SPRING BOOT

Edson Orivaldo Lessa Junior

edson.lessa@unisul.br

Spring Framework

- Objetivo de aumentar a produtividade na escrita de aplicações corporativas;
- Spring Framework Java criado com o objetivo de facilitar o desenvolvimento de aplicações
- Caracteriza-se pela Inversão de Controle e Injeção de Dependências
- Possui módulos para persistência de dados, integração, segurança, testes, desenvolvimento web
- Não precisa de um servidor de aplicação.
- Faz uso apenas da JVM;
- A plataforma J2EE e os EJBs propiciam a implementação de comportamentos que não eram necessários. Esse diferencial do Spring torna a arquitetura mais leve, fácil de compreender, manter e evoluir;
- A inversão de controle e injeção de dependência, fornecendo um container, que representa o núcleo do framework e que é responsável por criar e gerenciar os componentes da aplicação, os quais são comumente chamados de beans.

Spring Framework

- Spring Framework ainda é alvo de críticas a respeito do tempo necessário para se iniciar o desenvolvimento de novos projetos.
- A principal crítica feita ao Spring é sobre o modo como se configura o seu container de injeção de dependências e inversão de controle usando arquivos de configuração.
- Estes artefatos, conforme aumentam de tamanho, se tornam cada vez mais difíceis de serem mantidos, muitas vezes se transformando em um gargalo para a equipe de desenvolvimento.
- A complexidade na gestão de dependências é outro ponto que desfavorece o framework.
- Conforme os projetos precisam interagir com outras bibliotecas e frameworks como, por exemplo, bibliotecas de persistência, mensageria, frameworks de segurança e tantos outros, garantir que todas as bibliotecas estejam presentes no classpath da aplicação acaba se tornando um pesadelo.

Spring Framework

- Desta forma, foi introduzido no Spring 4.0 um novo projeto com o objetivo de responder a estas críticas e também mudar bastante a percepção acerca do desenvolvimento de aplicações para a plataforma Java EE. Este projeto é o Spring Boot.
- O projeto Spring Boot resolve estas questões e ainda nos apresenta um novo modelo de desenvolvimento, mais simples e direto, sem propor novas soluções para problemas já resolvidos, mas sim alavancando as tecnologias existentes presentes no ecossistema Spring de modo a aumentar significativamente a produtividade do desenvolvedor.

