP socket
- Just like HTTP enables RESTful architecture, STOMP enables messaging
- In the absence of a protocol, a custom protocol will have to be used

Example STOMP Frame

```
SEND
destination:/app/greetings
content-type:text/plain

Hello world!
```

Handle a Message

```
@Controller
public class PortfolioController {

  @MessageMapping("/greetings")
  public void add(String payload) { … }




}
```

Messaging + REST

```
@Controller
public class PortfolioController {

  @MessageMapping("/greetings")
  public void add(String payload) { … }

  @RequestMapping("/greetings", method=GET)
  public String get() { … }

}
```

SockJS

- Exact same WebSocket API
- Different transports underneath
    - long polling, HTTP streaming
- Wide range of browsers and versions
- WebSocket alone not practically usable without fallback options today

WebSocket Continued

See presentation:

https://github.com/rstoyanchev/springx2013-websocket

There is also a video available.

Spring Boot

- You are an expert but how long would it take you to start a new web application?
- Lot of choices to be made
- Boot makes reasonable default choices
- So you can be up and running in minutes

Spring Boot Web App

```java
@RestController
@EnableAutoConfiguration
public class Example {

    public static void main(String[] args) {
        SpringApplication.run(Example.class, args);
    }

    @RequestMapping("/")
    public String home() {
        return "Hello World!";
    }
}
```

REST API Docs

- Good REST API documentation can not be fully generated

- Every good API guide has some stories and use cases with example usage

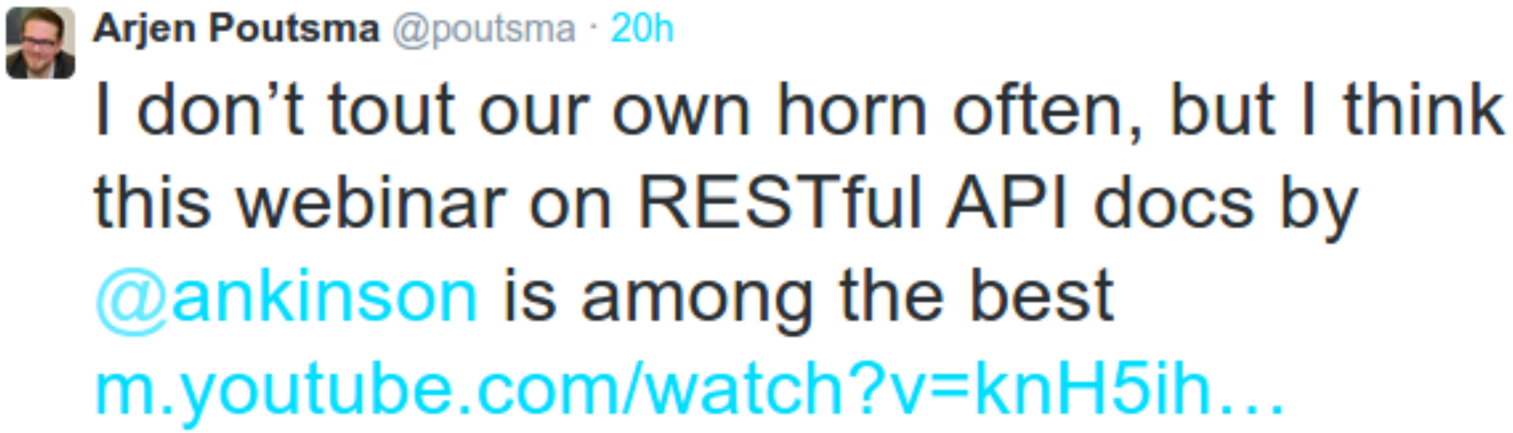
- Yet manually writing it all is too much

Spring REST Docs

- What if you could write real tests that demonstrate your REST API?
- Using Spring MVC Test…
- Then insert the code w/ actual output in your Asciidoctor documentation

Spring REST Docs Continued

Check out this webinar by Andy Wilkinson



Arjen Poutsma @poutsma · 20h

I don’t tout our own horn often, but I think this webinar on RESTful API docs by @ankinson is among the best m.youtube.com/watch?v=knH5ih…

Server-Sent Events v4.2

```
@RequestMapping
public ResponseEntity<SseEmitter> handle() {
    SseEmitter emitter = new SseEmitter();
    // ...
    return emitter;
}

// Later from another thread
emitter.send(event().name("foo").data(foo));
...
emitter.complete();
```

Server-Sent Events v4.2

```
@RequestMapping
public ResponseEntity<SseEmitter> handle() {
  if ( … ) {
     return ResponseEntity.status(204).body(null);
  }
  else {
     // …
     ResponseEntity.ok(sseEmitter);
  }
}
```

SPR-12672

HTTP Caching v4.2

- Comprehensive update according to the most recent HTTP 1.1. spec updates
- Central and per-request support for all Cache-Control directives
- A deep eTag strategy

SPR-11792

CORS v4.2

- Built-in support within Spring MVC
- Both central and fine-grained
- @CrossOrigin
- CorsConfigurationSource

SPR-9278

Custom @RequestMapping v4.2

```
@RequestMapping(
        method = RequestMethod.POST,
        produces = MediaType.APPLICATION_JSON_VALUE
        consumes = MediaType.APPLICATION_JSON_VALUE)
public @interface PostJson {
    String value() default "";
}
```

```
@PostJson("/input")
public Output myMethod(Input input) {
}
```

SPR-12296

JavaScript Templating v4.2

- Server-side JavaScript templates
- See very long SPR-12266
- Current plan is to plug Nashorn (JDK 1.8) behind the ViewResolver/View contracts
- Much like we did for Groovy in 4.1

STOMP Client v4.2

- There aren't any good Java clients
- So we've decided to write one
- Good for testing at least
- Like we added SockJS Java client in 4.1

SPR-11588

Topical Guides

- Part of effort to overhaul Spring Framework reference documentation
- Separate “conceptual” information from pure reference
- Example guides
  - “What is Spring”, “Intro to Spring Config”, etc.
- Track topical-guides repo

Questions

http://twitter.com/rstoya05

http://pivotal.io