Spring Boot

ANDRE ALBINO PEREIRA


GENTE
QUAL A NECESSIDADE DISSO?

Problema

- Modelo servidor de aplicação Java:
  - Para aplicações compartilharem recursos da JVM? Ou instalar um servidor pra cada aplicação?
  - Para gerar conflitos em libs provided? (JSF, RESTEASY, etc).
  - Fazer com que aplicações tenham que funcionar com versões diferentes de servidores de aplicação pois é inviável atualizá-los?
  - Para trazer inúmeros recursos não utilizados?
  - Para complicar a automatização de instalação e tarefas de start/stop/reset dos serviços?

Portanto...

- Concordam com os problemas?
- Sugestões de como essas questões podem ser resolvidas?
- Recomendo a leitura (autor berhard Wolff):
  Application Servers Are (Sort of) Dead!

Spring Boot?

”Estabelece um modelo para concepção de aplicações Spring prontas para produção. Spring Boot favorece convenção sobre configuração e é desenhado para 'subir' e “rodar” o mais rápido possível”

Spring Boot?

”Simplifica a criação de aplicações 'stand-alone' baseadas em Spring que você apenas 'roda'. Estabelecemos uma visão 'opinativa' da plataforma Spring e componentes terceiros para que você comece com o mínimo de 'estardalhaço'.”

Má como?

É simples comissário...

- Crie aplicações Spring “stand-alone”.
- Agregue Tomcat ou Jetty diretamente (esqueça WARs e Containers).
- “Dê a partida” nos projetos à partir de POMs “starter” (Maven/Gradle).
- **Configuradores automáticos** para Spring e bibliotecas terceiras.
- Nenhuma exigência de configuração via XML.

E mais...

- **REFERENCE GUIDE** Elaborado.
- **UBER JAR** executável, mas ainda pode mantér compatibilidade com **WAR**.
- **Spring Boot CLI** para desenvolvimento ágil de aplicações com Groovy (com auto-complete).
- **Logging** com configuradores automá-ticos.

E mais...

- Acesso a dados: **JPA**, **MONGO**, **JDBC**.
- **Provisão de conteúdo estático**.
- Suporte a template engines (além de JSP): **Thymeleaf**, Freemarker, **Groovy Templates**, Velocity Templates.
- Quanto a outros frameworks WEB:
  - RestEasy: Compatível, necessário desabilitar SpringMVC (na versão 3.0.9
  - JSF: **Possível**
  - EJB/CDI: Compatível...mas qual a necessidade?

Exemplo 1

- Disponível no **GITHUB**
- Aplicação com endpoint REST (via Spring MVC).
- Provisão de conteúdo estático (HTML, JS, etc), uso de webjars.
- Testes de integração com Groovy + Spring Boot.
- Exemplo de uso do **application.properties**.
- Gestão de dependências/build com Gradle.

Exemplo 2 (Portal dos Advogados - WS)

- Disponível no **BITBUCKET.**
- Aplicação com endpoint REST (via RestEasy).
- Acesso a dados com JDBC e Postgres.
- Envio de email, templating com Velocity.
- Testes de integração com Spring Boot em Java.
- Ambiente de desenvolvimento construído com Vagrant e Puppet.

Percepções

- Processo de desenvolvimento muito mais ágil (aplicação normal Java, com método main).
- Familiar, já que utiliza containers Web conhecidos (Tomcat, Jetty).
- Automatização de instalação facilitada (copiar e rodar o executável).
- Necessário “abraçar” o Spring...
- Realidade na UNJ (tribunais, ADV).

Outras referências:

- Spring Boot
- Application Servers Are (Sort of) Dead!
- Gradle
- Spring MVC – REST Services
- WebJars
- Exploring Micro-frameworks: Spring Boot
- Thymeleaf

Considerações?