Spring 4 Web Apps

Rossen Stoyanchev

DEVNEXUS™

About the Speaker

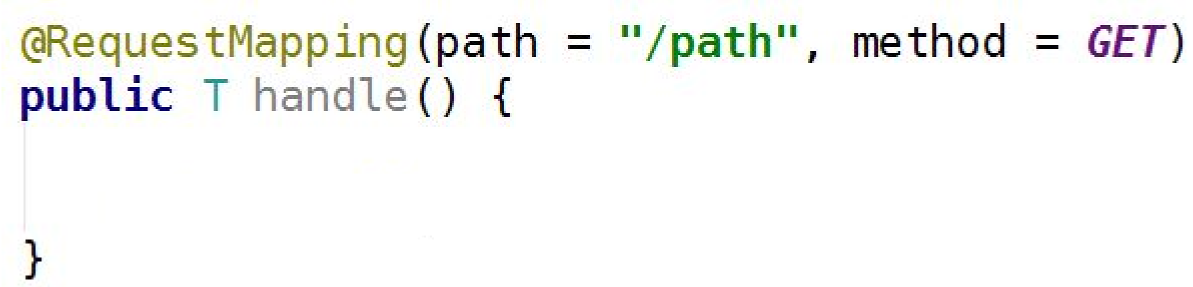
- Spring Framework committer
- Focus on web functionality
- Reactive Spring effort

Spring Framework Releases

03-15-2017	**5.0**	11 months
05-05-2016	4.3	09 months
07-31-2015	4.2	11 months
09-04-2014	4.1	08 months
12-12-2013	**4.0**	12 months
12-13-2012	3.2	12 months
12-13-2012	3.1	12 months

Async,
long polling, streaming,
non-blocking

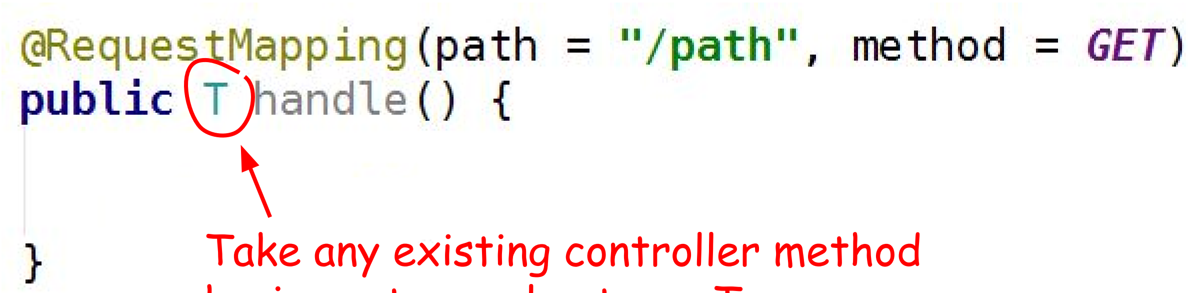
Asynchronous Request Handling



```
@RequestMapping(path = "/path", method = GET)
public T handle() {


}
```

Asynchronous Request Handling

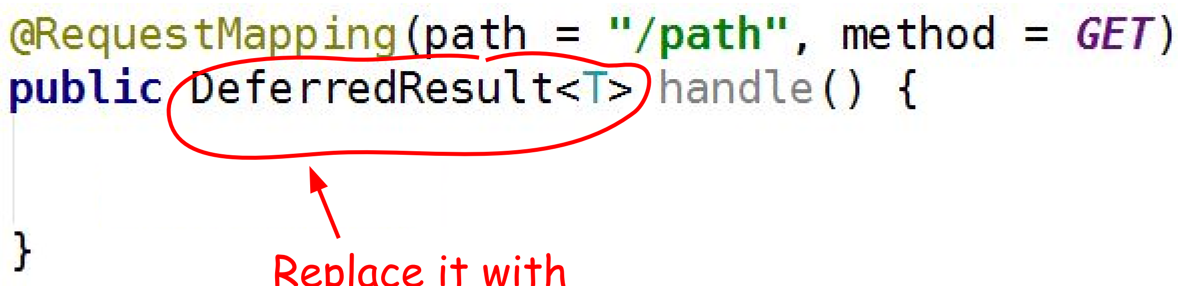


```java
@RequestMapping(path = "/path", method = GET)
public T handle() {


}
```

Take any existing controller method
having return value type <T>

Asynchronous Request Handling

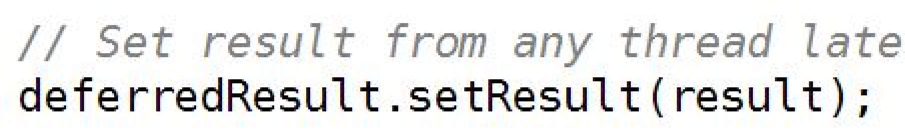


```java
@RequestMapping(path = "/path", method = GET)
public DeferredResult<T> handle() {


}
```

Replace it with
DeferredResult<T>

Asynchronous Request Handling



```
// Set result from any thread later on...
deferredResult.setResult(result);
```

