\o/

Brasília

Sou mulher, sou dev, sou java

Meetup

Mistérios do Spring Framework

Angélica Leite

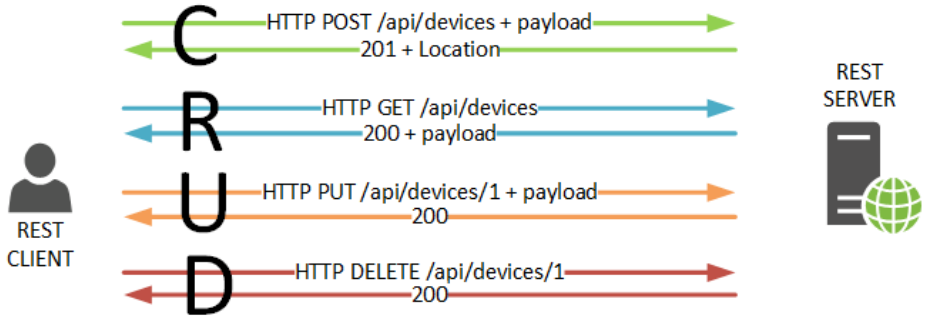
@angelicalleite

Temas Abordados

- Arquitetura Rest / Protocolo HTTP
- Spring Framework
- Java
- Dicas

Arquitetura Rest

- Por que estudar REST?
- Orientação e Padronização
- Base Protocolo HTTP
- Stateless
- Tá na **HYPE** \o/ /o\



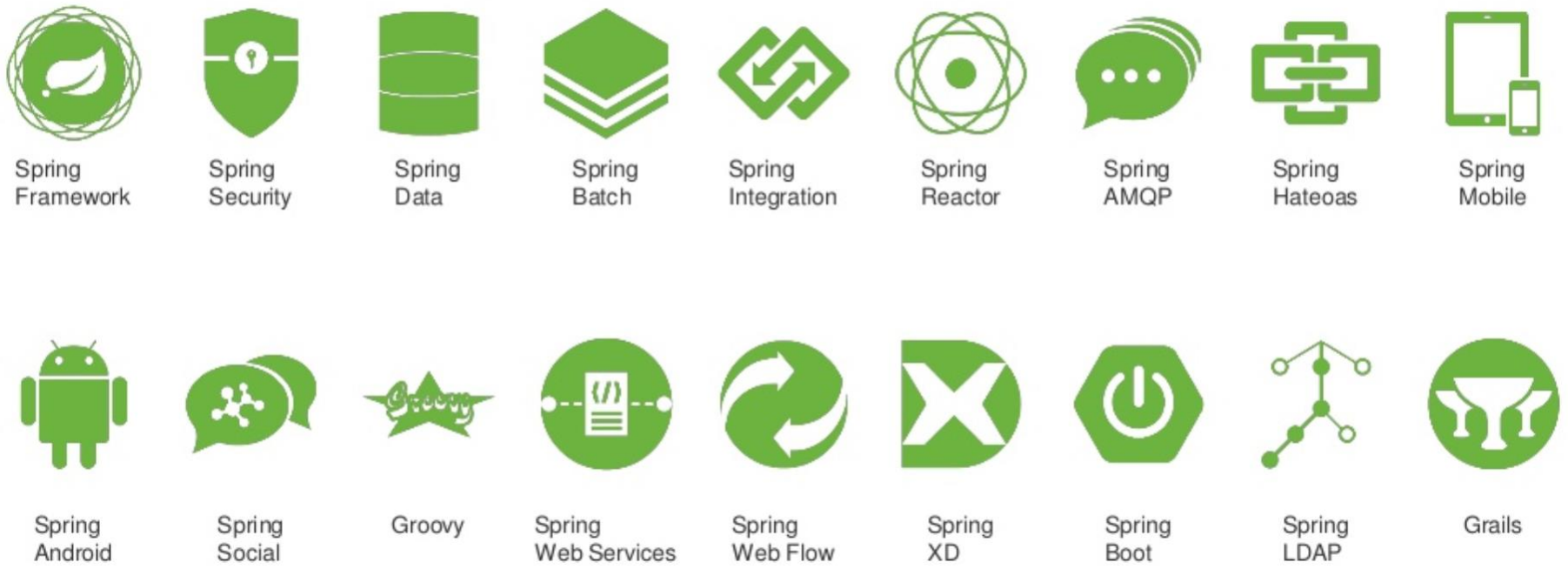
Protocolo HTTP

- Principal protocolo de comunicação web
- Request/Response cliente servidor
- **Composição:** header, payload, versão, agente, path, status code, verbos, propriedade.

