UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Delfina Bariša

Razvoj sodobnih aplikacij z uporabo Jave Enterprise Edition in aplikacijskega strežnika WebSphere

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN MATEMATIKA

Ljubljana, 2018

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Delfina Bariša

Razvoj sodobnih aplikacij z uporabo Jave Enterprise Edition in aplikacijskega strežnika WebSphere

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN MATEMATIKA

MENTOR: prof. dr. Branko Matjaž Jurič

Ljubljana, 2018

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Proučite platformo in programski jezik Java in novosti, ki jih prinaša Java Standard in Enterprise Edition. Podrobno analizirajte nove in posodobljene tehnologije v Enterprise Edition in jih ovrednotite. Opravite pregled in primerjavo profilov in aplikacijskih strežnikov. Podrobno proučite aplikacijski strežnik IBM WebSphere ter napravite primerjavo med klasičnim in Liberty profilom.

Iskreno se zahvaljujem mentorju prof. dr. Branku Matjažu Juriču in as. Janu Meznariču za pomoč in vodenje pri izdelavi diplomske naloge.

Velika zahvala gre tudi moji družini, fantu Tomiju, prijateljem in sodelavcem, ki so me ves čas šolanja in pisanja diplomske naloge motivirali, bodrili in mi stali ob strani.

Kazalo

Kazalo slik

Kazalo tabel

Seznam uporabljenih kratic

kratica	angleško	slovensko
Java EE	Java Platform, Enterprise Edition	poslovna verzija platforme Java
Java SE	Java Platform, Standard Edition	standardna verzija platforme Java
WAS	WebSphere Application Server	aplikacijski strežnik WebSphere
API	Application programming interface	aplikacijski programski vmesnik
JSTL	JSP Standard Tag Library	standardna knjižnica oznak JSP
SPI	service provider interface	vmesnik ponudnika storitev
JSR	Java Specification Request	zahtevek za specifikacijo Java
TCP	Transmission Control Protocol	protokol za nadzor prenosa
HTTP	Hypertext Transfer Protocol	protokol namenjen prenosu informacij
POJO	Plain old Java object	navaden Java objekt
REST	Representation State Transfer	predstavitev prenosa stanja
SPI	Service provider interface	vmesnik ponudnika storitev
URI	Uniform Resource Identifier	enolični identifikator vira
JDK	Java SE Development Kit	razvojna oprema Jave SE
EIS	Enterprise information system	informacijski sistem podjetja
EAI	Enterprise application integration	integracija poslovnih aplikacij
RFC	Request for Comments	zahtevek za spremembo
UTC	Coordinated Universal Time	univerzalni koordinirani čas
IaaS	Infrastructure as a Service	infrastruktura kot storitev
PaaS	Platform as a Service	platforma kot storitev
SAAJ	SOAP with Attachments API for Java	SOAP s prilogami API za Javo

Povzetek

Naslov: Razvoj sodobnih aplikacij z uporabo Jave Enterprise Edition in aplikacijskega strežnika WebSphere

Avtor: Delfina Bariša

Za razvoj modernih več-nivojskih aplikacij je danes na voljo več različnih platform. Ena izmed vodilnih platform je Java Platform, Enterprise Edition (Java EE), ki se je med pisanjem diplomske naloge preimenovala in bo znana pod imenom Jakarta Enterprise Edition (Jakarta EE). Največ funkcionalnosti v več-nivojskih aplikacijah se izvaja na aplikacijskem strežniku. Aplikacijskih strežnikov za platformo Java EE je na tržišču zelo veliko, med njimi je zelo pogosto uporabljen aplikacijski strežnik WebSphere (WAS).

V diplomski nalog najprej predstavimo Javo in Javo EE. Nato opišemo več-nivojske aplikacije in njihovo implementacijo s platformo Java EE. Po tem si podrobno ogledamo novosti zadnjih dveh različic Jave EE in profile Jave EE. Na koncu opišemo aplikacijske strežnike in aplikacijski strežnik WAS.

Diplomska naloga temelji na pregledu literature in dokumentacije Jave EE, s poudarkom na njenih zadnjih dveh verzijah. Rezultat diplomske naloge je povzetek in analiza novosti zadnjih dveh verzij Jave EE ter pregled in analiza aplikacijskih strežnikov.

Ključne besede: Java Platform, Standard Edition, Java Platform, Enterprise Edition, več-nivojske aplikacije, aplikacijski strežnik WebSphere

Abstract

Title: Development of modern applications using Java Enterprise Edition and WebSphere Application Server

Author: Delfina Bariša

There are many different platforms available for developing modern multi-tier applications. One of the leading platforms is Java Platform, Enterprise Edition (Java EE), which was renamed during the writing of the thesis and will be known as Jakarta Enterprise Edition (Jakarta EE). The application server performs most functionalities of multi-tier applications. There are many application servers for the Java EE platform on the market, one of the most commonly used is WebSphere application server (WAS).

In this thesis we first introduce Java and Java EE. Then we describe multi-tier applications and their implementation with Java EE platform. After this, we look closely at the novelties of the last two versions of Java EE and the Java EE profiles. Finally, we describe application servers and WAS.

The thesis is based on a review of the literature and documentation of Java EE, with an emphasis on its last two versions. The result of the thesis is a summary and analysis of the latest two versions of Java EE and an overview and analysis of Java EE application servers.

Keywords: Java Platform, Standard Edition, Java Platform, Enterprise Edition, Multi-Tier Applications, WebSphere Application Server

1 Uvod

1.1 Motivacija

V preteklosti je bila struktura aplikacij popolnoma drugačna od današnje. Obstajale so namizne aplikacije *(desktop applications)* in aplikacije tipa odjemalec/strežnik *(client/server applications)*. Namizne aplikacije so bile nameščene na računalnike uporabnikov in običajno niso komunicirale z zunanjim svetom ter z njim niso delile podatkov, ampak so podatke shranjevale v lokalno podatkovno bazo. Aplikacije tipa odjemalec/strežnik so bile višje nivojske in so si z namiznimi aplikacijami delile veliko značilnosti glede strukture aplikacij, kot je na primer lokacija poslovne logike aplikaciji. Razlikovale pa so se v tem, da so aplikacije tipa odjemalec/strežnik dostopale do zunanjega sistema podatkovnih baz. Ta sistem podatkovnih baz se je običajno izvajal na namenskem strežniku ali strežniški farmi v omrežju podjetja., ki so si ga ali jo delile mnoge odjemalske aplikacije. Skupna značilnost obeh tipov aplikacij je bila ta, da je bila skoraj vsa poslovna logika kodirana kar v sami aplikaciji na odjemalcu in ne na ločenem nivoju, kot je danes. Strežniški nivo s podatkovno bazo je tako skrbel le za shranjevanje podatkov. Na žalost pa je ta struktura imela veliko slabosti. Ena izmed slabosti, ki je bila omenjena malo prej, je bila na primer ta, da je bila koda poslovne logike del odjemalca in je bila zato razširjena po vsem omrežju in podvojena na vsaki delovni postaji. Posledično to pomeni, da je bilo potrebno programsko opremo odjemalca ob vsaki spremembi poslovne logike ponovno namestiti na vse odjemalce. Velika težava takšne strukture pa je bila tudi, da je bil sistem še bolj občutljiv na napade, saj so lahko zlonamerni uporabniki enostavneje celo z lastno spremembo programske opreme direktno dostopali do podatkovne baze. Zato se je razvila več-nivojska arhitektura aplikacij, ki razdeli dostop do baze na več nivojev in s tem zagotovi večjo varnost.

Obstaja več platform in delovnih okolij za razvoj več-nivojskih aplikacij. V času pisanja diplomske naloge je bil po spletnih statistikah, ki bodo podrobneje razložene v diplomski nalogi, najbolj uporabljen programski jezik objektno usmerjen programski jezik Java. Prav na tem programskem jeziku temelji tudi zelo močna platforma Java Platform, Enterprise Edition (Java EE) za razvoj več-nivojskih aplikacij. Java EE se je med pisanjem diplomske naloge preimenovala in se bo imenovala Jakarta Enterprise Edition (Jakarta EE).

