**Bevezetés a Spring Framework és Spring Boot használatába**

**Spring Framework céljai**

**(nagyvállalati alkalmazásfejlesztés, keretrendszer mert, életciklus vezérlés, nagyvállalati alkalmazás def. -> 3rd arty libraryk)**

Egy **keretrendszer nagyvállalati alkalmazásfejlesztésre**

keretrendszer, mert **az általunk létrehozott komponenseket ő példányosítja** le

a keretrendszer az adott **komponensek életciklusát vezérli**

**nagyvállalati alkalmazás:** ahol már **nem elégségesek** a Java SE-be beépített eszközök/tulajdonságok, hanem valami **további tulajdonságokra van szükség**(ritkább), bizonyosakra már **létező megoldásokat 3rd party libraryket** integrál egy egységes modellbe

**Nagyvállalati alkalmazás -1.**

**(igények -> komponens életciklus kezelés és kapcsolatok; távoli elérés, többszálúság, perzisztencia, tranzakciókezelés, aszinkron üzenetküldés, feladatok ütemezése)**

*Milyen igények, problémák jelentkeznek, amelyek SE-ben nem, vagy csak túl nagy munka árán megvalósíthatóak? (7)*

**komponens életciklus kezelése és kapcsolatok** (példányosítás, atri példányosítás)

**távoli elérés** – az adott alkalmazás elérhető legyen egy másik számítógépről is, tipikusan interneten, weben keresztül

**többszálúság** – több felhasználó **párhuzamos felhasználási** biztosított legyen fejlesztői plusz programozás nélkül; a fejlesztőnek **elég legyen egy felhasználóra megírni,** a párhuzamosságot oldja meg a keretrendszer

**perzisztencia**; Java SE-ben csak JDBC van, ami már elavult(checked exceptionok) és sok munka

**tranzakciókezelés** – tudjunk **atomic műveleteket** definiálni és elosztott esetnél vannak olyan műveletek is, amik **egyszerre két erőforráson(adatbázison)** is végeznek műveleteket

**aszinkron üzenetküldés** – egyik alkalmazás tudjon küldeni másik alkalmazásnak és tudjon is tőle fogadni üzenetet; **aszinkron, mert nem várja meg a választ** (http nem ilyen, az szinkronos és megvárja)

**ütemezés** – bizonyos műveletek valamilyen ütemezés szerint, ne pedig felhasználói beavatkozásra fussanak le

**Nagyvállalati alkalmazás -2. - további elvárások (6)**

**(integráció más alkalmazásokhoz, auditálhatóság, konfigurálhatóság, biztonság, tesztelhetőség)**

az alkalmazás tipikusan **nem önmagában létező;** más alkalmazásokhoz kell **integrálódnia** -> a keretrendszernek különböző integrációs módokat kell tudnia támogatni

**auditálhatóság** – ha a **felhasználó** valamilyen **írási műveletet** folytatott az alkalmazáson belül, az **legyen visszanézhető,** legyen nyoma

**konfigurálhatóság** – egy alkalmazás **futhasson különböző környezetekben**, pl. test környezet, éles környezet, fejlesztő számítógépén, különböző beállításokkal

naplózás, monitorozás és beavatkozás = **üzemeltetési szempontok**

**biztonság** – valahogy **be** lehessen **jelentkezni**, a **felhasználó csak a hozzárendelt szerepkörei** alapján tudjon műveleteket végezni

**tesztelhetőség** – **lehessen és legyen automatizált,** minimalizálva a hibák számát és az architektúra megértést is támogatja

**Spring Framework tulajdonságai -1.**

*Ahhoz, hogy ezeket a nagyvállalati igényeket kielégítse, a következő tulajdonságokkal rendelkezik:*

**(komponensek = egyszerű java objektumok, konténer = AC -> példányosítás, vezérlés, kapcsolatok, referencia elengedés; Dependency Injection és Inversion of Control; konfiguráció = komponensek és kapcsolataik leírása -> annotációk!; pehelysúlyú komponensek, POJO (nem kell interface); aspektusorientált programozás támogatása konfigurációval)**

**komponensek** – nekünk, **fejlesztőknek kell implementálnunk**; **egyszerű sima java objektumok**; a **konténer felügyelete alatt állnak** -> ő **példányosítja** le őket **és engedi el a referenciát**, amikor már nincs rá szükség**; konténer Springes neve**: **Applicaiton Context**

