Spring 4.x Web Application Overview

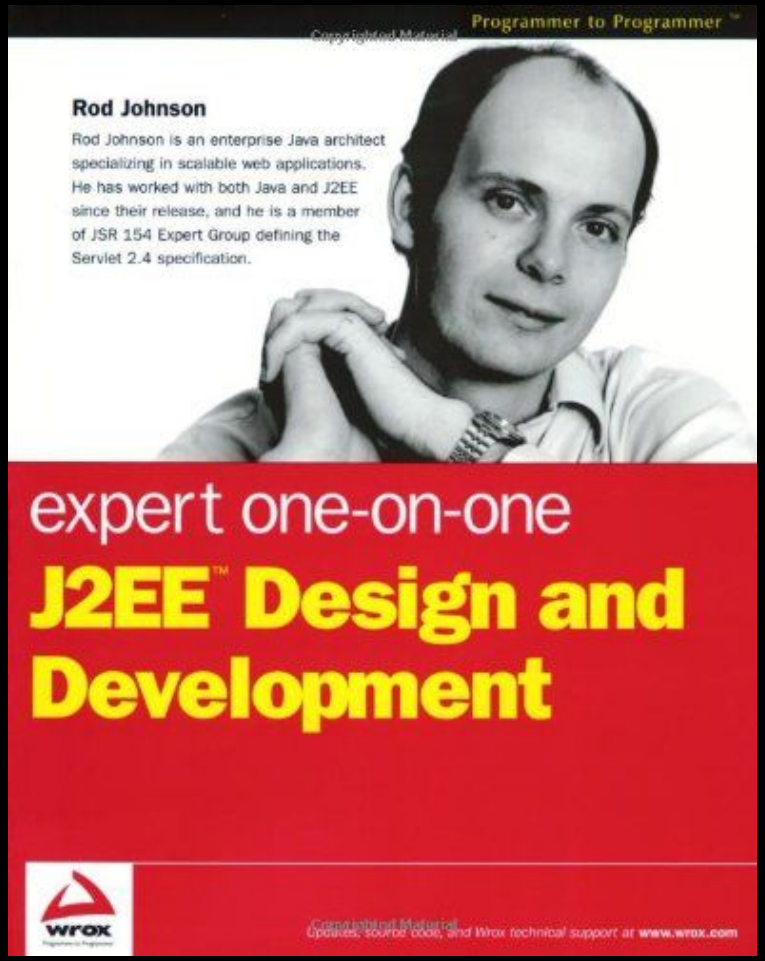
최신 웹 트렌드를 따르는 스프링 더듬기

발표자 소개

- 이름 : 김지헌 (== honeymon)
  - ihoneymon@gmail.com
  - http://honeymon.io
  - 이노트리 (http://innotree.com) 근무
  - KSUG 일꾼단 소속 잡무 수행
  - 개발자 …
  - 종합레저스포츠인을 표방

Spring framework

- 2003, JDK 1.4
  - https://spring.io/blog/2004/03/24/spring-framework-1-0-final-released
- Annotation, REST, SPAs, WebSocket 가 요구되기 전



Programming Model

Programming Model	Version
@Controller	2.5(2007)
REST	3.0(2009)
Async HTTP	3.2(2012)
WebSocket	4.0(2013)

스프링 프레임워크의
성공포인트

간결하고, 깔끔하고, 안정적인 설계

확장의 자유로움

HTTP, REST, Messaging
등 다양한 지원

강력한 커뮤니티!!

http://ksug.org

변화하는 환경에 따라 끊임없이 진화!


