Spring keretrendszer

Sághy Ádám

Principal Software Engineer

LinkedIn: adamsaghy
Email: adam.saghy@webvalto.hu

pcf

Miért Spring?

- Elismerten az egyik legpopulárisabb Java keretrendszere
- Gyorsabb, egyszerűbb és biztonságosabb Java alkalmazás fejlesztés
- Fókuszban a sebesség, produktivitás és egyszerűség
- Netflix is Spring Boot-ot használ már

Miért Spring? - II.

- Spring világszerte
  - Milliók használják nap-mint-nap a Spring-re épülő alkalmazások, szolgáltatások sokaságát
  - Még a nagy nevek is részt vesznek a fejlesztésében
    - Google, Microsoft, Alibaba, stb.
- Flexibilitás
  - A Spring flexibilis felépítése és széleskörű megoldásokat nyújtó könyvtárjai lehetővé teszik változatos alkalmazás készítését

Miért Spring? - III.

- Produktivitás
  - Az egyik fő fókusz mindig is a produktivitáson volt a Spring fejlődése során
    - Gyorsan, megfelelő megoldások készítése
  - Spring Boot
    - Minimális konfiguráció mellett lehetővé teszi alkalmazások futtatását egy előre konfigurált beágyazott webszerveren.
  - Spring Cloud
    - Támogató megoldásaival gyorsan és egyszerűen lehet egész mikroszervíz alapú megoldásokat a felhőben futtatni
- Sebesség
  - Gyors indulási, gyors végrehajtás, reaktív megoldások, konfiguráció sablonok

Miért Spring? - IV.

- Biztonság
  - Jó hírnévnek örvendhet a Spring-es megoldások biztonság terén is
  - Spring Security
    - Egyszerű integráció ipari biztonsági megoldásokkal és házon belüli biztonságos megoldások
- Támogatottság
  - Hatalmas közösséggel rendelkezik
  - Lelkes kezdőktől a profikig
  - Rendkívül kiterjedt dokumentáció és oktatási anyag áll rendelkezésre

Mire használható a Spring?

- Microservice
- Reactive
- Cloud
- Web apps
- Serverless
- Event Driven
- Batch

Egy kis történelem

- 2003. június: Rod Johnson készítette el az első változatát, s már a kezdetektől nyílt forráskódú volt.
- A célja egyszerű volt, alternatívaként szolgálni az akkoriban még nagyon merev és ”súlyos” EJB-kel szemben.
- Nincs meghatározott fejlesztési modellje, inkább szolgált a kor olvasztó tégelyeként ahol a cél az volt, hogy megfelelő funkcionalitásokat biztosítson az alkalmazások fejlesztéséhez és növelje a produktivitást.

Bevezetés a Spring keretrendszerbe

- Cél, hogy a fejlesztő az infrastruktúra helyett az alkalmazásra tudjon koncentrálni
- Lehetővé teszi, hogy POJO-kból építsünk alkalmazást, s vállalati szolgáltatásokat biztosíthassunk hozzájuk nem-invazívan
  - Pici történelem, az EJB-k akkoriban még különféle interfészek és absztrakt osztályok implementációját és függvények felülírását követelték meg)