Platforma Java EE se redno posodablja. Podjetja pa v praksi počasi uvajajo njene nove verzije, posledično pa tudi nove verzije Jave SE. Zato bo diplomska naloga vsebovala pregled novosti zadnjih treh verzij Jave SE in zadnjih dveh verzij Jave EE.

1.2 Cilji

V diplomski nalogi bomo opisali teoretične osnove programskega jezika Java, platforme Java EE in več-nivojskih aplikacij ter njihovo implementacijo z uporabo Jave EE. Zaradi rednega posodabljanja Jave SE in Jave EE, bo nato velik del diplomske naloge namenjen tudi njunim novostim. Sledi še opis profilov Jave EE, aplikacijskih strežnikov in aplikacijskega strežnika WebSphere.

Veliko podjetij, tudi slovenskih, želi preiti na platformo Java EE, ker že poznajo močno razširjeno Javo SE ter jim je zato lažje osvojiti novo platformo za razvoj modernih več-nivojskih aplikacij. Ker ta platforma omogoča razvoj sodobnih aplikacij, posledično privabi več uporabnikov, kar se rezultira kot večji dobiček podjetja. Na žalost pa veliko podjetij zaradi že obstoječih aplikacij, ki so napisane v drugih programskih jezikih, ki jih je potrebno vzdrževati, nima prav veliko časa, da bi se poglobila v to platformo. Če pa so že prešli na to platformo, običajno niso seznanjeni z novostmi novejših različic. Zato bo diplomska naloga vsebovala vsebine, ki bodo zanimive takšnim podjetjem. Tisti, ki niso seznanjeni, bodo v diplomski nalogi pridobili osnovno teoretično znanje, ki ga bodo pozneje lažje razširili. Drugi, ki so seznanjeni, pa se bodo lahko v kratkem času in v enem dokumentu seznanili z novostmi in se nato sami poglobili in raziskali tista področja, ki jih potrebuje njihovo podjetje. Ker veliko podjetij, tudi slovenskih, prehaja iz Jave EE verzije 6 na verzijo 7, malo bolj napredna podjetja pa iz verzije 7 na verzijo 8, bodo v diplomski nalogi podrobneje opisane novosti verzij 7 in 8.

Prav tako bo diplomska naloga vsebovala vsebine, ki bodo zanimive tudi študentom, na novo zaposlenim in vsem tistim, ki niso seznanjeni z več-nivojskimi aplikacijami, Javo, Javo EE in aplikacijskim strežnikom WebSphere, a bi radi pridobili osnovno teoretično znanje, ki ga bodo nato lažje razširili.

1.3 Struktura diplomske naloge

Drugo poglavje naloge opisuje Java Platform, Standard Edition (Javo SE) in Javo EE, ki sta potrebni za razumevanje diplomske naloge. Ob tem so v tem poglavju naštete tudi novosti in posodobitve Jave SE verzij 8, 9 in 10 ter Jave EE verzij 7 in 8. Podrobnejši opis novosti in posodobitev Jave EE pa se zaradi obsežnosti nahaja v posebnem poglavju, in sicer četrtem.

Tretje poglavje je namenjeno opisu več-nivojskih poslovnih aplikacij ter njihovi implementaciji z Javo EE in je tako kot drugo poglavje ključno za razumevanje diplomske naloge. V podpoglavjih se bomo sprehodili skozi vsak nivo in ga podrobneje razložili. V tem poglavju je razložen tudi pomen vsebnikov, ki je nujen za razumevanje več-nivojskih aplikacij v Javi EE.

Četrto poglavje opisuje novosti in posodobitev Jave EE verzije 7 in 8, ki je namenjeno še posebej tistim, ki želijo preiti iz verzije 6 na verzijo 7 ali iz verzije 7 na verzijo 8 in potrebujejo pregled in podrobnejšo razlago, kaj sta ti verziji prinesli novega.

V petem poglavju bodo podrobneje opisani vsi trije profili Jave EE.

Šesto poglavje je namenjeno Java aplikacijskim strežnikom. Poglavje bo osredotočeno predvsem na aplikacijski strežnik WebSphere, ki ga uporablja veliko velikih podjetij. Opisane bodo njegove lastnosti ter primerjava njegovih zadnjih dveh verzij, in sicer verzije 8.5 in verzije 9.

2 Java

Java je programski jezik in platforma. Programski jezik Java je visokonivojski objektno usmerjen programski jezik, ki ima posebno sintakso in slog. Platforma Java pa je posebno okolje, v katerem se izvajajo aplikacije programskega jezika Java [1].

Obstajajo štiri platforme programskega jezika Java:

- Java Platform, Standard Edition
- Java Platform, Enterprise Edition
- Java Platform, Micro Edition (Java ME)
- JavaFX [1].

Vse platforme Jave sestavljajo navidezni stroj *(Virtual Machine – VM)* in aplikacijski programski vmesniki *(application programming interface – API)*. Navidezni stroj je program za določeno strojno in programsko opremo, ki izvaja aplikacije tehnologije Java. API pa je zbirka komponent programske opreme. Te lahko programer uporabi za ustvarjanje drugih komponent programske oprema ali aplikacij [1].

Vsaka platforma Java ponuja navidezni stroj in API. Prav to pa omogoča izvajanje aplikacij, ki so napisane za to platformo, na katerem koli združljivem sistemu z vsemi prednostmi programskega jezika Java. Te so neodvisnost od platforme, moč, stabilnost, enostavnost razvoja in varnost [1].

V naslednjih dveh podpoglavjih bosta podrobno opisani Java SE in Java EE. Java ME in JavaFX nista del teme diplomske naloge, zato ju bomo izpustili [1].

2.1 Java Platform, Standard Edition

Java je programski jezik in računalniška platforma, ki jo je prvič objavil Sun Microsystem leta 1995. Platforma je osnovna tehnologija, ki omogoča izvajanje programov Java, vključno s pripomočki, igrami in poslovnimi aplikacijami. Java se izvaja na več kot 850 milijonih osebnih računalnikov in na milijardah naprav po celem svetu. Med naprave se štejejo tudi mobilne in televizijske naprave. Po nekaterih internetnih statistikah pa obstaja okoli 9 milijonov

razvijalcev Jave, kar po TIOBE indeksu za junij 2018 znaša 15,368% vseh razvijalcev na svetu [2]. TIOBE indeks je indikator popularnosti programskega jezika in glede na trenutne rezultate kaže, da je Java najbolj popularen programski jezik. Java pa je sestavljena iz več ključnih komponent, ki kot celota ustvarjajo platformo Java [3].

Java SE je operacijsko okolje, ki je nameščeno na računalniku uporabnika in omogoča razvoj in namestitev aplikacij Java na namizja in strežnike [4]. Ko večina ljudi govori o programskem jeziku Java, mislijo na API Java SE. Le ta zagotavlja osnovno funkcionalnost programskega jezika Java [1].

Opredeljuje vse od osnovnih tipov in objektov programskega jezika Java do visoko nivojskih razredov, ki se uporabljajo za mreženje, varnost, dostop do baze podatkov, razvoj grafičnega uporabniškega vmesnika *(GUI)* in razčlenjevanje razširljivega označevalnega jezika *(Extensible Markup Language – XML)* [1].

Poleg osnovnih API-jev, je platforma Java SE sestavljena še iz navideznega stroja, razvojnih orodij in drugih knjižnic, razredov ter orodij. Ti se pogosto uporabljajo v aplikacijah tehnologije Java [1].