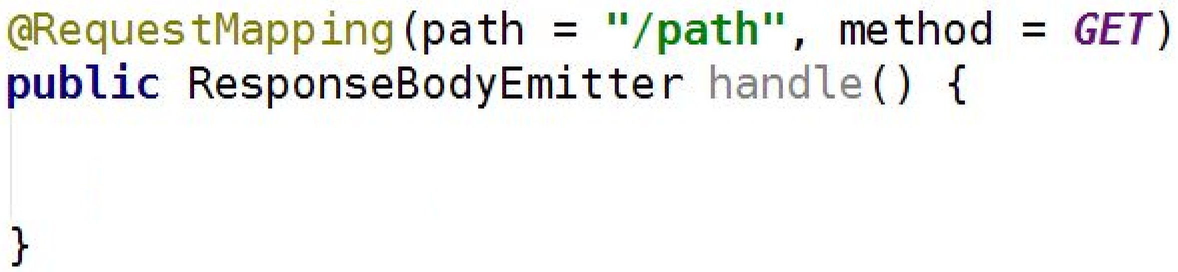
DeferredResult

- Supported with **any** existing controller method
  - just replace return value

- Good for latency-sensitive uses cases
  - chat, scatter-gather
  - long polling

- Only supports setting a **single async result**

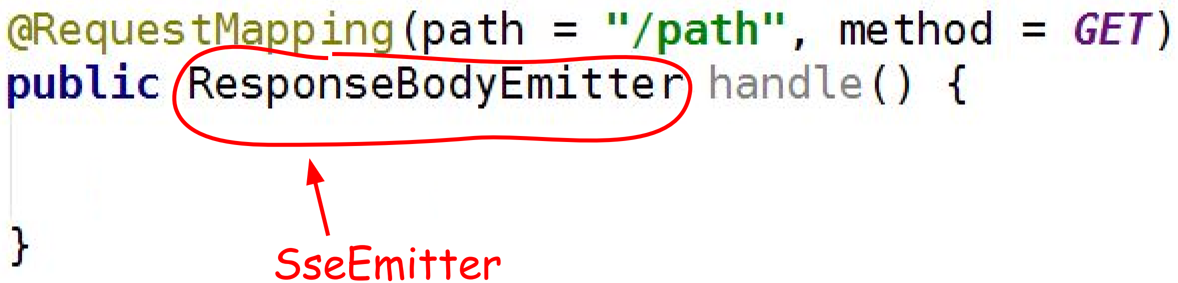
HTTP Streaming



```
@RequestMapping(path = "/path", method = GET)
public ResponseBodyEmitter handle() {


}
```

HTTP Streaming

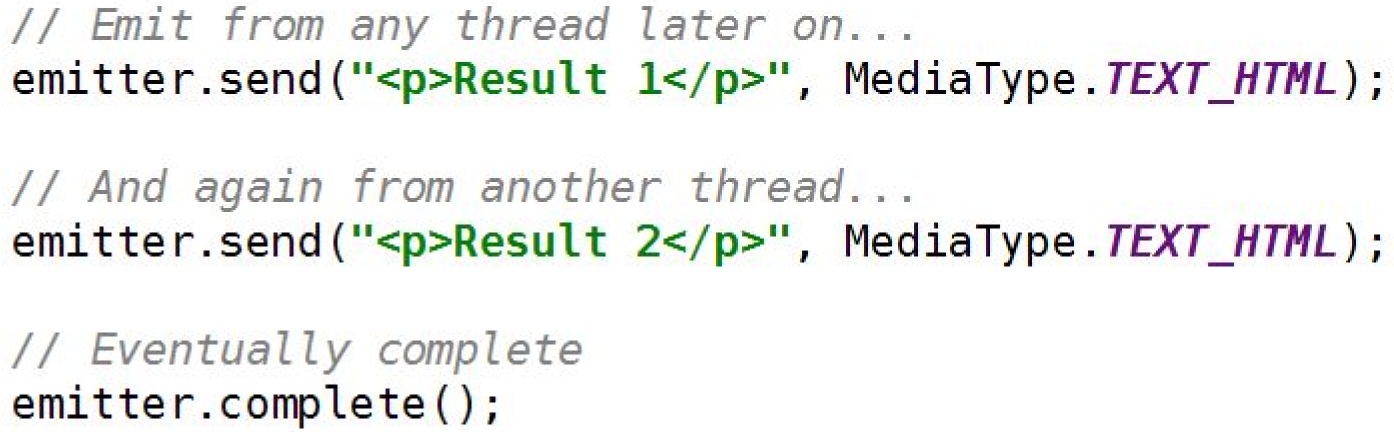


```
@RequestMapping(path = "/path", method = GET)
public ResponseBodyEmitter handle() {


}
```

