Spring framework — integrace

Pavel Mička

Obsah

Co již víme o Springu

- Inversion of control/Dependency injection container
- Open source
- POJO–based (zabraňuje vendor lock-inu)
- Usnadňuje vytváření enterprise aplikací
- Dobře integrovatelný s mnoha frameworky a knihovnami

Co již víme o Springu

- Inversion of control/Dependency injection container
- Open source
- POJO–based (zabraňuje vendor lock-inu)
- Usnadňuje vytváření enterprise aplikací
- Dobře integrovatelný s mnoha frameworky a knihovnami

Co již víme o Springu

- Inversion of control/Dependency injection container
- Open source
- POJO–based (zabraňuje vendor lock-inu)
- Usnadňuje vytváření enterprise aplikací
- Dobře integrovatelný s mnoha frameworky a knihovnami

Co již víme o Springu

- Inversion of control/Dependency injection container
- Open source
- POJO–based (zabraňuje vendor lock-inu)
- Usnadňuje vytváření enterprise aplikací
- Dobře integrovatelný s mnoha frameworky a knihovnami

Co již víme o Springu

- Inversion of control/Dependency injection container
- Open source
- POJO–based (zabraňuje vendor lock-inu)
- Usnadňuje vytváření enterprise aplikací
- Dobře integrovatelný s mnoha frameworky a knihovnami

Spring + JPA I.

- JPA je standardně konfigurováno pomocí souboru *persistence.xml*
- Spring má svou vlastní XML konfiguraci
- K integraci lze přistoupit několika způsoby
  - Ponechat persistence.xml a načíst jej Springem
    - Výhoda: JPA máme persistence.xml, nemusíme nic přepisovat
    - Nevýhoda: Máme dva typy XML konfigurace
  - Konfigurovat JPA pomocí Springu
    - Výhoda: jeden typ konfigurace
    - Nevýhoda: jsme závislí na Springu navíc

Spring + JPA I.

- JPA je standardně konfigurováno pomocí souboru *persistence.xml*
- Spring má svou vlastní XML konfiguraci
- K integraci lze přistoupit několika způsoby
  - Ponechat persistence.xml a načíst jej Springem
    - Výhoda: JPA máme persistence.xml, nemusíme nic přepisovat
    - Nevýhoda: Máme dva typy XML konfigurace
  - Konfigurovat JPA pomocí Springu
    - Výhoda: jeden typ konfigurace
    - Nevýhoda: jsme závislí na Springu navíc

Spring + JPA I.

- JPA je standardně konfigurováno pomocí souboru *persistence.xml*
- Spring má svou vlastní XML konfiguraci
- K integraci lze přistoupit několika způsoby
  - Ponechat persistence.xml a načíst jej Springem
    - Výhoda: Již mám persistence.xml, nemusím nic přepisovat
    - Nevýhoda: Mám dva typy XML konfigurace
  - Konfigurovat JPA pomocí Springu
    - Výhoda: jeden typ konfigurace
    - Nevýhoda: jedna závislost na Springu navíc

Spring + JPA I.

- JPA je standardně konfigurováno pomocí souboru *persistence.xml*
- Spring má svou vlastní XML konfiguraci
- K integraci lze přistoupit několika způsoby
  - Ponechat persistence.xml a načíst jej Springem
    - Výhoda: Již mám persistence.xml, nemusím nic přepisovat
    - Nevýhoda: Mám dva typy XML konfigurace
  - Konfigurovat JPA pomocí Springu
    - Výhoda: jeden typ konfigurace
    - Nevýhoda: jedna závislost na Springu navíc

Spring + JPA I.

- JPA je standardně konfigurováno pomocí souboru *persistence.xml*
- Spring má svou vlastní XML konfiguraci
- K integraci lze přistoupit několika způsoby
  - Ponechat persistence.xml a načíst jej Springem
    - Výhoda: Již mám persistence.xml, nemusím nic přepisovat
    - Nevýhoda: Mám dva typy XML konfigurace
  - Konfigurovat JPA pomocí Springu
    - Výhoda: jeden typ konfigurace
    - Nevýhoda: jedna závislost na Springu navíc

Spring + JPA I.