**konténer** – **vezérli a komponensek életciklusát** (nem nekünk kell példányosítani a létrehozott komponenseket); **felügyeli a komponensek közötti kapcsolatokat** -> A Spring egy **Dependency Injection** és **Inversion of Control** keretrendszer

**Inversion of Control:**

átadja a komponensek felügyeletét egy konténernek vagy keretrendszernek; hagyományos programozásban a programkód hívja a 3r party librarykat, míg az IOC-ben a keretrendszer hívja a kódunkat

**Dependency Injection**

objektum összekapcsolása másik objektummal = objektum injektálása egy másik objektumba, fejlesztői példányosítás nélkül

**AC**

**Felelős a komponensek=beanek példányosításáért, konfigruálásért, összegyűjtéséért, kapcsolataik kialakításáért és az életciklusuk vezérléséért**

**a komponenseket és a kapcsolatokat valahogy a konténernek le kell írni, hogy majd megfelelően tudja őket példányosítani és vezéreln**i (kapcsolatokat közöttük létre tudja hozni) -> ezen **konfigurációra** a létrehozási módok: XML(legrégebbi); **annotáció**, vagy Java kódból

**Pehelysúlyú kompenensek** (egyszerű java osztályok, nem kell interfaceket implementálni, sima egyszerű **POJO**-ok = Plain old Java object; **hétköznapi, speciális tulajdonságok nélküli)**

**Aspektusorientált programozás támogatása** – bizonyos **műveleteket** egy közös helyre **össze lehet gyűjteni**, majd **konfigurációval bekapcsolni**; ilyen pl. a **security, naplózás, tranzakciókezelés**

**Spring Framework tulajdonságai -2.**

**(saját implementációk helyett 3rd party library integrációk (pl. nincs saját scheduling); glue kód = 3rd partyk egységes integrált modellként); boilerplate kódok eliminálása(haszontalan, ismétlődő); üzleti logikára koncentrálás)**

**nem mindent önmaga implementál**, hanem legtöbbször a leggyakoribb legelterjedtebb **3rd party libraryket integrálja;** pl nincs saját ütemezése/schedulingja, helyette integrálni lehet több megoldást is, pl egy 3rd party erre a kvarc?

**Glue kód**nak is nevezik a Springet – a különböző modellbe felépített **3rd party libraryket egy egységes modellbe integrálja -**> egyszerű használat és konfiguráció

**Boilerplate kódok eliminálása** – egyik fő cél; meg kell írni, de **nincs haszna, ismétlődő**; csak az **üzleti logikára akarunk koncentrálni**

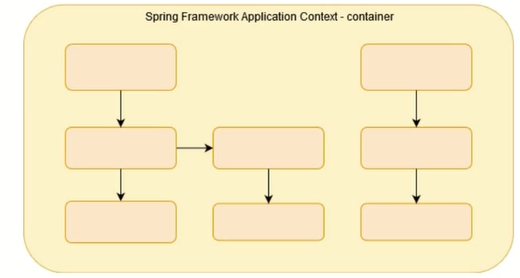
**Applicaiton Context**

**(mi van benne(fejlesztő által implementált komponensek = POJO objektumok és mit csinál ezekkel(példányosítás és kapcsolatok))**

= Spring konténer

**fejlesztő által implementált komponenseket** tartalmaz (= egyszerű Java osztályok) és ezek **kapcsolatait**

ezeket a komponenseket a konténer **példányosítja** és köztük a **kapcsolatokat** is ő **állítja be**

****

**Háromrétegű webes alkalmazás**

**(Spring erre való; repository-> perzisztens, adatbázis; service->üzleti logikai, controller -> user interface = prezentációs v. felhasználói réteg; saját Springes webes keretrendszer = Spring MVC; nincs saját http kezelés, ezt 3rd party web konténerre bízza)**

a Spring tipikusan ilyen alkalmazások fejlesztésére való; ettől függetlenül lehet használni vastagklienses, vagy egy számítógépen futó alkalmazások írására is, de **legszélesebb körben webes rétegelt alkalmazások fejlesztésére** használják

3 réteg:

*Repository* - perzisztens réteg az adatbázissal való kapcsolattartásra

*Service* - üzleti logikai réteg – az üzleti entitások és üzleti logika van itt implementálva

*Controller* - user interface, prezentációs réteg

**Spring MVC** (=Model-View-Controller) - a Framework része; **a Springes fejlesztők** nem találtak jó **webes keretrendszert,** ami megfelelően illeszthető lenne a Springbe, ezért **létrehoztak egy sajátot**

**A Spring önmaga nem kezeli a http-t, ennek kezelését web konténerre bízza** (pl. **Tomcat**, Jetty)

**HTTP** (***H****yper****T****ext****T****ransfer****P****rotocol*)

kérés-válasz alapú protokoll kliens és szerver között.