SseEmitter
for server-sent events

HTTP Streaming

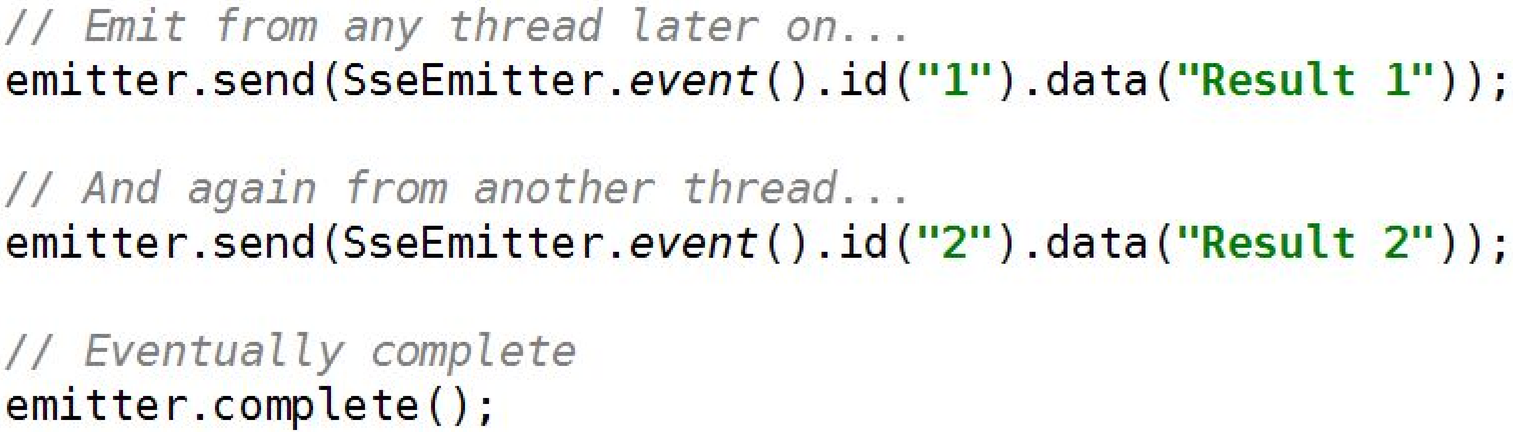


```
// Emit from any thread later on...
emitter.send("<p>Result 1</p>", MediaType.TEXT_HTML);

// And again from another thread...
emitter.send("<p>Result 2</p>", MediaType.TEXT_HTML);

// Eventually complete
emitter.complete();
```

Server-Sent Events Streaming



```java
// Emit from any thread later on...
emitter.send(SseEmitter.event().id("1").data("Result 1"));

// And again from another thread...
emitter.send(SseEmitter.event().id("2").data("Result 2"));

// Eventually complete
emitter.complete();
```

HTTP Streaming

- Similar to `@ResponseBody` or `ResponseEntity` method

- Except write many objects

- At different times, from different threads

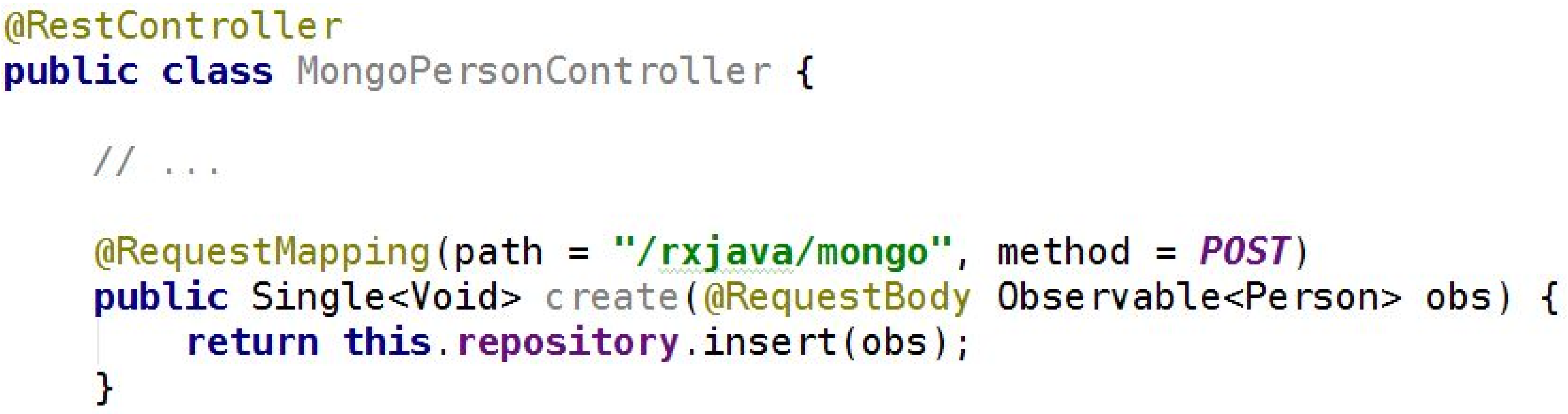
Non-Blocking & Async (v5.0)

❖ Support for reactive architectures

❖ Spring MVC programming model with reactive web engine

❖ Non-blocking I/O and request handling

Non-Blocking Controller (v5.0)

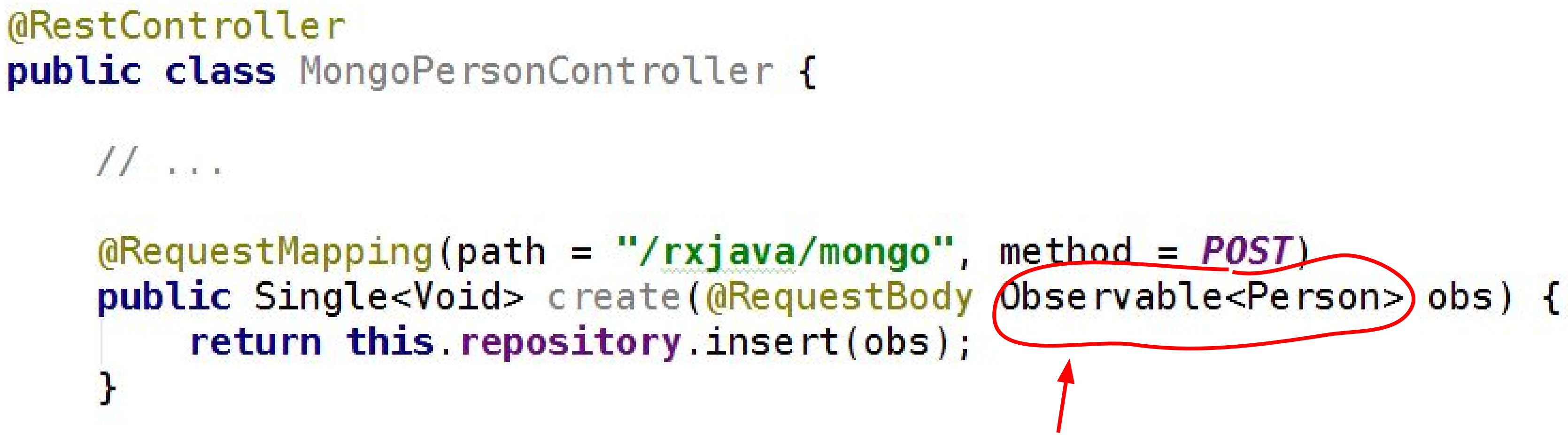


```
@RestController
public class MongoPersonController {

    // ...

    @RequestMapping(path = "/rxjava/mongo", method = POST)
    public Single<Void> create(@RequestBody Observable<Person> obs) {
        return this.repository.insert(obs);
    }
```