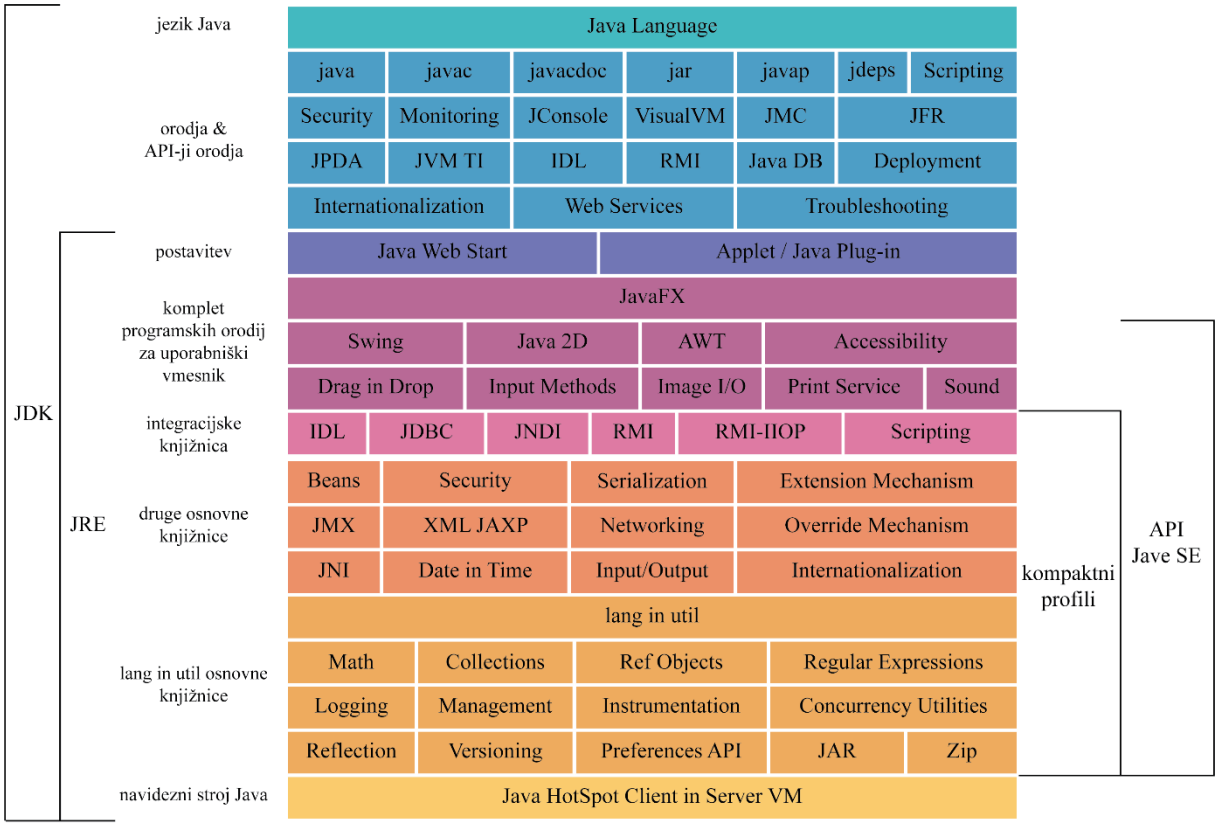
Oracle ponuja dva produkta za delo z Javo SE. To sta Java SE Runtime Environment (JRE) in Java SE Development Kit (JDK). JRE ponuja knjižnice, navidezni stroj Java in druge komponente, ki so potrebne za zagon *applet-ov* in aplikacij napisanih v Javi [7]. *Applet* je majhna odjemalska aplikacija Java, ki se izvaja na navideznem stroju Java v spletnem brskalniku [5]. JDK pa je nadgradnja JRE in vsebuje vse, kar je v JRE in še orodja, kot so prevajalniki in odpravljalniki napak. Ti so koristni ali potrebni za razvoj *applet-ov* in aplikacij [7].

Tabela 2.1 prikazuje primerjavo JRE in JDK. Za razumevanje tabele je potrebno razložiti kaj je *Web Start*. *Web Start* je programska oprema, ki omogoča prenos in zagon aplikacij Java s spleta [5].

JRE	JDK
Je implementacija navideznega stroja Java *(Java Virtual Machine – JVM)*, ki izvaja programe Java.	Je snop programske opreme, ki se lahko uporabi za razvoj aplikacij na osnovi Jave.
JRE je vtičnik, ki je potreben za izvajanje programov Java.	JDK je potreben za razvoj aplikacij
JRE je manjši od JDK, zato potrebuje manj prostora na disku.	JDK potrebuje več prostora na disku, saj vsebuje JRE skupa z različnimi razvojnimi orodji.
Vključuje JVM, jedrne knjižnice *(Core libraries)* in druge dodatne komponente za izvajanje *applet-ov* in aplikacij, ki so napisane v Javi.	Vključuje JRE, nabor razredov API, prevajalnik Java, *Web Start* in dodatne datoteke, ki so potrebne za pisanje oziroma razvoj Java *applet-ov* in aplikacij.

Tabela 2.1: Primerjava JRE in JDK [6]

Slika 2.1 oziroma konceptualni diagram na sliki podrobneje prikazuje komponente Oracle-ovih produktov Jave SE [7].



Slika 2.1: Komponente Oracle-ovih produktov Java SE [7]

V današnjem času želijo razvijalci omogočiti čim boljšo uporabniško izkušnjo. Posledično se tudi Java SE hitro razvija in posodablja že obstoječe komponente. Leta 2014 je bila izdana Java

SE 8, leta 2017 Java SE 9. Pred kratkim, marca 2018, pa je bila izdana Java SE 10. Diplomska naloga je osredotočena predvsem na Javo EE 7, ki temelji na Javi SE 8, in Javo EE 8, ki temelji na Javi SE 9, in je trenutno najnovejša različica Jave EE. Iz teh razlogov bodo v naslednjih podpoglavjih predstavljene novosti Jave SE 8 in Jave SE 9. Dober razvijalec mora spremljati novosti, zato bodo v podpoglavju opisane tudi novosti Jave SE 10, ki bo uporabljana v prihodnjih različicah Jave EE.

2.1.1 Novosti Java Platforme, Standard Edition 8

To poglavje povzema ključne novosti in posodobitve v Javi SE 8.

Nove funkcionalnosti v Java SE 8 so:

- anotacije na tipih *(Annotations on types)*,
- API za datum in čas *(Date and Time API)* in
- Lambda izrazi *(Lambda expressions)* [8].

Posodobljene funkcionalnosti:

- Java Database Connectivity (JDBC) RowSets,
- Java Management Extensions (JMX) Remote API,
- Java Compiler API,
- Streaming API for XML,
- Java API for XML Processing,
- JDBC 4.0 in
- Pluggable Annotation-Processing API [8].

V naslednjih podpoglavjih bodo podrobneje razložene le na novo dodane tehnologije.

2.1.1.1 Anotacije na tipih

Anotacije omogočajo, da izvorni kodi dodamo informacije o metapodatkih, čeprav niso del samega programa. Anotacije se začnejo s simbolom @, kateremu sledi ime anotacije. Prevajalnik prepozna anotacijo preko simbola @.

Anotacije so se pred to različico Jave SE lahko uporabljale le na deklaracijah tipa, kot so razredi, metode, definicije in spremenljivke. Sedaj pa so razširjene tudi za tista mesta, kjer se tipi uporabljajo, kot so na primer parametri.

Omogočajo tudi odkrivanje napak z vtičnim tipom pregledovalnikov *(pluggable type checkers)*. Slednji so na primer: stroj z ničelnim kazalcem napak, pogoji tekmovanja in drugi [8].

2.1.1.2 API za datum in čas

Nov API za datum, čas in koledar:

- Podpira standardne časovne koncepte:
  - parcialen, trajanje, obdobje, interval in
  - datum, čas, trenutek in časovni pas.
- Zagotavlja omejen nabor koledarskih sistemov ter je razširljiv za druge.
- Uporablja ustrezne standarde, vključno z ISO-8601, CLDR in BCP47.
- Temelji na eksplicitni časovni lestvici s povezavo na UTC [8].