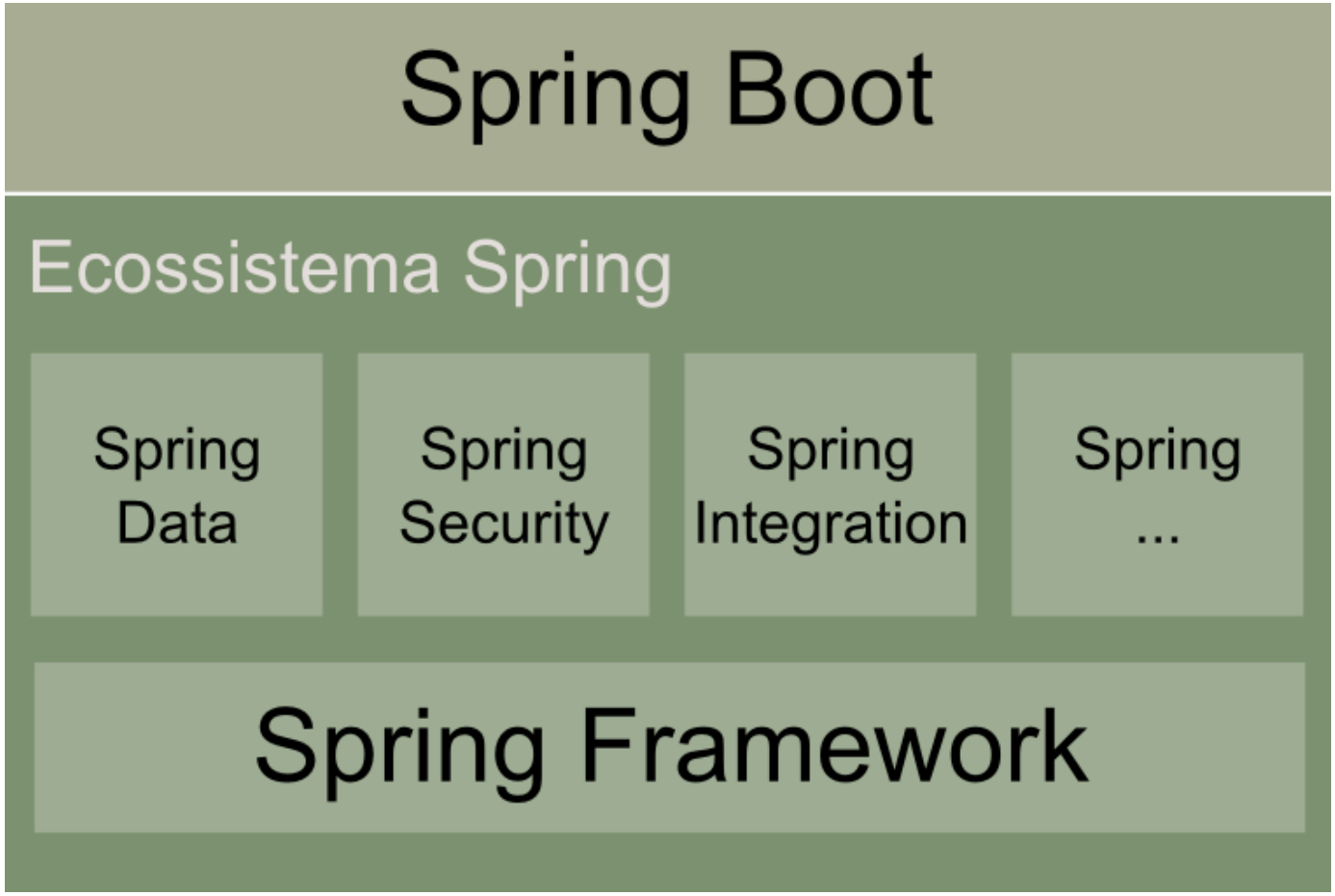
Conceitos fundamentais

- Inversão de Controle (IoC – Inversion of Control)
  - Delega a outro elemento o controle quanto a criação do objeto, chamada de métodos, entre outros
  - Controle da execução de alguns comportamentos passa a ser gerenciado pelo próprio elemento e não pelo programador.
  - No Spring, o elemento que controla e gerencia os comportamentos é o container.
- Injeção de Dependência (DI – Dependency Injection)
  - Classe deixa de se preocupar em como resolver as suas dependências
  - Mantem o foco apenas no uso dos recursos das dependências
  - Característica mais marcante é não necessitar utilizar o “new” para criar objetos por ele gerenciados.

Spring Boot

- É um framework baseado no Spring
- Também conhecido com micro framework
- Objetivo não é trazer novas soluções para problemas que já foram resolvidos, mas sim reaproveitar estas tecnologias e aumentar a produtividade do desenvolvedor.
- É uma excelente ferramenta que podemos adotar na escrita de aplicações que fazem uso da arquitetura de micro serviços.

Spring Boot


Spring Boot
Ecossistema Spring
Spring Data
Spring Security
Spring Integration
Spring ...
Spring Framework

Princípios do Spring Boot

1. Prover uma experiência de início de projeto (getting started experience) extremamente rápida e direta;
2. Apresentar uma visão bastante opinativa (opinionated) sobre o modo como devemos configurar nossos projetos Spring, mas ao mesmo tempo flexível o suficiente para que possa ser facilmente substituída de acordo com os requisitos do projeto;
3. Fornecer uma série de requisitos não funcionais já pré-configurados para o desenvolvedor como, por exemplo, métricas, segurança, acesso a base de dados, servidor de aplicações/servlet embarcado, etc.;
4. Não prover nenhuma geração de código e minimizar a zero a necessidade de arquivos XML.

Visão Opinativa

- Conceito de convenção sobre configuração, porém levemente modificado
- É o grande motor por trás do ganho de produtividade do Spring Boot
- Frameworks como Ruby on Rails e Grails já o aplicam há bastante tempo.
- Dado que a maior parte das configurações que o desenvolvedor precisa escrever no início de um projeto são sempre as mesmas, por que já não iniciar um novo projeto com todas estas configurações já definidas?
- Não temos uma imposição aqui, mas sim sugestões
- Se seu projeto requer um aspecto diferente daquele definido pelas convenções do framework, o programador precisa alterar apenas aqueles locais nos quais a customização se aplica

