Desenvolvimento de APIs com

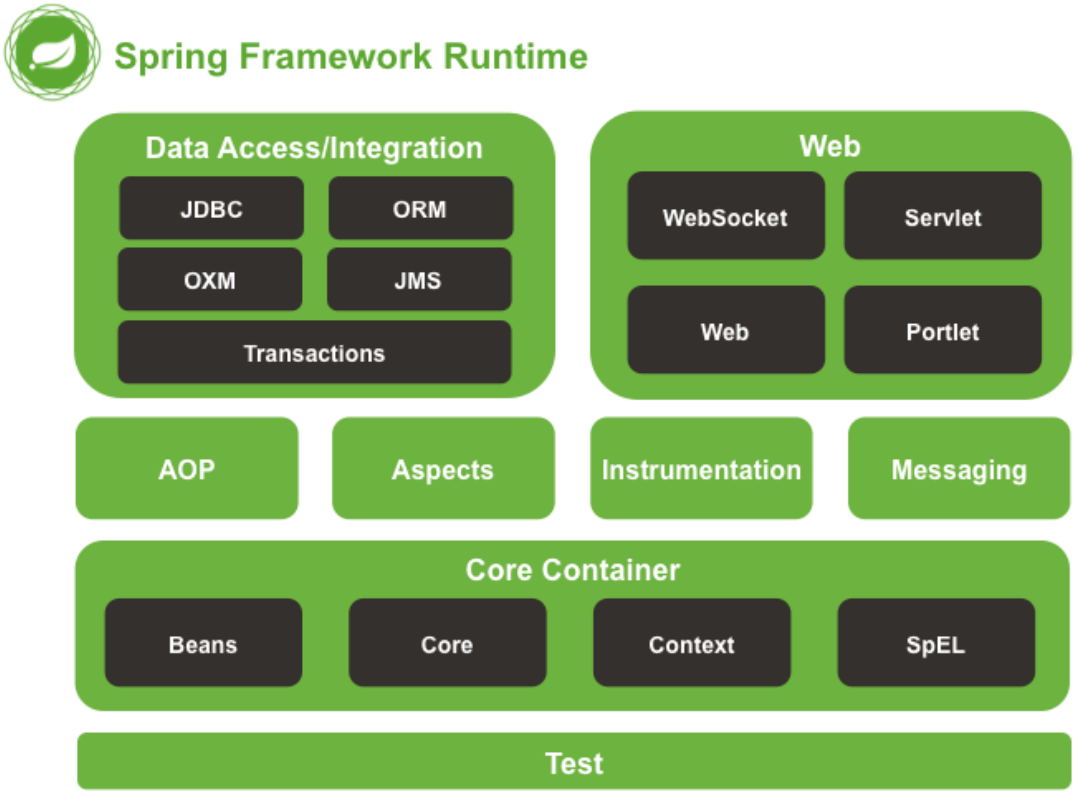
{
"name" : "Luiz Roberto Freitas",
"email" : "luizrobertofreitas@gmail.com",
"linkedin" : "@luizrobertofreitas",
"twitter" : "@luizrobertojr",
"role" : "Developer"
}

Tópicos

- Spring Framework
- Spring Beans/Components
- API’s REST
- Referências
- Show me the code!