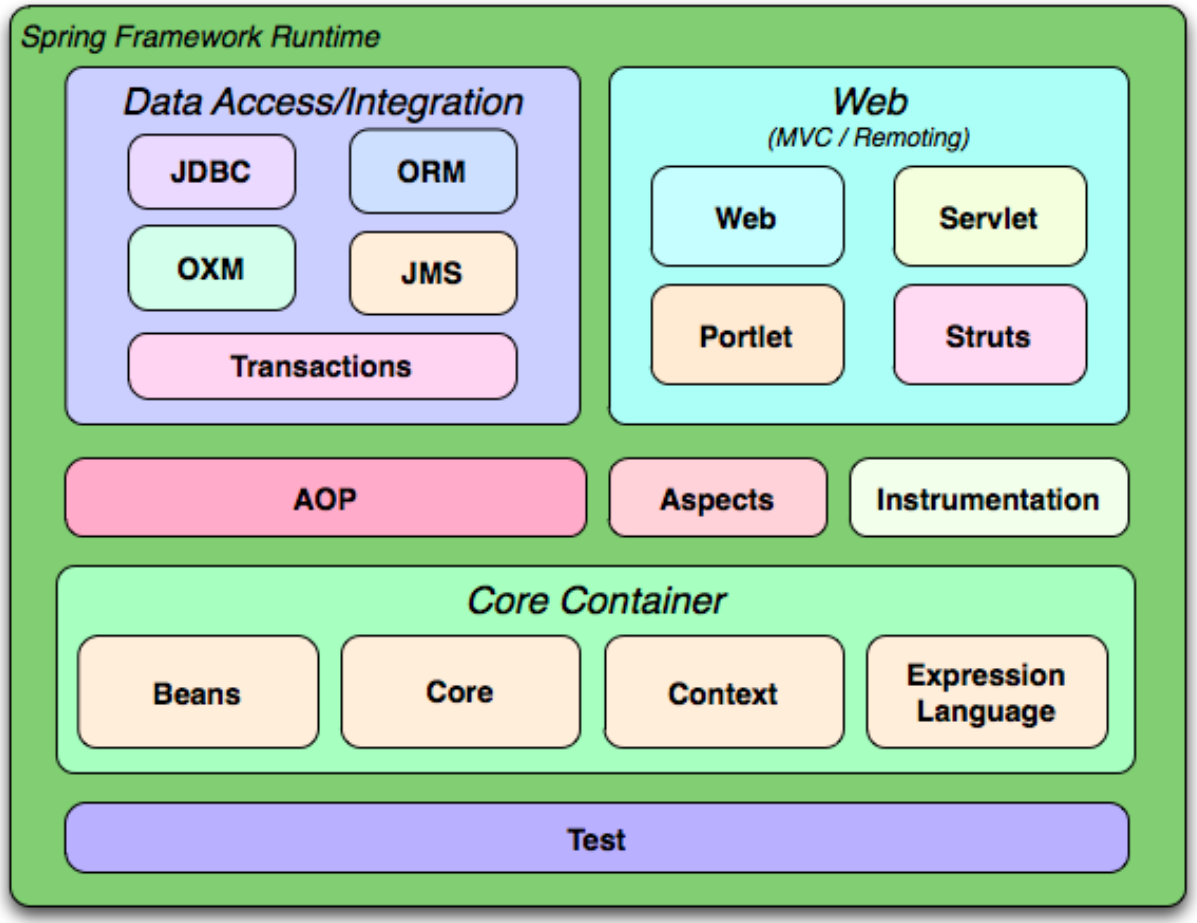
Spring gondolkodásmód:

- Java metódus tranzakcióban való végrehajtása a Transaction API-val való bíbelődés nélkül
- Lokális Java metódus meghívhatóvá tétele távolról a Remote API-val való bíbelődés nélkül

Convention over configuration

- A konfigurációval kapcsolatos konvenció a szoftverkeretrendszerek által használt szoftvertervezési paradigma, amely megpróbálja csökkenteni azon döntések számát, amelyeket a keretrendszert használó fejlesztőnek meg kell hoznia anélkül, hogy szükségszerűen elveszítené a rugalmasságot, és ne ismételje meg az elveket.
- Egyszerű példa: alapértelmezett port
- Komplexebb példa: Ha van in-memory H2 DB függőség a projekthez és nincs explicit konfiguráció a DB kapcsolathoz, automatikusan működni fog egy alapértelmezett viselkedéssel

Spring modulok és szolgáltatások



Core konténer

- Bean
  - Készítés
  - Elérhetőség
  - Életciklus kezelés
- Függőségek kezelése
- Kontext kezelés
- Expression language

Data Access/Integration

Adatok elérését támogató megoldásokat foglal magába

- JDBC
- ORM (JPA, JDO, Hibernate)
- OXM (Object/XML mapping, mint JAXB, XMLBeans, stb)
- JMS integráció
- Tranzakciókezelést (deklaratív / programozott)

Web

Web-servlet és egyéb kiegészítő modulok melyek lehetővé teszik webes alkalmazások készítését Spring alapokon:

- webes servlet listenerek
- Url mapping
- webes context kezelés
- Spring MVC framework
  - Model-View-Controller alapokon biztosít webes alkalmazások készítését és biztosítja az integrációt a többi Spring keretrendszeri modullal.