Spring Boot
Spring Scripts

- Uma mudança interessante a se observar é a aparente ausência do Servlet container.
- Ele está presente, porém é executado de uma forma ligeiramente diferente.
- Ao invés de empacotarmos nosso projeto em um arquivo WAR para ser implantado no servidor, agora o próprio Spring Boot se encarrega de embutir o servidor entre as dependências do projeto, iniciá-lo e, de uma forma completamente transparente, implantar nossa aplicação neste.
- Spring Scripts são uma solução interessante quando precisamos escrever algum serviço REST simples e direto como, por exemplo, aqueles que consistem em meros CRUDs a bases de dados pré-existentes.
- Muitos recomendam a utilização da linguagem Goovy, que é uma linguagem projetada para roda na JVM e é cada vez mais importância da esta linguagem dentro do ecossistema Spring.
    - Não faz parte do escopo da disciplinas.

Serviços Rest

- O modelo REST (Representational State Transfer) representa uma possibilidade para a criação de web services
- Utiliza as semântica dos métodos HTTP (GET, POST, PUT e DELETE)
- Pacotes leves dos pacotes de dados transmitidos na rede e na simplicidade, fazendo desnecessária a criação de camadas intermediárias para encapsular os dados.

Spring Boot
Micro serviços

- Desde 2012, um termo que tem ganhado cada vez mais atenção entre os arquitetos é “micro serviço” (ou micro services).
- Quando lançada a versão 4.0 do Spring, foi feita uma referência a esse estilo arquitetural, visto pela equipe responsável pelo desenvolvimento do framework como uma das principais tendências do mercado.
- Mas, afinal, o que é um micro serviço?

Spring Boot
Micro serviços

- O problema que a arquitetura baseada em micro serviços busca resolver é a eterna questão da componentização.
- Sendo assim, para que se possa definir o que é um micro serviço, primeiro precisa-se entender o que é um componente.
- “uma unidade de software que pode ser aprimorada e substituída de forma independente”[1].

1 - James Lewis e Martin Fowler, no artigo chamado Microservices.

Projeto Spring
Boot com Maven

- Facilitar a configuração básica foi criado o site:
  - SPRING INITIALIZR
  - https://start.spring.io/
- Assistente no qual o desenvolvedor seleciona quais tecnologias deseja incluir em seu projeto, como Spring Data, Spring Security e tantas outras, preenche os campos fornecidos e, uma vez finalizado o preenchimento do formulário, simplesmente clica sobre o botão Generate Project.
- Feito isso, será gerado um arquivo zip, a ser descompactado no diretório de preferência do usuário e na sequencia importado pela IDE
- A primeira observação é no arquivo pom.xml quanto ao gerenciamento de dependências.
- A modularização das diversas tecnologias que compõem o ecossistema Spring em starter POMs.
- Tratam-se de arquivos POM previamente configurados pela equipe de desenvolvimento do framework que já contêm todas as configurações necessárias para que o programador possa usar imediatamente

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-
parent</artifactId>
</parent>
```

Projeto Spring

Boot com Maven

- Com esta configuração o projeto já está pronto para a aplicação desktop
- Acrescentando mais o pacote abaixo está pronto para configurar serviços REST e projeto Web

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
</dependencies>
```

- Será necessário outras dependências conforme o objetivo da aplicação, como bibliotecas de banco de dados (MySql, Postgre, Oracle, etc), entre outras que será feito o uso conforme sua necessidade e deverão ser configuradas no pom
- As configurações e versão normalmente são baixadas do repositório do maven
    - https://mvnrepository.com/

Guia das annotations do Spring

- **@Configuration -** É uma annotation que indica que determinada classe possui métodos que expõe novos beans.
- **@Controller -** Associada com classes que possuem métodos que processam requests numa aplicação web.
- **@Repository -** Associada com classes que isolam o acesso aos dados da sua aplicação. Comumente associada a DAO's.
- **@Service -** Associada com classes que representam a ideia do Service do Domain Driven Design. Para ficar menos teórico pense em classes que representam algum fluxo de negócio da sua aplicação. Por exemplo, um fluxo de finalização de compra envolve atualizar manipular o carrinho, enviar email, processar pagamento etc. Este é o típico código que temos dificuldade de saber onde vamos colocar, em geral ele pode ficar num **Service**.