**Bevezetés a Spring Framework és Spring Boot használatába – gyakorlat**

**BOM** – bill of materials; tartalmazza a springes modulok verziószámait felsorolva

1.create new project – name: employees-spring, group id: microservice.training

2.pom xml beállítások: properties->UTF8, maven fordító; dependency->org-spring framework, type: pom, scope:import(=ez még így nem függőség, csak a spring moduljai közül egy konfig; majd a függőség konkrét felvevésénél már nem kellenek verziószámok); dependency->spring-webmvc; dependency->javax.servlet-api, scope: provided, igy nem lesz benne az alkalmazásban, a servlet apit a futtatókörnyezet webkonténerből majd biztostani fogja egy jettivel(a springes bom fileban ez nincs definiálva); projekt típus definíció: pacckaking: war(webes alkalmazás); kell még a legújabb Maven war plugin, ami nem szál el, ha nincs webxml -> build/plugin: maven-war-plugin; még egy plugin -> jetty; ez egy webkonténer, amire majd feltelepíti az alkalmazást és majd így lehet vele webböngészőn át kommunikálni a 8080as porton

3.WebappInitializer osztály és employees package; extends AbstractAnnotationConfigDispatcherServletInitializer -> webxml nélkül képes egy webalkalmazást inicializálni; az absztrat metódusait implementálni kell

getServletMapping - > a Spring milyen URLen lévő kéréseket szolgáljon ki -> ”/”

a másik kettő a rétegek felkonfigurálására jó; getServLetConfigClasses -> ControllerConfigClass(piros;create!); erre egy @ComponentScan annotáció -> erre a Spring fel fog olvasni osztályokat

új osztály: HelloController és erre @Controller annotáció; a bejövő http kérést kiszolgáló metódust kell még írni: public String sayHello(){}; return ..+.now!; a metódusra @GetMapping (”/”)annotáció

+ResponSeBody annotáció, hogy tudja, hogy be kell írnia a http válasz törzsébe majd

futtatás

Fent Add Configuration - > + -> Maven parancs -> Command line: clean package jettyrun

**Kérdések**

*Mi röviden a Spring?*

*Miért hívjuk keretrendszernek?*

*Nagyvállalati alkalmazásfejlesztésnél miért szükséges keretrendszer? A keretrendszer milyen megoldásokat használ?(2)*

*Milyen igények, problémák jelentkeznek, amelyek SE-ben nem, vagy csak túl nagy munka árán megvalósíthatóak?*

*Milyen elvárások vannak nagyvállalati alkalmazásfejlesztéskor a..*

*…a komponensek kezelésére? (3)*

*…távoli elérésre?*

*…a többszálúságra?*

*…a perzisztenciára?*

*…a tranzakciókezelésre?*

*…az aszinkron üzenetküldésre?*

*…az ütemezésre?*

*…más alkalmazásokkal történő integrációa?*

*…az auditálhatóságra?*

*…a konfigurálhatósága?*

*…az üzemeltetésre?*

*…a biztonságra?*

*…a tesztelhetőségre?*

*…az apsektusorientált programozásra?*

*Milyenek springben a komponensek és mit tudunk róluk?*

*Mi a springes konténer neve és miért felelős?*

*Mit jelent az, hogy a Spring egy Inversion of Control rendszer?*

*Mi a Dependency Injection?*

*Mivel konfiguráljuk a komponenseket?*

*Mit tudunk Springben a saját és a 3rd party implementációk kapcsolatáról?*

*Mi az a glue kód?*

*Mik azok a boilerplate kódok és eliminálásukkal a Springnek mi a célja?*

*Mi az a 3 rétegű webes alkalmazás?*

*Milyen webes keretrendszert használ a spring?*

*Hogy kezeli a Spring a httpt?*