Spring Framework

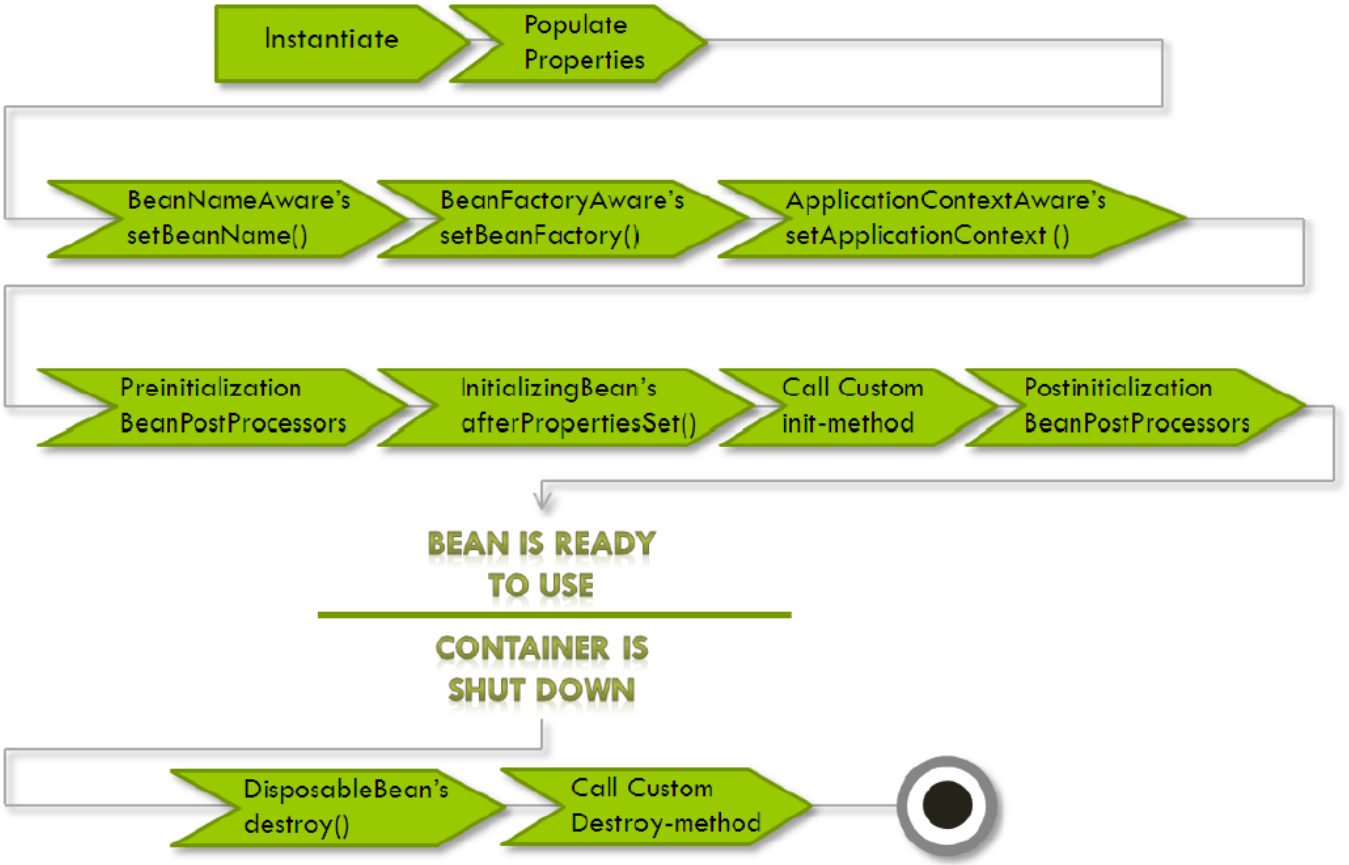
- Provê todos os aspectos de um servidor de aplicações… num container leve!
  - Transaction, Logging, AOP, Web, Caching, Persistency, NoSQL, Messageria, Batch, etc, etc, etc.
- Container IoC (inversion of control) e DI (Dependency Injection).
- Não depende de servidor de aplicações (Spring boot), caso dependa, é altamente portável.
- Pode plugar qualquer framework. Ex: persistência (JPA, iBatis, JDO, JDBC, etc).
- Componentes e Beans simples de entender.


Spring Framework Runtime
Data Access/Integration
JDBC
ORM
OXM
JMS
Transactions
Web
WebSocket
Servlet
Web
Portlet
AOP
Aspects
Instrumentation
Messaging
Core Container
Beans
Core
Context
SpEL
Test

Spring Beans/Components

- São classes comuns (POJO’s) gerenciados pelo Spring
- Permite a inversão de controle (IoC) e Injeção de dependências (DI)
- Possui um ciclo de vida bem definido
- O desenvolvedor não precisa se preocupar se há necessidade de reutilizar ou criar uma nova instância do Objeto
- Cada tipo de componente possui um tratamento específico dentro do container do Spring

Ciclo de vida de um Bean/Componente



Spring Beans/Components

- **@Component**: Mãe de todos os componentes
- **@Controller**: Utilizado para marcar classes da camada de controle (Interface API's, Controllers MVC, etc). Injeção de alguns componentes especiais (HttpRequest, HttpResponse, etc)
- **@Service**: Utilizado para marcar classes da camada de negócio.
- **@Repository**: Utilizado para marcar classes da camada de persistência. Tem tratamento especial das exceptions.