2.1.1.3 Lambda izrazi

Anonimni razredi so izrazi, kar pomeni, da z njimi definiramo razred v drugem izrazu. Kodo naredijo bolj jedrnato in omogočajo istočasno deklaracijo in instanciranje razreda. So kot lokalni razredi, le da nimajo imena.

Ena izmed težav anonimnih razredov je ta, da če je anonimni razred zelo enostaven, se lahko sintaksa anonimnih razredov zdi zelo težka in nejasna. Primer takšnega enostavnega razreda je vmesnik, ki vsebuje samo eno metodo. V takšnih primerih programer običajno želi podati funkcionalnost kot argument druge metode. Primer je na primer akcija, ki bi se morala zgoditi, ko nekdo klikne na gumb. Lambda izrazi omogočajo prav to, da se funkcionalnost obravnava kot argument metode ali pa kodo obravnava kot podatke.

Če zgoraj opisano zapišemo v dveh alinejah, lahko rečemo, da lambda izrazi v Javi zagotavljajo anonimne tipe funkcij, ki:

- Zamenjajo uporabo anonimnih notranjih razredov.
- Zagotavljajo bolj funkcionalen stil programiranja v Javi [9].

2.1.2 Novosti Java Platforme, Standard Edition 9

Platforma Java SE verzije 9 je zelo pomembna izdaja. To poglavje povzema ključne izboljšave v Javi SE 9.

Spodaj so naštete ključne spremembe Jave SE 9, ki pa vplivajo na več kot eno tehnološko področje.

- Modulni sistem platforme Java *(Java Platform Module System).*
- Nova shema različice nizov [10].

V naslednjih podpoglavjih bodo podrobneje opisani ti dve ključni spremembi.

2.1.2.1 Modulni sistem platforme Java

Uvaja novo vrsto Java programerske komponente, ki se imenuje modul. Ta je poimenovana, samo-opisna kolekcija kode in podatkov. Ta modulni sistem:

- Uvaja novo neobvezno fazo vezni čas, ki se nahaja med časom prevajanja in časom izvajanja. Med tem časom se lahko množica modulov sestavi in optimizira v sliko izvajanja *(runtime image)* po meri.
- Doda možnosti orodjem: *javac*, *jlink* in *java*. V njih se lahko določijo poti modulov, ki locirajo definicije modulov.
- Uvaja modularno datoteko JAR. Ta je datoteka JAR, z datoteko *module-info.class* v svojem korenskem imeniku.
- Uvaja JMOD format, ki je format za pakiranje. Podoben je JAR-u, vendar lahko vsebuje tudi izvorno kodo in konfiguracijske datoteke [10].

JDK je bil razdeljen na sklop modulov. Ta sprememba:

- Omogoča združevanje modulov JDK-ja v različne konfiguracije.
- Prestrukturira slike izvajanja JDK in JRE, da prilagajajo module in izboljšajo učinkovitost, varnost in vzdržljivost.
- Določa novo shemo URI za poimenovanje modulov, razredov in virov. Ti so shranjeni v sliki izvajanja, brez razkrivanja notranje strukture ali formata slike.
- Odstrani mehanizem za preklic podprtih standardov in mehanizem za razširitev.
- Odstrani *rt.jar* in *tools.jar* iz Java slike izvajanja.
- Naredi večino notranjih API-jev JDK-ja privzeto nedostopnih. Nekaj kritičnih, široko uporabljenih notranjih API-jev pa pusti dostopnih, dokler ne obstajajo podprte zamenjave za vse ali večino njihovih funkcionalnosti [10].

2.1.2.2 Nova shema različice nizov

Omogoča poenostavljen format različice nizov. Ta omogoča jasno razlikovanje med izdajami velikih, manjših, varnostnih in popravljenih posodobitev.

Nov format različice nizov je sledeč:

- *$MAJOR* je številka različice, ki se poveča ob večji izdaji.
- *$MINOR* je številka različice, ki se poveča ob vsaki manjši posodobitvi.
- *$SECURITY* je številka različice, ki se poveča ob izdaji varnostne posodobitve, ki vsebuje kritične popravke. Med njih štejemo tudi tiste popravke, ki so potrebni za izboljšanje varnosti.

- *$PATCH* je številka različice, ki se poveča ob izdaji, ki vsebuje varnostne in prednostne popravke odjemalcev, ki so bili testirani skupaj [10].

2.1.3 Novosti Java Platforme, Standard Edition 10

Ta izdaja nadaljuje razvoj platforme in s tem zagotavlja širši uspeh jedrne tehnologije Java. Ob tem pa s poenostavljanjem definicije spremenljivk v kodi Java in drugimi novostmi izboljšuje izkušnje razvijalca [11].

Novosti Jave SE 10, so sledeče:

- Sklepanje tipa lokalnih spremenljivk *(Local-Variable Type Inference).*
- Krepitev gozda JDK v enojni repozitorij *(Consolidate the JDK Forest into a Single Repository).*
- Vmesnik zbiralnika smeti *(Garbage-Collector).*
- Poln vzporedni zbiralnik smeti za Garbage First zbiralnik smeti *(Parallel Full Garbage Collector for Garbage First).*
- Delitev podatkov razreda aplikacij *(Application Class-Data Sharing).*
- Lokalno usklajevanje niti *(Thread-Local Handshakes).*
- Odstranitev orodja za generiranje izvornih glav *(Remove the Native-Header Generation Tool – javah).*
- Dodatne razširitve oznak jezika *unikode (Additional Unicode Language-Tag Extensions).*
- Alokacija kopice na alternativne pomnilniške naprave *(Heap Allocation on Alternative Memory Devices).*
- Eksperimentalni na Javi temeljen just-in-time *(JIT)* prevajalnik *(Experimental Java-Based JIT Compiler).*
- Izvorni certifikati *(Root Certificates).*
- Na času temeljeno označevanje novih verzij izdaj *(Time-Based Release Versioning)* [11].

Z vidika razvijalcev so tri izmed zgoraj naštetih novosti najbolj zanimive in bodo podrobneje razložene v naslednjih podpoglavjih. Te so: Local-Variable Type Inference, Parallel Full GC for G1 in Experimental Java-Based JIT Compiler.

2.1.3.1 Sklepanje tipa lokalnih spremenljivk

Razvijalci se pogosto pritožujejo glede stopnje šablonskega kodiranja, ki je zahtevana v Javi. Manifestni tip deklaracij za lokalne spremenljivke so pogosto dojemajo kot nepotrebni. Velikokrat pa je že iz njihovih dobrih imen vse razvidno. Manifestni tipi so v Javi na primer

List in *Hasmap*. Potreba po zagotavljanju manifestnih tipov za vsako spremenljivko pa razvijalce spodbuja, da uporabljajo preveč zapletene izraze. Skoraj vsi ostali popularni statično tipizirani programski jeziki podpirajo neko obliko lokalno spremenljivih tipov sklepanja, na primer: C++ (*auto*), C# (*var*) i in Scala (*var/val*). Zato je Java skoraj edini takšen programski jezik, ki še ni sprejela lokalno spremenljiv tip sklepanja [11].

Ta slabost je odpravljena v Javi SE 10, ki dodaja lokalno spremenljivi tip sklepanja, ki omogoča preskok deklaracije tipa za lokalne spremenljivke in *for* zanke, kar pomeni, da lahko sedaj programerji namesto *List list = new ArrayList()* pišejo kar *var list = new ArrayList()*. Torej programer namesto manifestnih tipov piše *var*, prevajalnik pa sam ugotovi kakšnega tipa je spremenljivka. Pomembno je še enkrat omeniti, da je takšen način deklaracije omogočen le za lokalne spremenljivke in *for* zanke [11].

2.1.3.2 Poln vzporedni zbiralnik smeti za G1 zbiralnik smeti

Za razumevanje te novosti moramo naprej razložiti pomen zbiralnika smeti *(garbage collector – GC))* in Garbage-first (G1) zbiralnika smeti.