Non-Blocking Controller (v5.0)



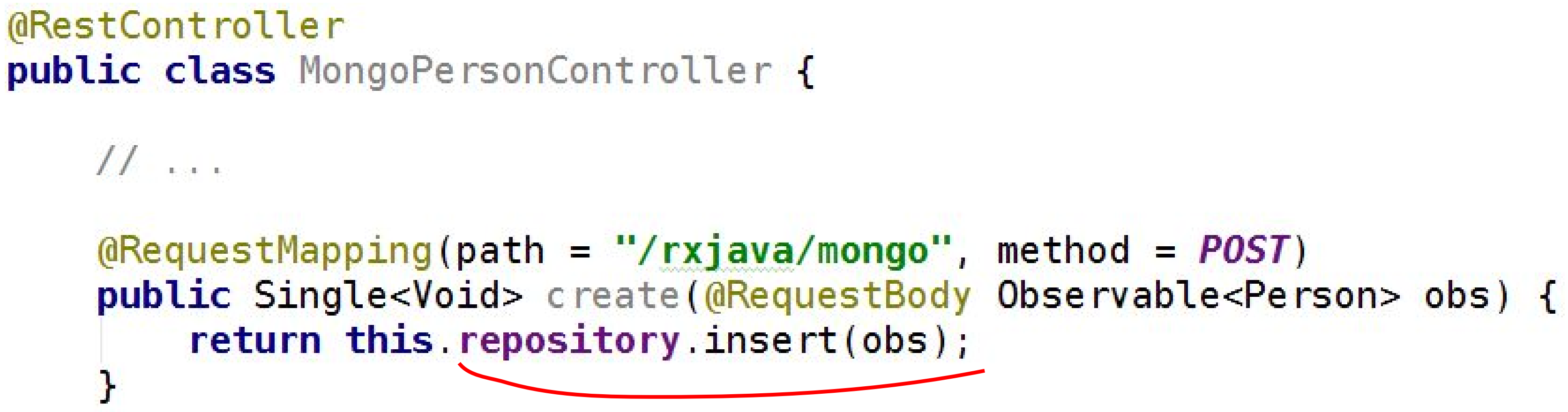
```
@RestController
public class MongoPersonController {

    // ...

    @RequestMapping(path = "/rxjava/mongo", method = POST)
    public Single<Void> create(@RequestBody Observable<Person> obs) {
        return this.repository.insert(obs);
    }
```

Non-blocking request body

Non-Blocking Controller (v5.0)



```java
@RestController
public class MongoPersonController {

    // ...

    @RequestMapping(path = "/rxjava/mongo", method = POST)
    public Single<Void> create(@RequestBody Observable<Person> obs) {
        return this.repository.insert(obs);
    }
```