Posted on July 13, 2016 by Alberto Souza.
https://domineospring.wordpress.com/2016/07/13/guia-das-annotations-do-spring/

Guia das annotations do Spring

- **@Component -** A annotation básica que indica que uma classe vai ser gerenciada pelo container do Spring. Todas as annotations descritas acima são, na verdade, derivadas de @Component. A ideia é justamente passar mais semântica.
- **@ComponentScan -** Em geral você a usa em classes de configuração (@Configuration) indicando quais pacotes ou classes devem ser scaneadas pelo Spring para que essa configuração funcione.
- **@Bean -** Anotação utilizada em cima dos métodos de uma classe, geralmente marcada com **@Configuration,** indicando que o Spring deve invocar aquele método e gerenciar o objeto retornado por ele. Quando digo gerenciar é que agora este objeto pode ser injetado em qualquer ponto da sua aplicação.
- **@Autowired -** Anotação utilizada para marcar o ponto de injeção na sua classe. Você pode colocar ela sobre atributos ou sobre o seu construtor com argumentos.

Posted on July 13, 2016 by Alberto Souza.
https://domineospring.wordpress.com/2016/07/13/guia-das-annotations-do-spring/

Guia das annotations do Spring

- **@Scope -** Annotation utilizada para marcar o tempo de vida de um objeto gerenciado pelo container. Pode ser utilizada em classes anotadas com @Component, ou alguma de suas derivações. Além disso também pode usada em métodos anotados com @Bean. Quando você não utiliza nenhuma, o escopo default do objeto é o de aplicação, o que significa que vai existir apenas uma instância dele durante a execução do programa. Você alterar isso, anotando o local e usando alguma das constantes que existem na classe *ConfigurableBeanFactory* ou *WebApplicationContext.*
- **@RequestMapping -** Geralmente utilizada em cima dos métodos de uma classe anotada com **@Controller.** Serve para você colocar os endereços da sua aplicação que, quando acessados por algum cliente, deverão ser direcionados para o determinado método.
- **@ResponseBody -**Utilizada em métodos anotados com **@RequestMapping** para indicar que o retorno do método deve ser automaticamente escrito na resposta para o cliente. Muito comum quando queremos retornar JSON ou XML em função de algum objeto da aplicação.

Posted on July 13, 2016 by Alberto Souza.
https://domineospring.wordpress.com/2016/07/13/guia-das-annotations-do-spring/

Guia das annotations do Spring

- **@Primary -** Caso você tenha dois métodos anotados com **@Bean** e com ambos retornando o mesmo tipo de objeto, como o Spring vai saber qual dos dois injetar por default em algum ponto da sua aplicação? É para isso que serve a annotation **@Primary.** Indica qual é a opção padrão de injeção. Caso você não queira usar a padrão, pode recorrer a annotation @**Qualifier.**
- **@Profile -** Indica em qual profile tal bean deve ser carregado. Muito comum quando você tem classes que só devem ser carregadas em ambiente de dev ou de produção. Essa eu utilizo bastante e acho uma ideia simples, mas super relevante dentro do Spring.
- **@SpringBootApplication -** Para quem usa Spring Boot, essa é uma das primeiras que você. Ela engloba a **@Component, @ComponentScan** e mais uma chamada **@EnableAutoConfiguration**, que é utilizada pelo Spring Boot para tentar advinhar as configurações necessárias para rodar o seu projeto. Bom, acho que é isso aí. Adoro escrever posts que foram solicitados diretamente pela galera que acompanha o meu trabalho, é uma motivação bem grande.

Posted on July 13, 2016 by Alberto Souza.
https://domineospring.wordpress.com/2016/07/13/guia-das-annotations-do-spring/

Guia das annotations do Spring

- **@EnableAsync -** Essa aqui não é tão comum, mas muitas vezes você precisa ações no sistema em background(outra thread). Essa annotation deve ser colocada em alguma classe marcada com **@Configuration,** para que o Spring habilite o suporte a execução assíncrona.
- **@Async -** Uma vez que seu projeto habilitou o uso de execução de métodos assíncronos com a **@EnableAsync,** você pode marcar qualquer método de um bean gerenciado do projeto com essa annotation. Quando tal método for invocado, o Spring vai garantir que a execução dele será em outra thread.

Posted on July 13, 2016 by Alberto Souza.
https://domineospring.wordpress.com/2016/07/13/guia-das-annotations-do-spring/