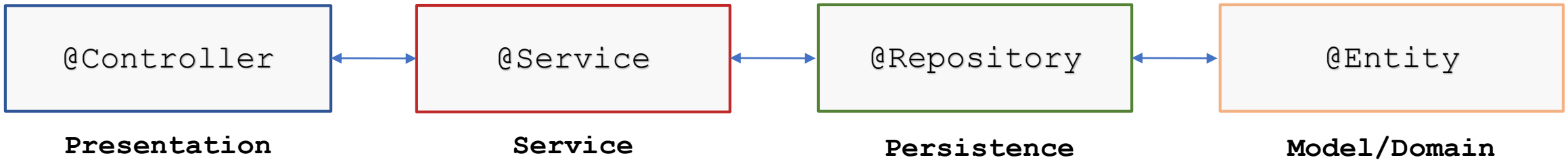
Spring Framework

- O que é Spring
- Integração
- Comunidade Ativa
- Java linguagem base
- Modularidade e Ecossistemas

Ecossistema Spring



Fluxo Base Camadas



Request

- Criação recursos e end-points
- Camada de controles gerência de rotas

Anotação	Verbo
@PostMapping	POST
@GetMapping	GET
@PutMapping, @PatchMapping	PUT,PATCH
@DeleteMapping	DELETE
@RequestMapping	

- Uso boas praticas arquitetura REST: nome rotas, uso verbos corretos e cabeçalhos http.

Response

- **Tratamento Responses**
- Formato: **Content-Type**: json, xml, text, html

Faixa	Grupo	Descrição
1xx	Informação	100 - Continue
2xx	Sucesso	202 - Accepted
3xx	Redirecionamento	302 - Found
4xx	Cliente Error	404 - Not Found
5xx	Servidor Error	502 - Bad Gatewayded

ResponseEntity destina-se a representar toda a resposta HTTP, permitindo o controle de: código de status, cabeçalhos e corpo.

Response

- **Tratamento Responses**
- Anotações **@ResponseBody** e **@ResponseStatus**

Tipo	**Retorno Flexível**	**Tratamento Status Code**
Entidade **[String, Long, Pessoa]**	**Não**	**Não**
Genérico **[Object]**	**Sim**	**Não**
ResponseEntity<Tipo>	**Não**	**Sim**
ResponseEntity<?>	**Sim**	**Sim**

ResponseEntity destina-se a representar toda a resposta HTTP, permitindo o controle de: código de status, cabeçalhos e corpo.

Injeção de Dependência

- Anotações

  @Autoriwed, **@Component**, **@Controller**, **@Service**, **@Repository**

- Anotações propriedade, construtor ou métodos
- Inversão de Controle

Manipulação Exceções

- **@RestControllerAdvice** permite escrever código global aplicado aos controladores.
- **@ExceptionHandler** permite que você defina um método que, como o nome sugere, manipule exceções.

Manipulação de Exceções

Exemplo

```
@RestControllerAdvice
public class GlobalExceptionHandler implements ErrorExceptionHandler, Loggable {

    @ExceptionHandler(BadCredentialsException.class)
    public ResponseEntity handleBadCredentials(final BadCredentialsException ex, HttpServletRequest http) {
        loggerError(ex);

        return unauthorized(message(401, 3000, http.getRequestURI(), ex.getMessage()));
    }

    @ExceptionHandler(AccessDeniedException.class)
    public ResponseEntity handleAccessDenied(AccessDeniedException ex, HttpServletRequest http) {
        loggerError(ex);

        return unauthorized(message(401, 3001, http.getRequestURI(), ex.getMessage()));
    }
}
```