Non-blocking handling

My 2nd talk tomorrow at 4pm:
Reactive Web Apps

WebSocket Architecture

WebSocket Messages

- Messaging architecture

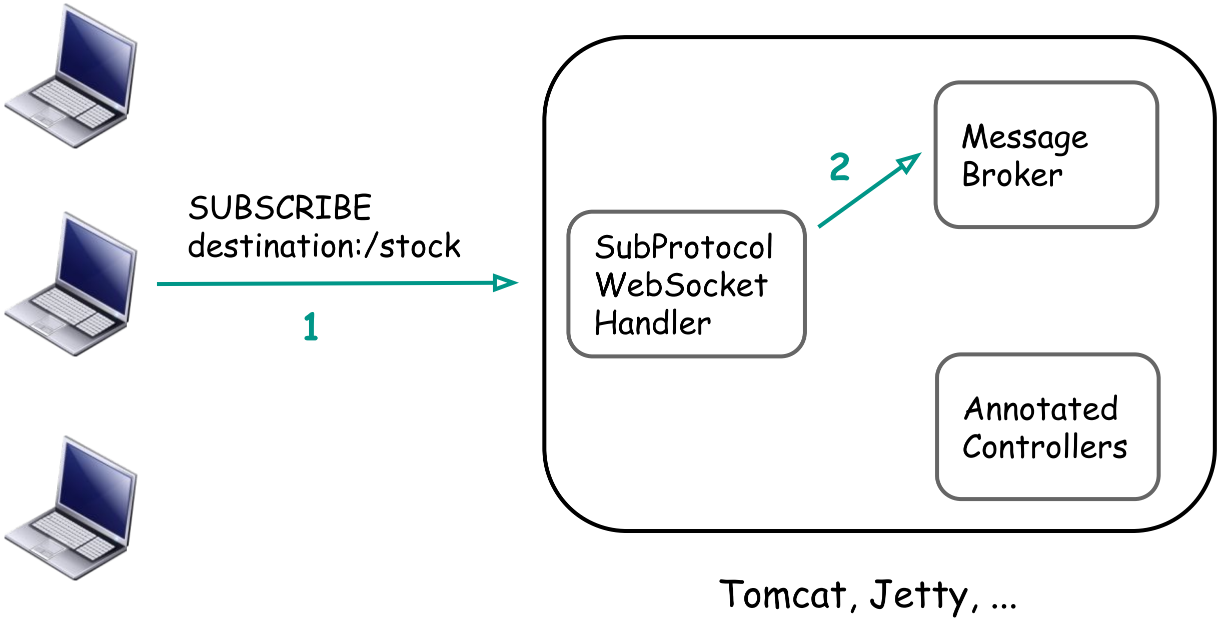
- In contrast to REST architecture

- More like traditional messaging like JMS

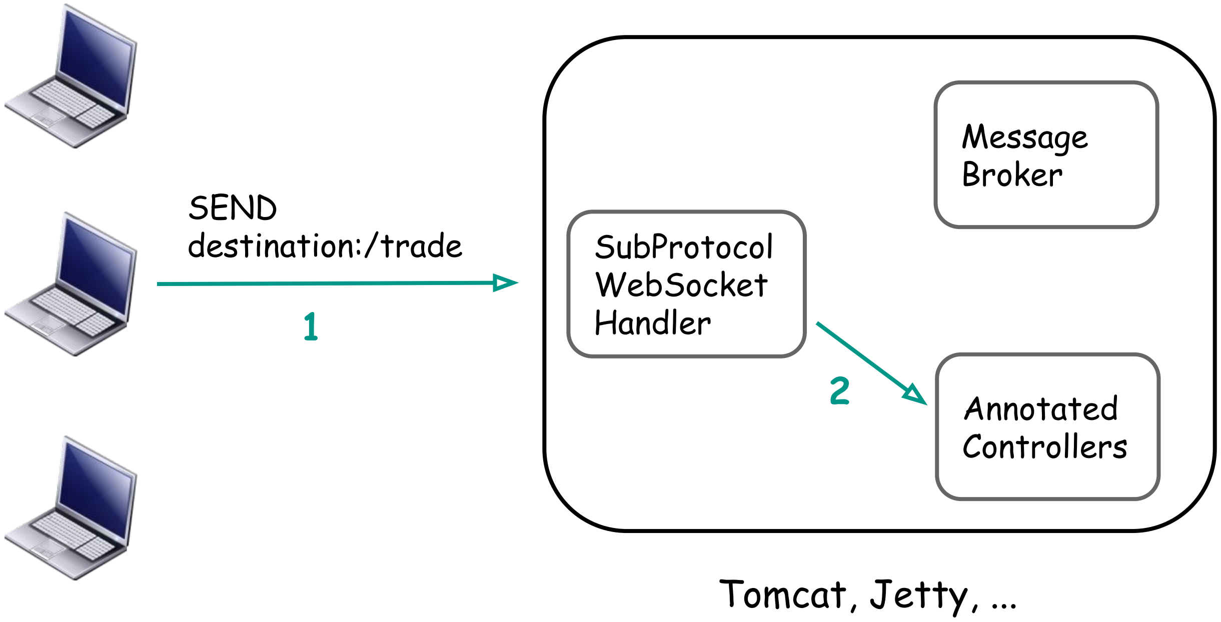
STOMP Sub-Protocol

- Simple, application-level, messaging protocol
- Built-in or external message broker
- Broadcast to subscribers