Začnimo z GC. Samodejno zbiranje smeti je proces pregledovanja pomnilnika kopice. Kar pomeni prepoznavanje, kateri predmeti so in kateri niso v uporabi ter brisanje neuporabljenih predmetov. Objekt v uporabi ali objekt z referenco je tisti, na katerega del programa še vedno ohranja kazalec. Neuporabljen objekt ali objekt brez reference pa je tisti na katerem program nima več nobenih referenc. Tako je lahko neuporabljen objekt, ki se nahaja v pomnilniku sproščen iz njega [3].

V programskem jeziku kot je C, je dodeljevanje in osvoboditev pomnilnika ročni postopek. V Javi pa proces osvoboditve pomnilnika samodejno obravnava GC [3].

Sedaj pa razložimo še pomen G1. G1 GC je GC, ki je usmerjen v večprocesorske stroje z veliko pomnilnika. Prizadeva si zagotoviti najboljše ravnovesje med zakasnitvijo in prepustnostjo z uporabo trenutnih ciljnih aplikacij in okolij [12].

G1 GC je postal privzet v Java SE 9, ki je predstavljena v poglavju 2.1.2. Predhodno privzeti GC je bil vzporedni zbiralnik, ki je imel vzporedno poln GC in je bil namenjen aplikacijam s podatkovnimi nizi srednjih in velikih velikosti in ki se izvajajo na večprocesorskih ali večnitnih strojih. Poln GC je tisti GC, ki omogoča začasno zamrznitev izvajanja tekočih programov in omogoča GC ekskluziven dostop do procesorja JVM-ja in pomnilnika. Za zmanjšanje vpliva na uporabnike, ki imajo izkušnje s polnim GC-ji, je bilo potrebno narediti vzporeden tudi G1 poln GC [13].

G1 GC je zasnovan tako, da se izogiba polnim zbirkam. Vendar, ko sočasne zbirke ne morejo povrniti polnilnika dovolj hitro, se pojavi padec *(fall back)* polnega GC-ja. Izvedba polnega GC-ja za G1 uporablja enojno-nitni algoritem. Razvijalci JDK-ja so ta algoritem poskušali narediti vzporeden in uporabiti enako številno niti, kot uporabljajo zbirke Young in Mixed. Kar jim je tudi uspelo in je omogočeno v Jave SE 10. Število niti je mogoče nadzorovati z možnostjo: *-XX:ParalelGCThreads*, a bo ta možnost vplivala tudi na število niti uporabljenih za zbirke Young in Mixed [13].

2.1.3.3 Eksperimentalni na Javi temeljen JIT prevajalnik

Za razumevanje te novosti moramo najprej razložiti pojma prevajalnik in JIT prevajalnik. Računalniki ali natančneje centralne procesne enote (CPE) lahko izvajajo le relativno malo specifičnih navodil. Ta se imenujejo zbirne ali binarne kode. Vsi programi, ki jih CPE izvede, morajo iz teg razloga biti prevedeni v ta navodila [13].

Programski jeziki, kot sta na primer C++ in Fortran, se imenujejo prevedeni programski jeziki. Tako so poimenovani, zato ker so njihov programi dostavljeni kot binarno (prevedena) koda. To pomeni, da statični prevajalnik napisan program prevede v binarno obliko [13].

Po drugi strani pa obstajajo tudi programski jeziki, kot sta PHP in Perl, ki so interpretirani. Pri njih se lahko ista programska izvorna koda zažene na kateremkoli CPE, vse dokler ima naprava pravilen interpreter (na primer programa php ali perl). Interpreter vsako vrstico programa prevede v binarno kodo, ko se izvrši ta vrstica [13].

Tako prevajalnik kot tudi interpreter imata svoje prednosti in slabosti. Java pa poskuša najti srednjo pot med njima. Aplikacije Java so prevedene, a so prevedene v idealiziran zbirni jezik, namesto da bi bila prevedene v binarno kodo za specifičen CPE. Binarna Java (*java binary*) nato izvaja ta zbirni jezik (imenovan Java bytecodes). Zaradi tega je platforma Java neodvisna od interpretiranega jezika. Platforma Java izvaja idealizirano binarno kodo, zato lahko program Java, ko se koda izvede, prevede kodo v platformo v binarno obliko. JIT prevajalnik pa se imenuje zato, ker se takšno prevajanje pojavi, ko se program izvede in se zgodi »ravno v času« *(»just in time«)*. Pomembno je omeniti, da je JIT prevajalnik srce JVM-ja, saj najbolj vpliva na njegovo učinkovitost [13].

Graal je na Javi zasnovan JIT prevajalnik in je osnova eksperimentalnega naprednega *(Ahead-of-Time – AOT)* prevajalnika predstavljenega v Javi SE 9. Omogočiti mu, da se uporablja kot eksperimentalni JIT prevajalnik in je naslednji korak pri preučevanju izvedljivosti na javi zasnovanega JIT-a za JDK [11].

V Javi SE 10 pa se je zgodilo ravno to, saj je Graal-u omogočeno, da je postal eksperimentalni JIT prevajalnik na Linux/x64 platformi. Graal bo uporabljal vmesnik JVM prevajalnika *(JVM compiler interface – JVMCI)*, ki je bil prestavljen v JDK 9. Ker je Graal že v JDK-ju, je njegovo omogočanje, da se uporabi kot eksperimentalni JIT primarno prizadevano za testiranje in odstranjevanje napak [11].

2.2 Java Platform, Enterprise Edition

Obljuba, »Razvij enkrat, izvajaj povsod!« *(»Write once, Run Anywhere!«)* je bila ključna gonilna sila za uspeh, ki ga je deležen programski jezik Java. Z leti je zanimanje za Javo vse bolj naraščalo, posledično pa se je povečala potreba po robustnih poslovno razrednih aplikacijah. Pojav interneta in prvih različic brskalnikov pa je privedel do implementacije prvega spletnega strežnika v Javi, kot tudi do uvedbe Servelt in JavaServer Pages (JSP) specifikacij, ki bosta podrobneje razloženi pozneje. Prav ti dve specifikaciji sta postali temelj platforme Java 2 Enterprise Edition (J2EE). Od leta 1999 do 2003 pa se je število Java Specification Request (JSR) povečalo iz 10 na 18. JSR je zahtevek za specifikacijo Java. JSR je potrebno predložiti, ko se želi uvesti nova tehnologija ali nova verzija tehnologije. Leta 2006 je bila platforma preimenovana v Java Enterprise Edition (Java EE), ki je prav tako prinesla novo številčenje verzij, ki se je začelo z Javo EE 6 [14]. Java EE je z leti postala platforma, ki zagotavlja API in izvajalno okolje za razvijanje in izvajanje obsežnih, več-nivojskih, razširljivih, zanesljivih in varnih omrežnih aplikacij [1].

Podjetja so Javo EE zaradi številnih prednosti, ki jih je obljubljala, sprejeli zgodaj. Te so na primer: skalabilnost, transakcijski in standardizirani programski model, visoka pretočnost in zanesljivost operacij. Vendar pa je vsaka obljuba imela kakšno pomanjkljivost, zato je trajalo kar nekaj časa, da je platforma kot specifikacija vključevala operativno in razvijalsko zmogljivost [14].

Iz zgoraj opisanega vidimo, da je Java EE poslovni standard, ki je namenjen izdelavi prilagodljivih poslovnih storitev. Platforma Java EE je zgrajena na platformi Java SE, ki je bila podrobneje razložena v poglavju 2.1. Java EE je sestavljena iz množice storitev, API-jev in protokolov, ki zagotavljajo funkcionalnosti za razvoj več-nivojskih aplikacij [6]. Njen namen je, da razvijalcem poslovnih aplikacij zagotavlja močan nabor API-jev. Z njihovo pomočjo pa skrajša razvojni čas, zmanjša kompleksnost aplikacij in izboljšuje njeno učinkovitost. API je množica rutin, protokol, funkcij in/ali ukazov. Te programerji uporabljajo za olajševanje interakcije med različnimi sistemi ali razvoj programske opreme [5].