WHY

@Controller method

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

@RestController

```
@Controller
public class MyController {

    @RequestMapping @ResponseBody
    public Foo handleFoo() {
    }

    @RequestMapping @ResponseBody
    public Boo handleBoo() {
    }
}
```

```
@RestController
public class MyController {

    @RequestMapping
    public Foo handleFoo() {
    }

    @RequestMapping
    public Boo handleBoo() {
    }
}
```

@RestController = @Controller + @ResponseBody

@ControllerAdvice

- InitBinder
- ModelAttribute
- ExceptionHandler

@ControllerAdvice

- @ControllerAdvice
- @ControllerAdvice(annotations=RestController.class)
- @ControllerAdvice(basePackages="org.ksug")
- @ControllerAdvice(assignableTypes=
  {MyController.class})

ResponseEntityExceptionHandler - v3.2

- @ControllerAdvice를 함께 사용하는 예외 처리 추상클래스
- Spring MVC 예외처리
- REST API 친화적임

ResponseBodyAdvice

- @ControllerAdvice와 함께 사용하는 인터페이스
- v4.1에 추가됨
- 컨트롤러에서 반환된 응답결과를 HttpMessageConverter가 다루기 전에 작업

RequestBodyAdvice + v4.2

Jackson 이야기

- Jackson: JSON & XML 조작
- Jackson2ObjectMapperBuilder 를 이용해서 ObjectMapper 인스턴스를 생성
- LastestJackon integration improvements in spring
  - @JsonView: Jackson's Serialization View
  - JSONP: @ResponseEntity & ResponseEntity

@RequestMapping

- java.util.Optional(JDK 8) 지원
- ResponseEntity/RequestEntity buider-style method
- MvcUriComponentBuilder를 사용하여 다른 컨트롤러의 메서드를 호출할 수 있는 URI 생성
- @ModelAttribute 사이의 의존성 부여

정적자원 지원

- ResourceHttpRequesthandler < 4.1
- ResourceResolver, ResourceTransformer, ResourceUrlProvider 4.1+
- 컨텐츠 기반 해시를 가진 Versioned URL
  - e.g. "/css/font-awesome.min-7fbe76cdac.css"
- 보다 자세한 사항은!!
  - Resource Handling Spring MVC 4.1 - SpringOne 2014
  - Resource handling Spring MVC - Adieu 2014, 봄싹 (박용권)

Groovy markup template

- HTML 템플릿용 DSL을 Groovy 로 컴파일
- Groovy 의 강력한 힘으로!
- See GroovyMarkup View

MVC 설정

- ViewResolver 등록 지원
- ViewController 설정
- Path matching 설정

비동기 HTTP 요청

- Servlet 3.0+ Async Request
- 클라이언트에 이벤트 푸시, 긴 연산
- 상당히 쉽게 할 수 있다.
- 조금 더 복잡한 상황에 대해서는 구현이 어려움

Web Messaging Architecture

- WebSocket -> Low-level transport
- SockJS -> fallback options
- STOMP -> application-level protocol

STOMP Frame

```
>>> SEND
destination:/app/hello
content-length:19

{"name":"honeymon"}
stomp.js:130

<<< MESSAGE
destination:/topic/greetings
content-type:application/json;charset=UTF-8
subscription:sub-0
message-id:9n_v3m2s-0
content-length:30

{"content":"Hello, honeymon!"}
```

Handle Messages from WebSocket clients

```
@Controller
public class PortfolioController {
    @MessageMapping("/greetings")
    public void add(String payload) {
        // ...
    }
}
```

Broadcast Messages to WebSocket Clients

```
@Autowired
private SimpMessagingTemplate messagingTemplate;

@RequestMapping(value = "/boardcast-echo", method = RequestMethod.POST)
public void broadcastEcho() {
    this.messagingTemplate.convertAndSend("/app/stomp/echo", "Hello, KSUG");
}
```

Spring Framework 4.2

HTTP Streaming

- New return type ResponseBodyEmitter
- Write a series of Objects to response body
- via HttpMessageConverter
- Also works with ResponseEntity

HTTP Streaming Example

```
@RequestMapping
public ResponseBodyEmitter handle() {
        ResponseBodyEmitter emitter = new ResponseBodyEmitter();
        // ...
        return emitter;
}

// Later from some other thread
emitter.send(new Foo());
emitter.send(new Bar());
emitter.complete()
```

HTTP Streaming: SSE-style

- SseEmitter extends ResponseBodyEmitter
- ResponseBodyEmitter와 같은 동작을 하지만 다른 점 하나!
  - SSE-style: W3C Server-Send Event specifications