AOP

Aspektus orientált programozást tesz lehetővé:

- egy magasabb szintű absztrakciót vezet be az OO-hoz képest.
- Másképp fogalmazva: megadja nekünk az alkalmazáslogikát keresztbe-kasul vagdosó síkok (mint például a felhasználó-azonosítás) kiemelésének lehetőségét.
- A feladatokat bizonyos aspektusok szerint értelmezzük:
  - Naplózás
  - Tranzakció-kezelés

Test

- Támogatást nyújt a Spring komponensek teszteléséhez
  - Junit avagy TestNG tesztrendszerekkel.
- Spring ApplicationContext betöltés
- Spring ApplicationContext Caching
- Mock objektum támogatás

Inversion of Control és Dependency Injection

- A technika lényege, hogy a komponenskezelést (pl. létrehozást, példányosítást, paraméterezést, megszüntetést, metódus hívás) kiemeljük a programkódból, és általában egy külső keretrendszerre bízzuk (esetünkben ugye ez lenne a Spring)

- A lényege, hogy egy objektum más objektumok függőségeit elégíti ki.
  - Egyedül csak az interfészt határozza meg, mivel az határozza meg, hogy a kliens milyen funkcionalitásait éri el.

Mit nyerünk ezáltal?

Feladatkörök elválasztása

Hordozhatóság

Olvashatóság

Típushelyesség

Komplexitás csökkentése

Egy példa

Erős függőség

```
public class App {
    public App() {
        Driver myDriver = new oracle.jdbc.driver.OracleDriver();
        DriverManager.registerDriver( myDriver );

        String URL =
"jdbc:oracle:thin:username/password@amrood:1521:EMP";
        Connection conn = DriverManager.getConnection(URL);
        Statement st = con.createStatement();
    }
}
```

Függőségi injekció

```
public class App {
    public App(Connection connection) {
        Statement st = con.createStatement();
    }
}

public class OracleConnection implements Connection  {
    public OracleConnection() {
        Driver myDriver = new oracle.jdbc.driver.OracleDriver();
        DriverManager.registerDriver( myDriver );
        String URL =
"jdbc:oracle:thin:username/password@amrood:1521:EMP";
        DriverManager.getConnection(URL);
    }
}
```

Bean-ek regisztrálása Springben

- Két módszer
  - XML konfigurációs fájl

```
<?xml version = "1.0" encoding = "UTF-8"?>
<beans xmlns = "http://www.springframework.org/schema/beans"
        xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation = "http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

    <bean id = "App" class = "hu.webvalto.App">

</beans>
```

  - Annotációk
    - @Bean
    - @Component
    - ...

Függőségi injekció

- XML konfigurációs fájl

```
<?xml version = "1.0" encoding = "UTF-8"?>
<beans xmlns = "http://www.springframework.org/schema/beans"
       xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation = "http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
    <bean id="formalisKoszones" class="hu.webvalto.component.FormalisKoszontes"/>
    <bean id = "helloWorld" class = "hu.webvalto.component.HelloWorld">
        <property name="messageHandler" ref="formalisKoszones"/>
    </bean>
</beans>
```

- Annotáció
  - @Autowired
  - @Qualifier

```
@Autowired
public HelloWorld(@Qualifier("formalis") Koszontes koszonesHandler) {
    koszontesHandler = koszonesHandler;
}
```

Teszt

```
package hu.webvalto.component;

import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import static org.junit.Assert.assertNotNull;

public class AppTest {

    @Test
    public void findHelloWorldBean() {
        ApplicationContext applicationContext = new
ClassPathXmlApplicationContext("beans.xml");
        HelloWorld helloWorld = applicationContext.getBean("helloWorld", HelloWorld.class);
        assertNotNull(helloWorld);
    }
}
```

Projektek

- Spring Framework
- Spring Boot
- Spring Cloud
- Spring Cloud Data flow
- Spring Data
- Spring Integration
- Spring Batch
- Spring Security
- Spring Authorisation Server
- Spring GraphQL
- Spring Session
- Spring HATEOS
- Spring REST Docs
- Spring AMQP
- Spring Flo
- Spring Apache Kafka
- Spring LDAP
- Spring Shell
- Spring Statemachine
- Spring Web flow
- Spring Web services