Java EE je razvit z uporabo Java Community Process (JCP), s prispevki strokovnjakov iz industrije, komercialnih in odprtokodnih organizacij, uporabniških skupin Java in številnih posameznikov. JCP je proces, ki standardizira in formalizira tehnologije Java. V tem procesu sodelujejo zainteresirane stranke, kot so podjetja in razvijalci, da razvijejo platformo. Vsaka izdaja platforme Java EE vključuje nove funkcionalnosti. Te se prilagajajo potrebam industrije, izboljšujejo prenosljivost aplikacij in povečujejo produktivnost razvijalcev [1].

Ker se diplomska naloga osredotoča na Javo EE 7 in 8, so za lažjo predstavo sestave Jave EE, v tabeli 2.2 prikazane vse tehnologije, ki jih vsebujeta platformi Java EE 7 in 8. Za razumevanje tabele je potrebno razložiti pomen spletnega profila *(Web Profile-a)* in polnega profila *(Full Profile-a)*.

Java EE vsebuje tri profile: poln profil, spletni profil in MicroProfile, ki bodo v petem poglavju tudi podrobneje razloženi. Za sedaj pa samo na kratko razložimo poln in spletni profil. Poln profil je profil Jave EE, ki zajema vse tehnologije specifikacije Java EE. Spletni profil pa je profil Java EE, ki zajema tehnologije platforme Java EE za izdelavo spletnih aplikacij.

Platforma Java EE	**Java Platform, Enterprise Edition 7 in 8**						
	Ime tehnologije	**Verzije v Javi EE 7**	**Verzije v Javi EE 8**	**Opis**	**JSR**	**Spletni profil**	**Poln profil**
Tehnologije za spletne aplikacije	WebSocket	1.0	1.1	Java API for WebSocket	356	✓	✓
	JSON-B	×	1.0	Java API for JSON Binding	367	✓	✓
	JSON-P	1.0	1.1	JavaScript Serialization Object Notation Protocol	353	✓	✓
	Servlet	3.1	4.0	Java Servlet	340	✓	✓
	JSF	2.2	2.3	JavaServer Faces	344	✓	✓
	EL	3.0	3.0	Unified Expression Language for configuration of web components and context dependency injection	341	✓	✓
	JSP	2.3	2.3	JavaServer Pages	245	✓	✓
	JSTL	1.2	1.2	Standard Tag Library for JavaServer Pages	052	✓	✓

Platforma Java EE	Java Platform, Enterprise Edition 7 in 8						
	Ime tehnologije	**Verzije v Javi EE 7**	**Verzije v Javi EE 8**	**Opis**	**JSR**	**Spletni profil**	**Poln profil**
Tehnologije za poslovne aplikacije	Batch Process	1.0	1.0	Batch Applications for the Java Platform	352	×	✓
	Concurrency Utilities	1.0	1.0	Concurrency Utilities for the Java platform	236	×	✓
	CDI	1.1	2.0	Contexts and Dependency Injection for Java	346	✓	✓
	DI	1.0	1.0	Dependency Injection for Java	330	✓	✓
	Bean Validation	1.1	2.0	Bean Validation framework	349	✓	✓
	EJB	3.2	3.2	Enterprise JavaBeans	345	✓	✓
				EJB entity beans and associated EJB QL		✓	○
	Interceptors	1.2	1.2	Interceptor technology	318	✓	✓
	JCA	1.7	1.7	Java EE Connector Architecture	322	×	✓
	JPA	2.1	2.2	Java Persistence API	338	✓	✓
	Common Annotations	1.2	1.3	Common Annotations for the Java Platform	250	✓	✓
	JMS	2.0	2.0	Java Message Service API	343	×	✓
	JTA	1.2	1.2	Java Transaction API	907	✓	✓
	JavaMail	1.5	1.6	JavaMail	919	×	✓

Platforma Java EE	Java Platform, Enterprise Edition 7 in 8						
	Ime tehnologije	**Verzije v Javi EE 7**	**Verzije v Javi EE 8**	**Opis**	**JSR**	**Spletni profil**	**Poln profil**
Tehnologije za spletene storitve	JAX-RS	2.0	2.1	Java API for RESTful Web Services	339	✓	✓
	Implementing Enterprise Web Services	1.3	1.3	Implementing Enterprise Web Services	109	×	
	JAX-WS	2.2	2.2	Java API for XML-Based Web Services including SOAP and WSDL	224	×	✓
	Web Services Metadata	2.1	2.1	Web Services Metadata for the Java Platform	181	×	✓
	JAX-RPC	1.1	1.1	Java API for XML-Based RPC	101	×	○
	JAXM	1.3		Java APIs for XML Messaging	067	×	
	SAAJ	1.3	1.3	SOAP with Attachments API for Java Specification	067	×	
	JAXR	1.0	1.0	Java API for XML Registries	093	×	○

Platforma Java EE	Java Platform, Enterprise Edition 7 in 8						
	Ime tehnologije	**Verzije v Javi EE 7**	**Verzije v Javi EE 8**	**Opis**	**JSR**	**Spletni profil**	**Poln profil**
Tehnologije za upravljanje in varnost	Java EE Security API	×	1.0	Java EE Security API	375	✓	✓
	JASPIC	1.1 M/B	1.1	Java Authentication Service Provider Interface for Containers	196	×	✓
	JACC	1.5	1.5	Java Authorization Contract for Containers	115	×	×
	Java EE Application Deployment	1.2	1.2	Java EE Application Deployment	088	×	○
	J2EE Management	1.1	1.1	J2EE Management	077	×	✓
	Debugging support	1.0	1.0	Debugging Support for Other Languages	045	✓	✓
Java SE tehnologije za povezavo z Java EE aplikacijami	JAXB	2.2	2.2	Java Architecture for XML Binding	222	×	✓
	JAXP	1.3	1.6	Java API for XML Processing	206	×	
	JDBC	4.0	4.0	Java Database Connectivity	221	×	
	JMX	2.0	2.0	Java Management Extensions	003	×	
	JAF	1.1	1.1	JavaBeans Activation Framework	925	×	
	StAX	1.0	1.0	Streaming API for XML	173	×	

Tabela 2.2: Tehnologije v platformi Jave EE 7 in 8 in prikaz v katere profile spadajo tehnologije [15], [16].

Danes Java EE ponuja bogato programsko platformo za podjetja, z več kot 20 različnimi izvedbami Jave EE, med katerimi lahko izbirajo [1]. V diplomski nalogi se bomo osredotočili na Javo EE 7 in 8. V naslednjih dveh podpoglavjih bodo opisane njune novosti.

Pomembno je tudi omeniti, da platforma Java EE uporablja za poslovne aplikacije porazdeljen več-nivojski model aplikacije, kar pa bo podrobneje razloženo v tretjem poglavju.

2.2.1 Novosti Java Platforme, Enterprise Edition 7

Java EE 7 je bila izdana leta 2013 in je prinesla kar nekaj novosti in posodobitev, ki so naštete spodaj in bodo zaradi obsežnosti podrobneje opisane v četrtem poglavju.

Nove specifikacije v Java EE 7 so:

- WebSocket,
- JavaScript Object Notation Protocol (JSON-P),
- Concurrency Utilities in
- Batch Process [16]

Posodobljene specifikacije so:

- Servlet 3.1,
- Java Server Faces (JSF) 2.2,
- Expression Language (EL) 3.0,
- Java Server Pages (JSP) 2.3,
- Contexts and Dependency Injection (CDI) 1.1,
- Bean Validation 1.1,
- Enterprise JavaBeans (EJB) 3.2,
- Interceptors 1.2,
- Java EE Connector Architecture (JCA) 1.7,
- Java Persistence API (JPA) 2.1,
- Common Annotations 1.2,
- Java Message Service API (JMS) 2.0,
- Java Transaction API (JTA) 1.2,
- JavaMail 1.5,
- Java API for RESTful Web Services (JAX-RS) 2.0,
- Java Authentication Service Provider Interface for Containers (JASPIC) 1.1 in
- Java Authorization Contract for Containers (JACC) 1.5 [16]

Java EE 7 nadaljuje temo, ki se je začela v zgodnejših verzijah platforme, in sicer izboljševanje enostavnosti razvoja [16].