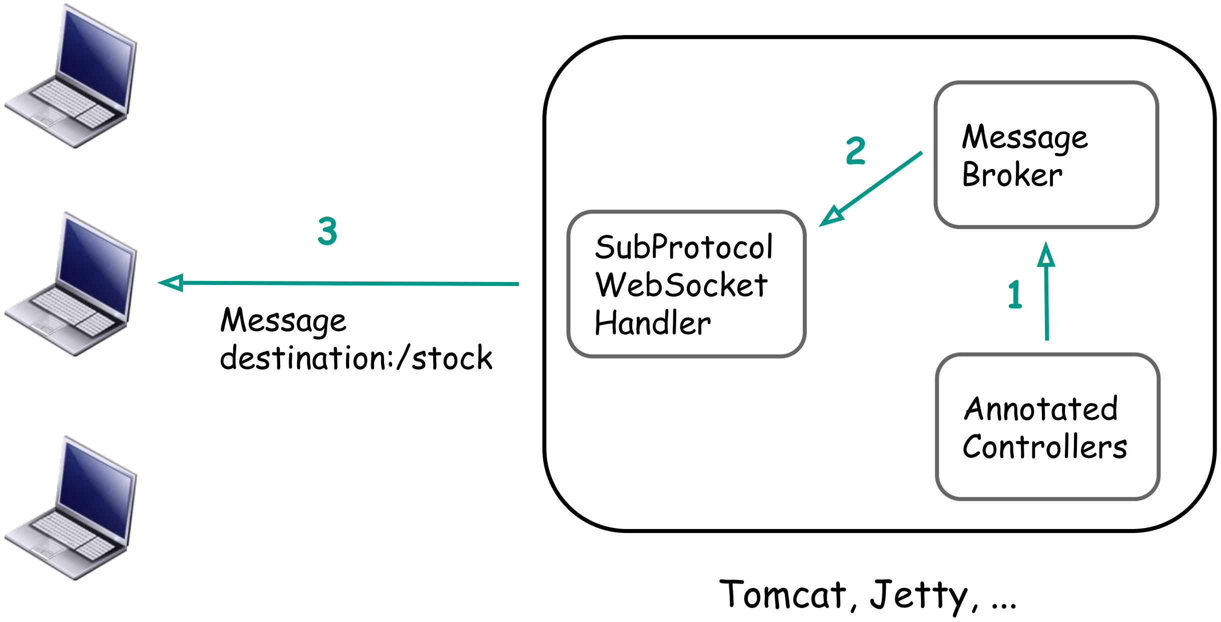
Client Subscribes



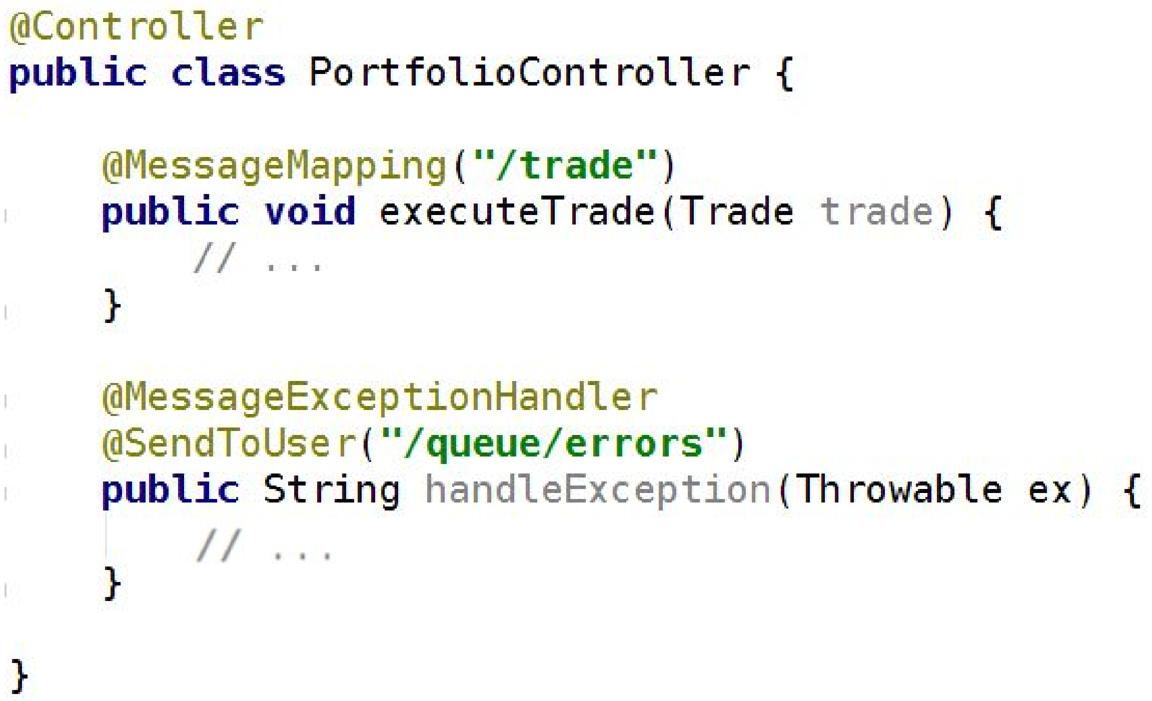
Client Sends Message



Broadcast To Subscribers



Controller Programming Model



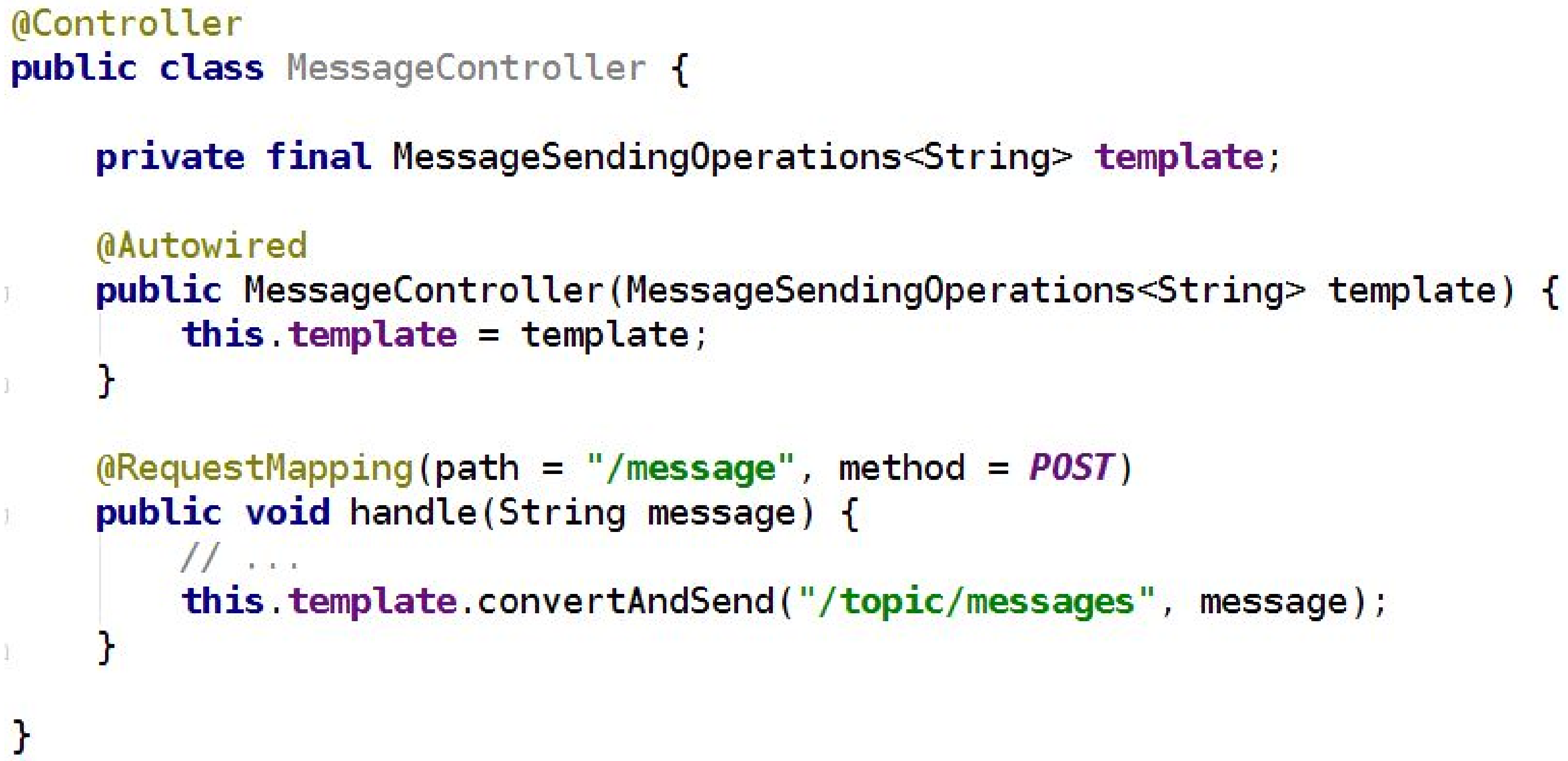
```java
@Controller
public class PortfolioController {

    @MessageMapping("/trade")
    public void executeTrade(Trade trade) {
        // ...
    }

    @MessageExceptionHandler
    @SendToUser("/queue/errors")
    public String handleException(Throwable ex) {
        // ...
    }

}
```

Broadcast To Subscribers



```java
@Controller
public class MessageController {

    private final MessageSendingOperations<String> template;