- JPA je standardně konfigurováno pomocí souboru *persistence.xml*
- Spring má svou vlastní XML konfiguraci
- K integraci lze přistoupit několika způsoby
  - Ponechat persistence.xml a načíst jej Springem
    - Výhoda: Již mám persistence.xml, nemusím nic přepisovat
    - Nevýhoda: Mám dva typy XML konfigurace
  - Konfigurovat JPA pomocí Springu
    - Výhoda: jeden typ konfigurace
    - Nevýhoda: jedna závislost na Springu navíc

Spring + JPA I.

- JPA je standardně konfigurováno pomocí souboru *persistence.xml*
- Spring má svou vlastní XML konfiguraci
- K integraci lze přistoupit několika způsoby
  - Ponechat persistence.xml a načíst jej Springem
    - Výhoda: Již mám persistence.xml, nemusím nic přepisovat
    - Nevýhoda: Mám dva typy XML konfigurace
  - Konfigurovat JPA pomocí Springu
    - Výhoda: jeden typ konfigurace
    - Nevýhoda: jedna závislost na Springu navíc

Spring + JPA I.

- JPA je standardně konfigurováno pomocí souboru *persistence.xml*
- Spring má svou vlastní XML konfiguraci
- K integraci lze přistoupit několika způsoby
  - Ponechat persistence.xml a načíst jej Springem
    - Výhoda: Již mám persistence.xml, nemusím nic přepisovat
    - Nevýhoda: Mám dva typy XML konfigurace
  - Konfigurovat JPA pomocí Springu
    - Výhoda: jeden typ konfigurace
    - Nevýhoda: jedna závislost na Springu navíc

Spring + JPA I.

- JPA je standardně konfigurováno pomocí souboru *persistence.xml*
- Spring má svou vlastní XML konfiguraci
- K integraci lze přistoupit několika způsoby
  - Ponechat persistence.xml a načíst jej Springem
    - Výhoda: Již mám persistence.xml, nemusím nic přepisovat
    - Nevýhoda: Mám dva typy XML konfigurace
  - Konfigurovat JPA pomocí Springu
    - Výhoda: jeden typ konfigurace
    - Nevýhoda: jedna závislost na Springu navíc

Spring + JPA II.

```
<bean id="entityManagerFactory"
  class="org.springframework.orm.jpa.
     LocalContainerEntityManagerFactoryBean">
  <property name="dataSource" ref="dataSource"/>
    <property name="jpaVendorAdapter">
      <bean class="org.springframework.orm.jpa.
         vendor.HibernateJpaVendorAdapter">
        <property name="databasePlatform" value="${
           jpa.platform}"/>
        <property name="generateDdl" value="true"/>
        <property name="showSql" value="true"/>
      </bean>
    </property>
  <property name="packagesToScan" value="cz.xy" />
</bean>
```

Integrace kontejnerů

- Některé frameworky jsou samy o sobě kontejnerem (např. JSF)
- Pro integraci volíme obvykle Spring-centric přístup
  - Snažíme se použití druhého kontejneru maximálně vytěsnit (eliminace dvojí konfigurace)
  - Musíme však zajistit, aby části integrovaného frameworku, které se Springem nepočítají, měly k dispozici původní kontejner se všemi jeho vlastnostmi
- Řešením bývá owrapování původního kontejneru integrované technologie, tak aby resolvoval nejprve ze svého obsahu, a pokud nenajde příslušnou beanu, tak aby nechal resolvovat Spring

Integrace kontejnerů

- Některé frameworky jsou samy o sobě kontejnerem (např. JSF)
- Pro integraci volíme obvykle Spring-centric přístup
  - Snažíme se použití druhého kontejneru maximálně vytěsnit (eliminace dvojí konfigurace)
  - Musíme však zajistit, aby části integrovaného frameworku, které se Springem nepočítají, měly k dispozici původní kontejner se všemi jeho vlastnostmi
- Řešením bývá owrapování původního kontejneru integrované technologie, tak aby resolvoval nejprve ze svého obsahu, a pokud nenajde příslušnou beanu, tak aby nechal resolvovat Spring