Ta različica Jave EE se je uporabljala več kot pet let, nato pa je leta 2017 prišla nova različica, in sicer Java EE 8. Njene novosti bodo podrobneje opisane v naslednjem podpoglavju.

2.2.2 Novosti Java Platforme, Enterprise Edition 8

Težko pričakovana izdaja Java EE verzije 8 je prva objava poslovne platforme Java od junija 2013. Oracle je strateško prerazporedil Javo EE. To je storil s poudarjanjem tehnologij, ki podpirajo računalništvo v oblaku, mikrostoritve in odzivno programiranje. Slednje je sedaj tkano v tkanine številnih API-jev Jave EE, izmenjevani format JSON pa podprt v jedru platforme.

Novi funkcionalnosti v Java EE 8 sta:

- Java API for JSON Binding (JSON-B) in
- Java EE Security API.

Posodobljene specifikacije so:

- Servlet 4.0,
- Bean Validation 2.0,
- CDI 2.0,
- WebSocket 1.1,
- JSON-P 1.1,
- JSF 2.3,
- JPA 2.2,
- Common Annotations 1.3,
- JavaMail 1.6,
- JAX-RS 2.1 in
- Java API for XML Processing (JAXP) 1.6 [17].

Nove in posodobljene specifikacije bodo zaradi obsežnosti podrobneje opisane v četrtem poglavju.

Pred izidom Jave EE 8 se je govorilo, da je ta polovica dvodelne izdaje, ki se bo zaključila z Javo EE 9. Nato pa se je zgodilo to, da je Javo EE od Oracle-a prevzel Eclipse Foundation in tako postal njen nov upravitelj. Oracle je izdal tudi predlog za sprejetje Jave. Del enega izmed

predlogov je bilo novo ime za Javo EE. Po izvedbi glasovanja za novo ime, so se odločili, da se bo imenovala Jakarta EE. Tako lahko pričakujemo nove izdaje Jave EE pod tem imenom.

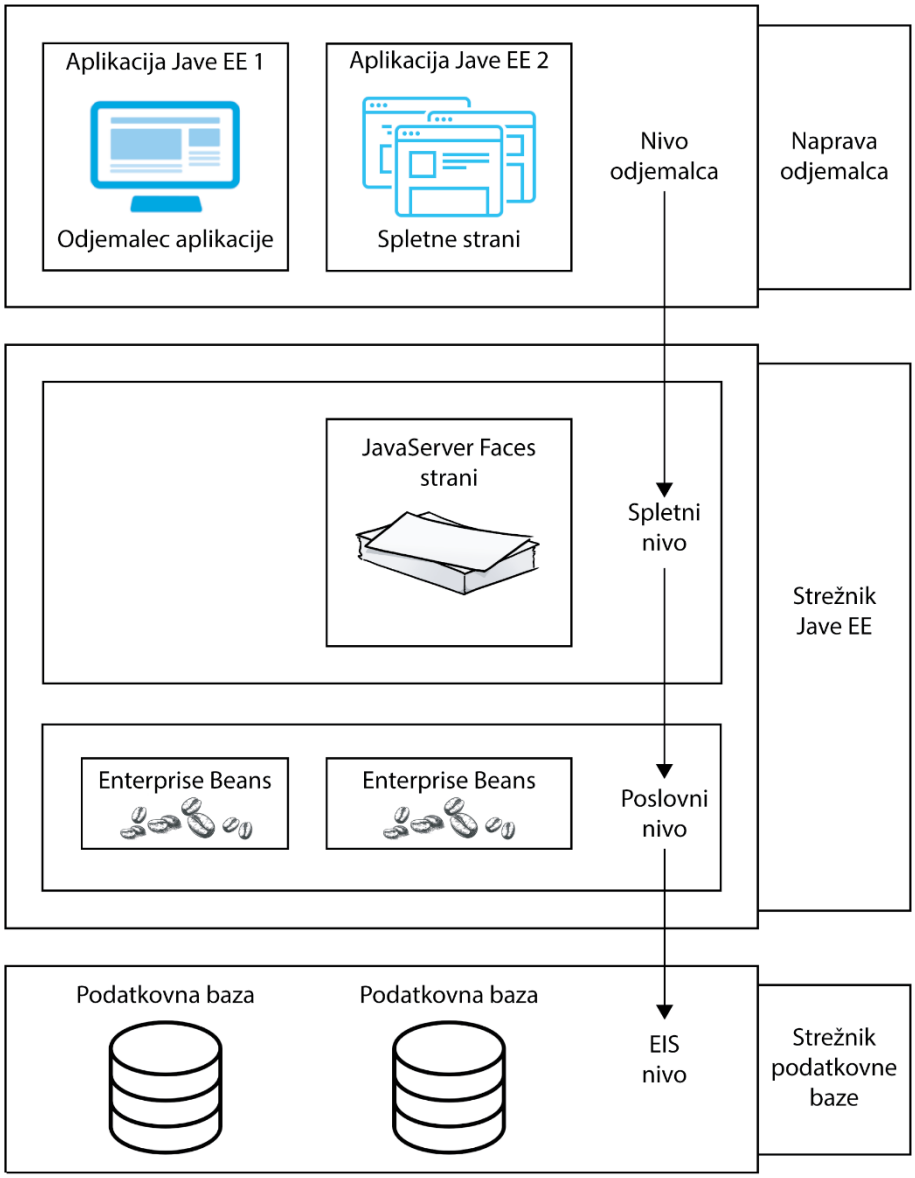
Sledi poglavje o več-nivojskih poslovnih aplikacijah, ki so bile omenjene v uvodu, ter njihovi implementaciji z uporabo Jave EE.

3 Več-nivojske poslovne aplikacije ter njihova implementacija z uporabo Jave EE

Java EE za poslovne aplikacije uporablja porazdeljen več-nivojski model aplikacij, kar pomeni, da z njo implementiramo več-nivojske poslovne aplikacije. Aplikacijska logika je razdeljena na komponente po funkciji, te pa so umeščene v ustrezne nivoje. Komponente so samostojne funkcionalne programske enote, ki so s svojimi povezanimi razredi in datotekami sestavljene v aplikacijo Java EE. Komponente tudi komunicirajo z drugimi komponentami. Nameščene so na različne stroje, odvisno od nivoja v več-nivojskem okolju Java EE, v katerega spada komponenta aplikacije [1].

Več-nivojske Java EE aplikacije delimo na spodaj naštete nivoje in njihove komponente [1]. Komponente v oklepajih so Java EE komponente, definirane po specifikaciji Java EE. Slika 3.1 prikazuje spodaj naštete nivoje.

- Komponente odjemalskega nivoja (aplikacijski odjemalec in *applet-i*), ki se izvajajo na stroju odjemalca.
- Komponente spletnega nivoja (komponente tehnologij Java Servlet, JSF, JSP in druge), ki se izvaja na strežniku Java EE.
- Komponente poslovnega nivoja (EJB komponente in druge), ki se izvaja na strežniku Java EE.
- Programska oprema nivoja informacijskega sistema za upravljanje *(enterprise information system – EIS)*, ki se izvaja na strežniku EIS [1].