    @Autowired
    public MessageController(MessageSendingOperations<String> template) {
        this.template = template;
    }

    @RequestMapping(path = "/message", method = POST)
    public void handle(String message) {
        // ...
        this.template.convertAndSend("/topic/messages", message);
    }

}
```

SockJS Fallback

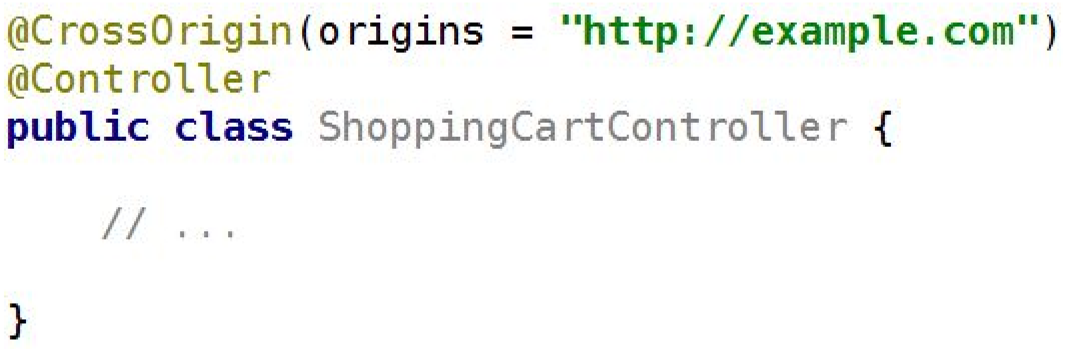
- For older browsers and restrictive proxies
- WebSocket emulation over HTTP
- Completely transparent to programming model

Programming Model

Cross-Origin Resource Sharing

- Built-in CORS
- Handler mappings understand pre-flight requests
- Layered config, global + fine-grained