Integrace kontejnerů

- Některé frameworky jsou samy o sobě kontejnerem (např. JSF)
- Pro integraci volíme obvykle Spring-centric přístup
  - Snažíme se použití druhého kontejneru maximálně vytěsnit (eliminace dvojí konfigurace)
  - Musíme však zajistit, aby části integrovaného frameworku, které se Springem nepočítají, měly k dispozici původní kontejner se všemi jeho vlastnostmi
- Řešením bývá owrapování původního kontejneru integrované technologie, tak aby resolvoval nejprve ze svého obsahu, a pokud nenajde příslušnou beanu, tak aby nechal resolvovat Spring

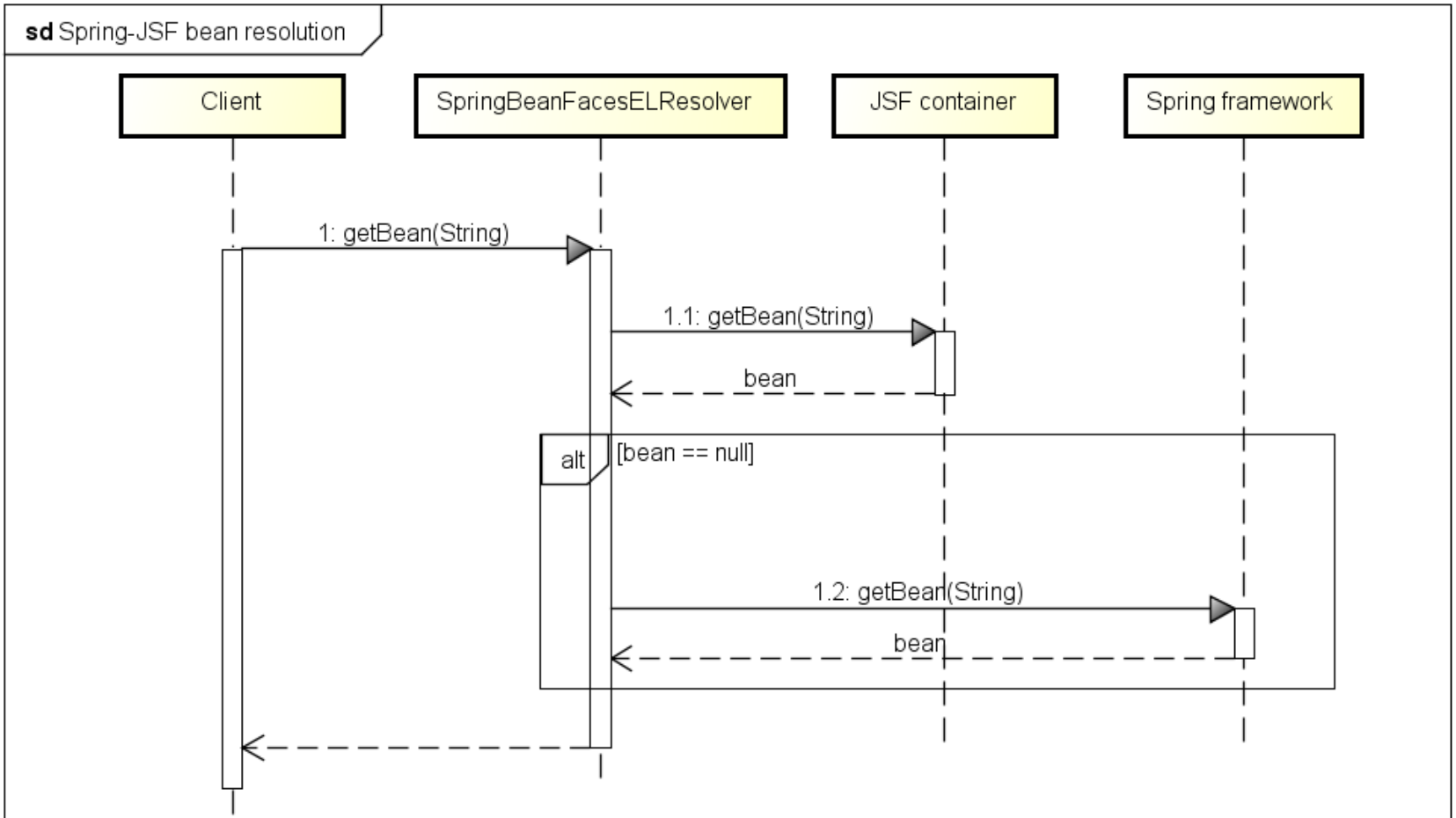
Integrace kontejnerů

- Některé frameworky jsou samy o sobě kontejnerem (např. JSF)
- Pro integraci volíme obvykle Spring-centric přístup
  - Snažíme se použití druhého kontejneru maximálně vytěsnit (eliminace dvojí konfigurace)
  - Musíme však zajistit, aby části integrovaného frameworku, které se Springem nepočítají, měly k dispozici původní kontejner se všemi jeho vlastnostmi
- Řešením bývá owrapování původního kontejneru integrované technologie, tak aby resolvoval nejprve ze svého obsahu, a pokud nenajde příslušnou beanu, tak aby nechal resolvovat Spring

Integrace kontejnerů

- Některé frameworky jsou samy o sobě kontejnerem (např. JSF)
- Pro integraci volíme obvykle Spring-centric přístup
  - Snažíme se použití druhého kontejneru maximálně vytěsnit (eliminace dvojí konfigurace)
  - Musíme však zajistit, aby části integrovaného frameworku, které se Springem nepočítají, měly k dispozici původní kontejner se všemi jeho vlastnostmi
- Řešením bývá owrapování původního kontejneru integrované technologie, tak aby resolvoval nejprve ze svého obsahu, a pokud nenajde příslušnou beanu, tak aby nechal resolvovat Spring

Spring a JSF


sd Spring-JSF bean resolution
Client
SpringBeanFacesELResolver
JSF container
Spring framework
1: getBean(String)
1.1: getBean(String)
bean
alt
[bean == null]
1.2: getBean(String)
bean

XML-based injection

```
<faces-config version="2.0"