API’s REST

- **API (Application Programming Interface)**: é uma interface para um conjunto de instruções, procedures, rotinas.
- **REST (REpresentional State Transfer)**: Estilo arquitetural ou design para API’s
- **RESTfull**: Conjunto de regras para construir API’s REST, neste caso, utilizando protocolo HTTP
- **Protocolo HTTP**
  - **Recurso**: Objeto para transferência de estado. Ex: Arquivos, fotos, TXT, etc.
  - **Verbo**: Ação para representar o tipo de operação a ser realizada para determinado recurso no protocolo HTTP.
  - **Código de Status**: código informado na resposta da requisição HTTP.

API’s REST – Verbos HTTP

- Verbos HTTP utilizados em API’s: **GET, POST, PATCH, PUT, DELETE**.
  - **GET**: Recupera um recurso
  - **PUT**: Atualiza um recurso específico
  - **PATCH**: Atualiza partes de um recurso
  - **DELETE**: Exclui um recurso
  - **POST**: Cria um novo recurso
- Idempotência: várias chamadas com requisições identicas devem produzir o mesmo resultado: **GET, PUT, DELETE**

API’s REST – Status HTTP

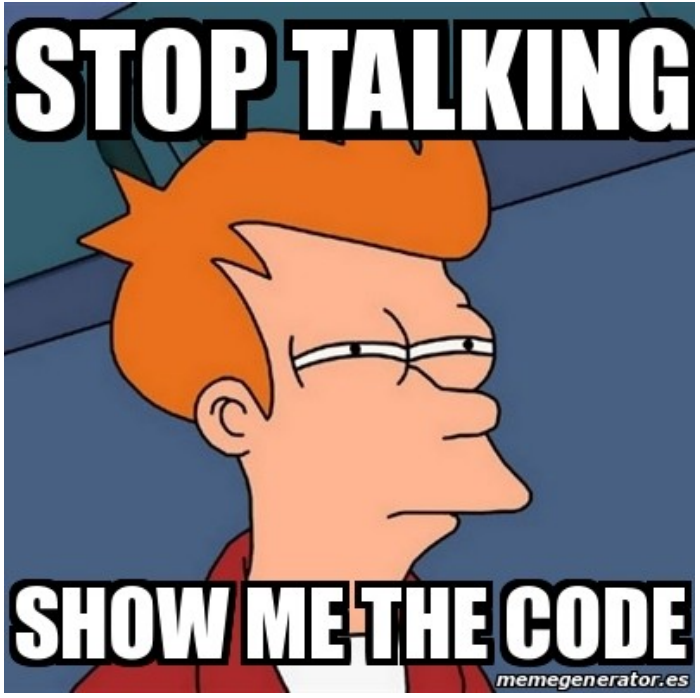
Status Codes HTTP:

- **1xx**: Informativas
- **2xx**: Sucesso
- **3xx**: Redirecionamento
- **4xx**: Erro do cliente
- **5xx**: Erro do servidor

- Veja + em https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status

Referências

- https://spring.io/guides/gs/rest-service/
- https://medium.com/extend/what-is-rest-a-simple-explanation-for-beginners-part-1-introduction-b4a072f8740f
- http://wildfly.org/
- https://en.wikipedia.org/wiki/Application_programming_interface
- https://restfulapi.net/http-methods/
- https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status
- https://dzone.com/articles/spring-data-jpa-1
- https://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html



https://github.com/luizrobertofreitas/usermgmt

Branches (Fooooooooork!!)

- **step-1**: Criação de controller + endpoint ping
- **step-2**: MVP completo. Fim de curso! :D
- **step-3**: Refactoring: Business Layer e Versionamento de endpoints
- **step-4**: Desconforto de expor entidades de BD via API? Lets add some domain models!!
- **step-5**: Business Layer não serve pra nada! Vamos fazer alguma coisa com ela? Validações!!
- **step-6**: Exceptions customizadas e tratamento de erros
- **step-7**: Swagger, um inicio para documentar API's

Indo além: Autenticação (Basic, Oauth), Tunning Tomcat/Jetty/Undertow, Async, Events, NoSQL, AMQ, Batch.


THANK YOU
GOOD SIR