Mensagem Error

Error Response + @RestControllerAdvice + @ExceptionHandler

```
public interface ErrorExceptionHandler extends RestResponse {

    default ErrorResponse message(int status, int code, String path,
                                  String message, List<String> errors) {

        return new ErrorResponse(code, path, message, status, errors);
    }

    default ErrorResponse message(int status, int code, String path,
                                  String message, String... errors) {

        return new ErrorResponse(code, path, message, status, Arrays.asList(errors));
    }
}
```

Métodos Default

- **Tirando proveito Java 8**
- **Uso métodos default**
- Mapeamento status code response
- Código intuitivo e semântico

Métodos Default

Exemplo Interface RestResponse

```
public interface RestResponse {

    default ResponseEntity<Void> ok() {
        return ResponseEntity.ok().build();
    }

    default ResponseEntity<Void> notFound() {
        return ResponseEntity.notFound().build();
    }

    default <B> ResponseEntity<B> ok(B body) {
        return response(200, body);
    }
}
```

Métodos Default

Exemplo uso interface RestResponse

```
@GetMapping("/{id}")
public ResponseEntity<?> method() {

    Project project = projectService.getProjectRepository().findOne(1L);

    return project != null ? ResponseEntity.ok().body(project)
                           : ResponseEntity.ok().body("Não foi possível retornar");
}
```

Métodos Default

Exemplo uso interface RestResponse

```
@GetMapping("/{id}")
public ResponseEntity<?> method() implements RestResponse{

    Project project = projectService.getProjectRepository().findOne(1L);

    return project != null ? ok(project) : badResquet("Não foi possível retornar");
}
```

Exception Handler

@RestControllerAdvice + @ExceptionHandler

```
@RestControllerAdvice
public class GlobalExceptionHandler implements ErrorExceptionHandler, Loggable {

    @ExceptionHandler(BadCredentialsException.class)
    public ResponseEntity handleBadCredentials(BadCredentialsException ex, HttpServletRequest http) {

        loggerError(ex);
        return unauthorized(message(401, 3000, http.getRequestURI(), ex.getMessage()));
    }

    @ExceptionHandler(AccessDeniedException.class)
    public ResponseEntity handleAccessDenied(AccessDeniedException ex, HttpServletRequest http) {

        loggerError(ex);
        return unauthorized(message(401, 3001, http.getRequestURI(), ex.getMessage()));
    }
}
```

Dicas ;)

Maven

- Gerência e centralização de dependência
- Ciclo de vida
- Plugins e Archetype
- Modularização
- Deploy

Lombok

- Amenizar código verboso
- Evita criar **construtores, getter, setter, equals, hashCode, toString.**
- Disponibiliza padrão builder.

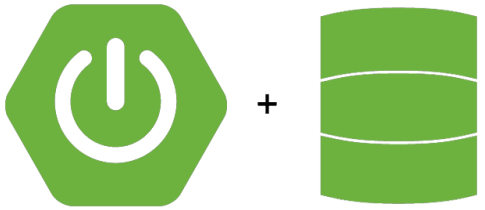
```
@Data
public class Entity {

    @Id
    @GeneratedValue
    private Long id;
    private String prop1;
    private String prop2;
    private String prop3;

}
```

Spring Security

- Melhor camada de segurança
- Autenticação via memoria, banco, OAuth e JWT
- Classes pré prontas para lidar com propriedades de usuário e permissões.
- Simplicidade manipular autenticação das rotas

Spring Data

- **@Repository**
- Principais métodos manipulação implícitos
- **Interfaces:** Mongo, Crud, JPA, Neo4j, **PagingAndSorting**
- DSLs consultas por métodos, query nativas ou hsql
- Integração diversas arquitetura e tipos de banco



Actuator

- Disponibiliza métricas sobre aplicação: **end-points, logs, trace, health**
- Fácil integração

API Manager

- Spring Cloud + Netflix OSS: **Zuul**, **Hystrix**, **Ribbon**, **Feign**.
- Ferramentas de integração para API Manager
- Monitoramento, desempenho e segurança

