

# Spring Boot for a hoot!

## SPA Day 2018, Ken Rimple

---

# SPAs and the Enterprise

# Java and JVM Languages on the Backend

---

- Much of the enterprise runs on Java
- Spring is an enabling technology for Java integration
- Many Spring apps do Ajax now, they are ready for integration with an SPA

# Spring Features for SPA

---

- Spring Boot for standing up a Spring server
- Spring MVC REST Controllers
- Spring Security for SPA

# Spring Boot

---

- Quickest way to stand up a web server in Java
- Can add
  - Spring Web MVC
  - Spring Data (for database API integration)
  - Spring Security and SPA security with Tokens
  - And more...

# Spring MVC REST

```
1  import org.springframework.boot.SpringApplication;
2  import org.springframework.boot.autoconfigure.SpringBootApplication;
3  import org.springframework.boot.autoconfigure.domain.EntityScan;
4  import org.springframework.context.annotation.ComponentScan;
5  import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
6
7  @SpringBootApplication
8  public class SpadayApplication {
9      private static final Logger LOGGER =
10         Logger.getLogger(SpadayApplication.class.getName());
11
12     public static void main(String[] args) {
13         SpringApplication.run(SpadayApplication.class, args);
14     }
15 }
```

# Spring REST Controller - fetching a collection

```
1  @RestController
2  @RequestMapping(path = "/api")
3  public class ApiController {
4      @GetMapping(path = "/session")
5      public ResponseEntity<List<Session>> getSessions() {
6          List<Session> sessions = apiService.getSessions();
7          return ResponseEntity.ok(sessions);
8      }
9      ...
10 }
```

## Spring REST Controller - posting a new entry

```
1 @PostMapping(path = "/session/{sessionId}/subscribe")
2 public ResponseEntity<Registration> registerForSession(
3     @PathVariable("sessionId") Long sessionId,
4     @RequestBody Registration registration) {
5
6     registration.setSessionId(sessionId);
7     return ResponseEntity.ok(apiService.saveRegistration(reg
8 }
```

- **@PostMapping** maps this method to a POST call with the given path
- **@PathVariable** accesses the **sessionId** mapping variable
- **@RequestBody** captures the incoming payload and converts it to Java

💡 **ResponseEntity** allows use to return an HTTP response with a payload or error message



# Spring and Security

---

- Spring's Security API
  - Powerful server-side authentication and authorization library
  - Highly configurable, but recommend finding an expert to guide your team

# Authentication

---

- Spring supports pluggable authentication
  - Usernames and passwords
  - JSON Web Tokens (for SPA applications)

# Authorization

---

- Supports pluggable authorization
  - hard-coded roles for simple testing
  - database-driven roles
  - logic-driven roles

# Spring is a Server-Side monster

---

- Handles connection pooling and datasources
- Integrates with messaging systems
- Provides monitoring via JMX
- Powerful data abstraction APIs for RDBMS, NoSQL, etc.
- Spring just needs a web server (Tomcat, Jetty) and not an expensive app server

## General App Hosting Options

---

- **Proxied Backend** - Your web server can proxy all requests to a given path to the Spring Boot server (simplifies the client configuration)
- **CORS** - Your web server asserts which network servers and methods / paths can be called
- Spring can host content as well, but **nginx** or **apache** are usually better choices