    xmlns="..."
    xmlns:xsi="..."
    xsi:schemaLocation="...">
  <application>
      <el-resolver>
        org.springframework.web.jsf.el.
            SpringBeanFacesELResolver
        </el-resolver>
  </application>
</faces-config>
```

- Integraci zajistíme vložením *el-resolveru* do konfigurace JSF (*faces-config.xml*)

Integrace kontejnerů

- Existují frameworky, jejichž objekty nemohou být managované Springem (jejich lifecycle musí být řízen externí entitou)
- Bohužel i do těchto objektů musíme vložit závislosti
- Příkladem může být framework Vaadin a jeho session-global objekt *Application*
- Pro integraci použijeme anotaci *@Configurable*

Integrace kontejnerů

- Existují frameworky, jejichž objekty nemohou být managované Springem (jejich lifecycle musí být řízen externí entitou)
- Bohužel i do těchto objektů musíme vložit závislosti
- Příkladem může být framework Vaadin a jeho session-global objekt *Application*
- Pro integraci použijeme anotaci *@Configurable*

Integrace kontejnerů

- Existují frameworky, jejichž objekty nemohou být managované Springem (jejich lifecycle musí být řízen externí entitou)
- Bohužel i do těchto objektů musíme vložit závislosti
- Příkladem může být framework Vaadin a jeho session-global objekt *Application*
- Pro integraci použijeme anotaci *@Configurable*

Integrace kontejnerů

- Existují frameworky, jejichž objekty nemohou být managované Springem (jejich lifecycle musí být řízen externí entitou)
- Bohužel i do těchto objektů musíme vložit závislosti
- Příkladem může být framework Vaadin a jeho session-global objekt *Application*
- Pro integraci použijeme anotaci *@Configurable*

Spring + Vaadin

```
@Configurable(preConstruction = true)
public class SpringHelloWorld extends com.vaadin.
    Application {

  @Autowired
  private MyBeanInterface bean;

  public void init() {
    final Window main = new Window("Hello window");
    setMainWindow(main);
    main.addComponent(new Label(bean.myMethod()));
  }
}
```