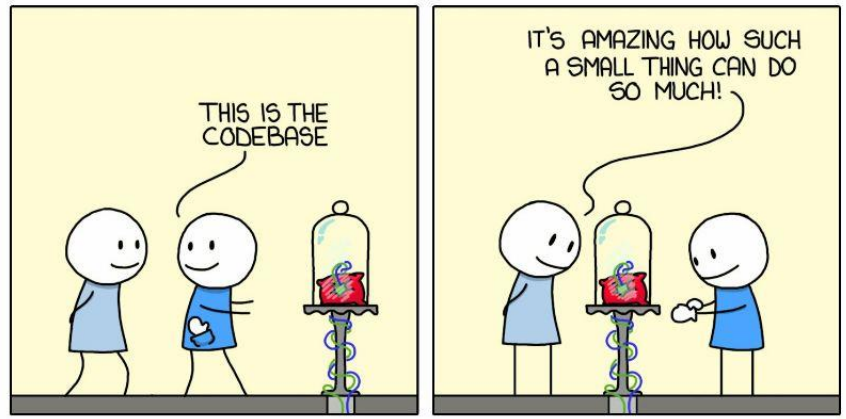
Swagger e Redoc

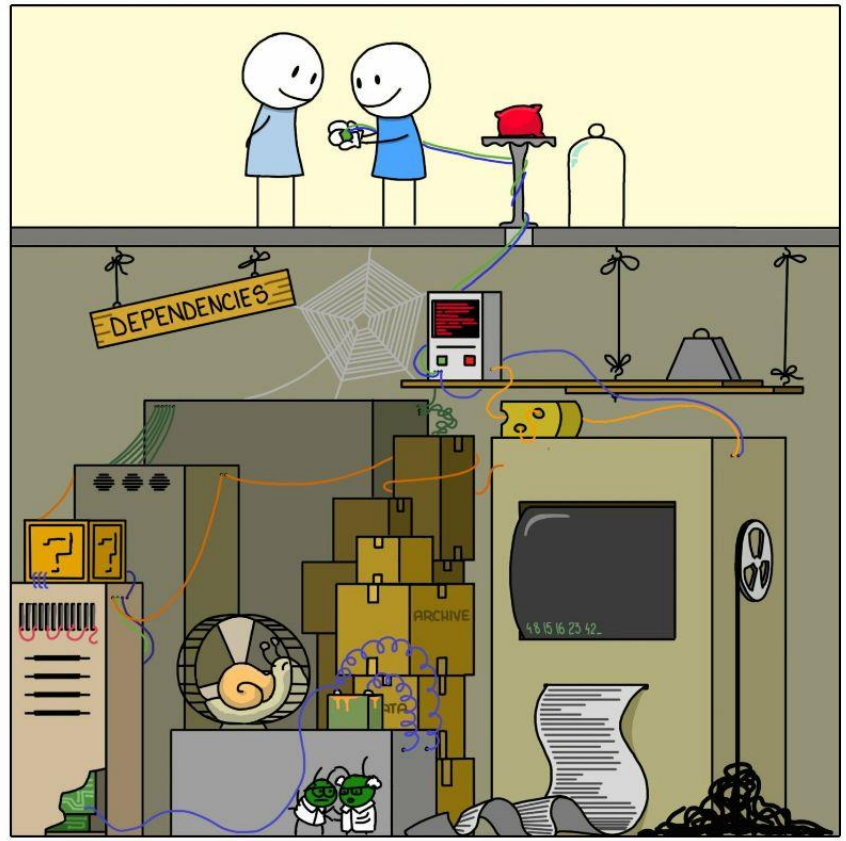
- Documentação recursos e end-points
- Padrão OpenAPI Specification
- Flexibilidade de ferramentas para geração aplicação: **editor e anotações.**

Resumo

- Protocolo HTTP
- Arquitetura Rest
- Compreensão camadas e fluxo Spring
- Tratamento de exceções e erros
- Orientação e padrões arquitetura rest
- Código semântico e intuitivo
- Integração outros recursos

IMPLEMENTATION





MONKEYUSER.COM

Exemplo

Arquitetura Proposta

Sou mulher, sou dev, sou java

Obrigada ;)