Spring Boot

- Önálló alkalmazás készítés
  - Beépített webszerver
    - Tomcat, Jetty vagy undertow
- ”Extra” függőségek (kényelmi funkcionalitások)
- Autokonfiguráció
- Automatikus metrikák, életjel ellenőrzés
- Nincs kód generálás, nincsenek szükségtelen xml konfigurációk

- A cél a gyors alkalmazás készítés

Spring Cloud

- Hasznos tool-ok és funkcionalitások elosztott rendszerekre való fejlesztéshez
- Cloud szolgáltatóval való integráció
  - Spring Cloud Azure, AWS, GCP
- Elosztott konfiguráció támogatás
- Service registration and discovery
- Routing
- Load balancing
- Elosztott üzenetkezelés, session-ök
- ...

Spring Data

- Adatelérés Spring módra
- Egyszerű és egységes adatelérés és kezelés biztosítása különböző adatbázisok, adattároló megoldásokhoz
  - Relációs adatbázisok, Nem-relációs adatbázisok
  - Big Data, Felhős adattárolás
- Hasznos Repository és objectmapping megoldások
- Dinamikus query generálás
- Auditing

- Pár almodul: Spring Data JPA, Spring DATA JDBC, Spring Data REST,..

Spring Data JDBC

- JDBC: Java Database Connectivity
- Object relation mapping keretrendszer
- CRUD operations
- @Query annotation
- Nem túl komplikált adatelérés
  - Nincs caching, lazy loading, stb amit a JPA szolgál ki

Spring DATA JPA

- JPA: Java Persistence API
- JPA Repository támogatás
  - Kevesebb boilerplate kód
- QueryDSL
- Entitás audit
- Validáció
- Pagination

Spring Data REST

- REST: Representational State Transfer
- A REST egy szoftverarchitektúra típus, elosztott kapcsolat, nagy, internet alapú rendszerek számára
- Kényelmesen készíthető REST API-k
  - Beépített hibakezelés
  - Domain objektum transzformációk
  - Kérés / Válasz content-type transzformáció
  - Spring Data Repository REST-en keresztüli elérhetővététele

Spring Security

- Authentikációs és authorizációs megoldás
  - Név/jelszó
  - Oauth 2.0
  - SAML
  - Roles / Authorities
- Beépített védelmi vonalak: CSRF, Clickjacking, Session fixation
- Out of box funkcionalitás
- Hatékonyan testreszabható, kiegészíthető

Spring Integration

- Üzenetkezelés Spring alapú alkalmazások között
- Integráció külső rendszerekkel deklaratív adaptereken keresztül
  - EIP támogatás
- Messaging pattern
  - Message -> Üzenet
  - Channel -> Üzenet átvitel
  - Adapter -> Kapcsolódás a rendszerhez
  - Bridge -> Kapcsolat Channel-ek / Adapter-ek között
  - ServiceActivator -> Beérkezett üzeneten elvégzendő műveletek

Spring Batch

- Automatizált feladatok
  - Bizonyos lépések végrehajtása, egymásra épülő feladatok, szinkronizációs pontok, stb
- Tömeges adatfeldolgozás
  - Nagy mennyiségű, sokáig futó feldolgozás
  - Parallel, elosztott feldolgozás

Spring Flow

- Spring MVC-re épülő, azt kiegészítő webes megoldás
- Flow engine
  - Stateful alkalmazások készítése
  - Egymásra épülő, több lépéses webes folyamatok támogatása
- Megoldás a legégetőbb stateful webes problémákra
  - Session kezelés
  - Navigáció
  - Böngésző „Vissza” gomb támogatás
  - Konkurencia problémák

PEOPLE COME FIRST
INFORMATIKAI SZAKÉRTŐK EGYESÜLETE

Kérdések!

PEOPLE COME FIRST
INFORMATIKAI SZAKÉRTŐK EGYESÜLETE

Köszönöm a figyelmet!

Források

- Spring.io:
  - https://spring.io/why-spring
  - https://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/overview.html
- Wikipédia: https://en.wikipedia.org/wiki/Spring_Framework
- Prog.hu: https://prog.hu/cikkek/905/aspektus-orientalt-programozas

Sághy Ádám

Senior Software Engineer

LinkedIn: adamsaghy
Email: adam.saghy@webvalto.hu

pcf