Fine-grained Config

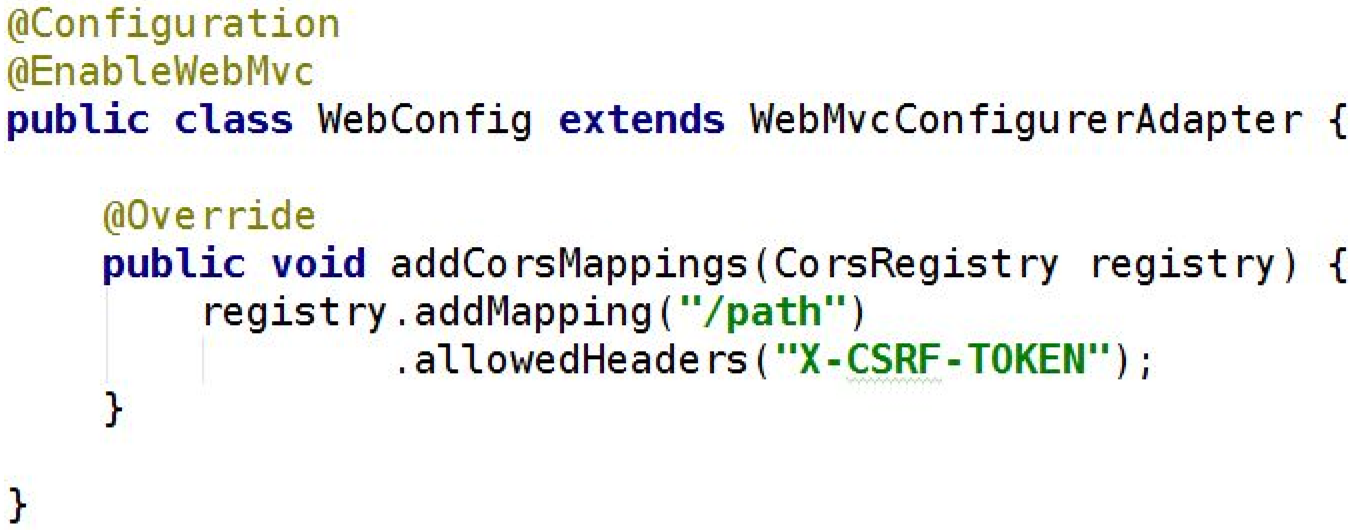


```java
@CrossOrigin(origins = "http://example.com")
@Controller
public class ShoppingCartController {

    // ...

}
```

Global Config



```java
@Configuration
@EnableWebMvc
public class WebConfig extends WebMvcConfigurerAdapter {

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/path")
                .allowedHeaders("X-CSRF-TOKEN");
    }

}
```

CorsFilter

- Same underlying CORS support but packaged as a Filter
- Analogous to similar filters from Tomcat/Jetty
- Alternative to @CrossOrigin

HTTP OPTIONS (v4.3)

- Built-in support
- Determine HTTP methods from @RequestMapping
- Response with `Allow` header

HTTP HEAD (v4.3)

- HEAD is mapped implicitly where GET is mapped
- Request handled like GET
- Response with headers only, including `Content-Length`

HTTP Caching

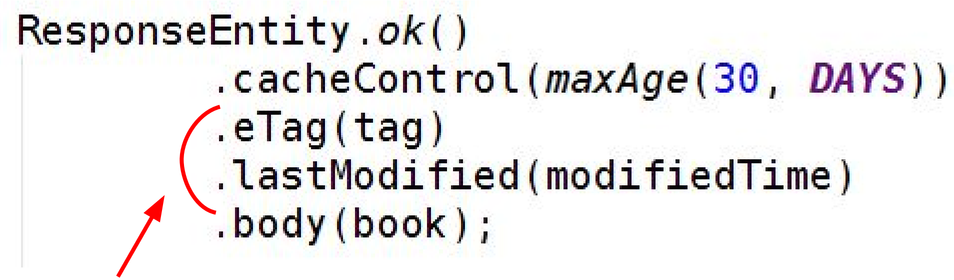
- CacheControl response header builder
- Prescriptive for common scenarios
- Support for all directives

Examples

CacheControl.**maxAge**(1, HOURS)	*Cache for 1 hour*
CacheControl.**noStore**()	*Prevent caching*
CacheControl.**noCache**()	*Conditional cache with eTag/lastModified*

CacheControl.maxAge(1, HOURS)
    .noTransform().cachePublic()
	Advanced

ResponseEntity Example



```
ResponseEntity.ok()
        .cacheControl(maxAge(30, DAYS))
        .eTag(tag)
        .lastModified(modifiedTime)
        .body(book);
```

Request is checked for eTag/lastModified
and 304 sent if match

CacheControl In Configuration

- WebContentInterceptor

- ResourceHttpRequestHandler

- Exposed in MVC Java config & XML namespace

Composed Annotations

- Custom `@RequestMapping` since 4.2
- Built-in shortcut annotations in 4.3
- `@GetMapping`, `@PostMapping`, `@DeleteMapping`, …

New and noteworthy

- @RequestMapping(path=*”/path”*)
- @SessionAttribute, @RequestAttribute (@since 4.3)
- @ModelAttribute(binding=*false*) (@since 4.3)
- @RestControllerAdvice (@since 4.3)

HTTP Client

- AsyncRestTemplate
- RestTemplate with Netty and OkHttp client
- Non-blocking, reactive HTTP client in Spring Framework 5
  - work in progress under spring-reactive

View Layer

Static Resource Handling

- Not merely serving js, css, images, …
- Flexible two-way resolution between URLs and resources
- Transformation of resources

Content Version URL

“/main-7fbe76cdac.css” → “/main.css”

Hash calculated
from content

Fixed Version URL

“/fe61cb/js/app.js”→ “/js/app.js”

Reduced SHA, version name,
release date, etc.

Working with Static Resources

- Optimize (minify, concatenate)
- Transform (sass, less)
- Use CDN
- Develop and debug (F5)
- Source code layout (separate client module)

Support Diverse Approaches

- JavaScript build plugins (gulp, grunt)
- WebJars
- Full asset pipeline (e.g. WUIC)

Script Templating

- JSR-223 scripting engines
- Nashorn JavaScript engine JDK 8
- Server-side rendering with React JavaScript

Groovy Markup Templating

- ViewResolver for Groovy markup template engine
- DRY markup option
- Groovy DSL

Additional Resources

- spring-reactive project
- "Resource Handling in Spring MVC 4.1" video
- "Resource Handling" workshop

Questions

@rstoya05

We are hiring!

http://pivotal.io/careers