Server-sent events example

```
@RequestMapping
public SseEmitter handle() {
        SseEmitter emitter = new SseEmitter();
        // ...
        return emitter;
}
// Later from some other thread
emitter.send(event().name("foo").data(new Foo()));
emitter.send(event().name("bar").data(new Bar()));
emitter.complete();
```

HTTP Streaming: Directly to Response

```
@RequestMapping
public StreamingResponseBody handle() {
    return new StreamingResponseBody() {
        @Override
        public void writeTo(OutputStream out) {
        // ...
        }
    }
}
```

HTTP caching update

- CacheControl builder
- Configurable in various places
  - WebContentGenerator/Interceptor
  - ResourceHttpRequestHandler
  - ResponseEntity
- Improved ETag/Last-Modified support in WebRequest

CORS support

- 계층적인 접근
- 세밀한 설정
  - @CrossOrigin 컨트롤러에 사용
- URL 패턴을 기반으로 한 전역설정

How CORS is supported

- AbstractHandlerMapping 에 의해 생성
- 하위 클래스에 대한 CORS 메타데이터 제공
  - @CrossOrigin
  - CorsConfigurationSource
- AbstractHandlerMapping 안에 CorsProcessor 설정
  - CORS 체크 수행, 응답헤더 설정

@RequestMapping meta annotation

```
@RequestMapping(method = RequestMethod.POST, produces =
    MediaType.APPLICATION_JSON_VALUE, consumes =
    MediaType.APPLICATION_JSON_VALUE)
public @interface PostJson {
   String value() default "";
}
//..
@PostJson("/account")
 public ResponseEntity<Account> add(Account account) {
    return ResponseEntity.ok(account);
 }
```

Javascript View Templating

- Server and Client-side rendering
  - using same template engine(e.g. React)
- `ScriptTemplateView`, `ViewResolver`
  - built on Nashorn(JDK 1.8)
- 서버에서 초기화해두고, 서버에서 가져온 코드를 가지고 클라이언트 사이드에서 랜더링

다음세션!! + 정성용 Isomorphic!!

MISC == 기타등등

- 정적자원에 대한 byte-range 요청 지원
- @ExceptionHandler 메서드에 메서드 인자로 HandlerMethod 사용
- MvcUriComponentsBuilder custom baseUri

STOMP/WebSocket

- @ControllerAdvice 기반으로 메서드 전역에 @Exceptionhandler 적용가능
- @SendTo/SendToUser Destination value placehodler 정의 : @DestinationVariable
- 다중앱 서버사이에서 사용자 목적지 사용
- SockJS 내에서 CORS 향상

Simple Broker(STOMP over WebSocket)

- HeartBeat support
- SpEL-based message selector

```
SUBSCRIBE
destination:/app/greetings
selector:headers.foo == 'bar'
```

STOMP client

- 자바를 사용하여 WebSocket 혹은 TCP 접근
- 테스트에 유용함
- 서버사이드에서 STOMP broker 접근

정리

- 설정의 응집도를 높이고!
- 세부설정은 진입지점에서도 가능!
- 웹소켓의 기능을 HTTP에서도!
- 조금 더! 빠르게! 쉽게!


WHY

발표자료

- 예제 :
  - https://github.com/ihoneymon/rocking-the-spring-4x-web-application

참조자료

- Spring Framework - wikipedia
- 4.0.0 release
- Spring 4 Web applications
- Spring Framework 4.0 Web Applications - SpringOne 2014
- Overview of Spring 4.0 - 박용권 , KSUG
- Spring MVC 4.2: New and Noteworthy
- Workshop - From Spring 4.0 To 4.2
- Exception handling in Spring MVC
- Spring - Modify response headers after controller processing
- Using WebSocket to build an interactive web application
- CORS support in Spring Framework
- Isomorphic templating with Spring Boot, Nashorn and React
- Script template
- HTTP Streaming

KSUG 일꾼단 모집!

동료가 돼라!

끝!!

궁금하신 것이 있으시다면,

ihoneymon@gmail.com
@ihoneymon