Slika 3.1: Več-nivojske aplikacije [1]

Čeprav aplikacijo Java EE lahko sestavljajo vsi štirje nivoji, se aplikacije Java EE običajno štejejo kot tri-stopenjske aplikacije. Med tri-stopenjske se štejejo, zato ker so razdeljene na tri lokacije:

- Odjemalske naprave
- Strežnik Jave EE
- Baze podatkov [1]

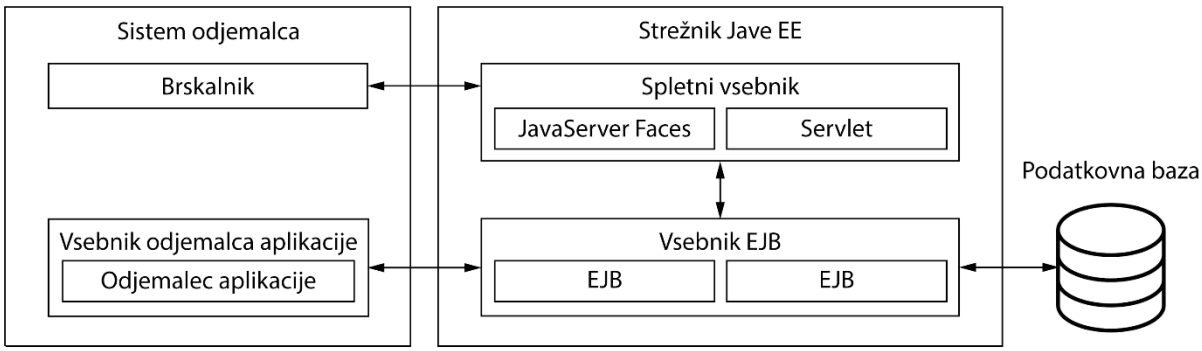
Tri-nivojske aplikacije, ki so sestavljene na ta način, razširjajo dvo-nivojski model odjemalec-strežnik. To je narejeno tako, da se namesti več nitni aplikacijski strežnik med odjemalsko

aplikacijo in zalednim skladiščenjem [1]. Na sliki 3.1 je razvidno zgoraj opisano. Vsi trije nivoji morajo med seboj komunicirati. To komunikacijo poenostavljajo odprti, standardni protokoli in izpostavljeni API-ji [18].

Za razumevanje aplikacij Java EE je potrebno razložiti tudi pomen vsebnikov Jave EE. Ti so vmesniki med komponento in za platformo specifično nizko nivojsko funkcionalnostjo, ki podpira komponento. Preden se katera koli izmed komponent izvede, mora biti sestavljena v modul Java EE in naložena v svoj vsebnik. Obstajajo štirje tipi vsebnikov: vsebnik EJB, spletni vsebnik, vsebnik odjemalca aplikacij in *applet* vsebnik [1].

Applet vsebnik upravlja izvedbo *applet-ov*. Sestavljen je iz spletnega brskalnika in vtičnika Java, ki se izvajata skupaj na odjemalcu [1].

Sedaj lahko aplikacijski model Java EE prikažemo tako, kot kaže slika 3.2.



Slika 3.2: Aplikacijski model Java EE [1]

V naslednjih podpoglavjih bo opisan vsak nivo aplikacijskega modela Jave EE in njegovi vsebniki. V četrtem poglavju, ki opisuje posodobljene in nove tehnologije Jave EE 7 in 8, bo pri vsaki tehnologiji napisano tudi v kateri vsebnik spada. Ker se nekatere tehnologije pojavijo v več vsebnikih, bo zato pri nekaterih tehnologijah naštetih več vsebnikov. *Applet* vsebnik ne bo omenjen, zato je bil na kratko opisan že v tem poglavju [1].

3.1 Nivo odjemalca

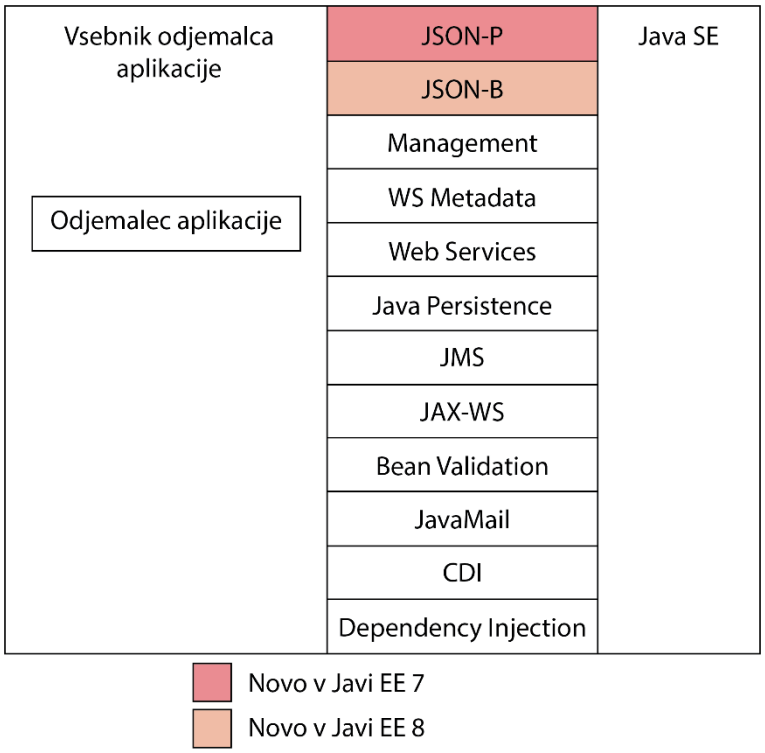
Nivo odjemalca je sestavljen iz ene ali več komponent odjemalca in je odgovoren za predstavitev in interakcijo uporabnika z aplikacijo. Komponente odjemalca omogočajo uporabniku, da na varen in intuitiven način komunicira s procesi na strežniku Jave EE. Aplikacijski strežnik WebSphere je eden izmed Java EE strežnikov na katerega se bomo osredotočili v tej diplomski nalogi in podpira več vrst odjemalcev [5]. Odjemalci so lahko

spletni brskalnik, samostojna aplikacija ali server in se ne izvajajo na istem stroju kot strežnik Jave EE [1]. Pomembno je omeniti, da odjemalci neposredno ne dostopajo do storitev podatkovne baze. Nivo odjemalca ponuja dve vrsti vsebnikov, in sicer vsebnike odjemalcev aplikacij, ki bo podrobneje opisan v podpoglavju, in *applet* vsebnike [5].

3.1.1 Vsebnik odjemalca aplikacije

Vsebnik odjemalca aplikacije *(Application Client Container)* je vmesnik med odjemalci aplikacij Jave EE in strežnikom Jave EE. Pomembno je omeniti, da so odjemalci aplikacij Jave EE posebne aplikacije Java SE, ki uporabljajo komponente strežnika Jave EE. Vsebnik odjemalca aplikacij teče na stroju odjemalca in je prehod med aplikacijo odjemalca in komponentami strežnika Jave EE, ki jih uporablja odjemalec [1].

Slika 3.3 prikazuje zahtevane tehnologije v vsebniku odjemalca aplikacije. Z rozasto barvo so obarvana tiste tehnologije, ki so na novo dodane v Javi EE 7, z bež barvo pa tiste, ki so na novo dodane v Javi EE 8. Opcijski tehnologiji, ki nista prikazani na sliki, a tudi spadata sem, sta Java Authentication Service Provider Interface for Containers (JAX-RPC) in Java API for XML Registries (JAXR).



Slika 3.3: API-ji Java EE v vsebniku odjemalca aplikacije [5]

3.2 Nivo strežnika

Nivo strežnika se običajno imenuje kar nivo poslovne logike aplikacije. Procesi na tem nivoju upravljajo poslovno logiko aplikacije in dovoljujejo dostop do tretjega nivoja, t. j. podatkovne baze. Na tem nivoju je definirana večina poslovnih procesov. Ker komponente odjemalca lahko istočasno dostopajo do procesov na strežniku, mora ta nivo upravljati tudi lastne transakcije [5].

Aplikacijski strežnik Jave EE ponuja dve vrsti vsebnikov, ki ju opisujeta naslednji dve podpoglavji.

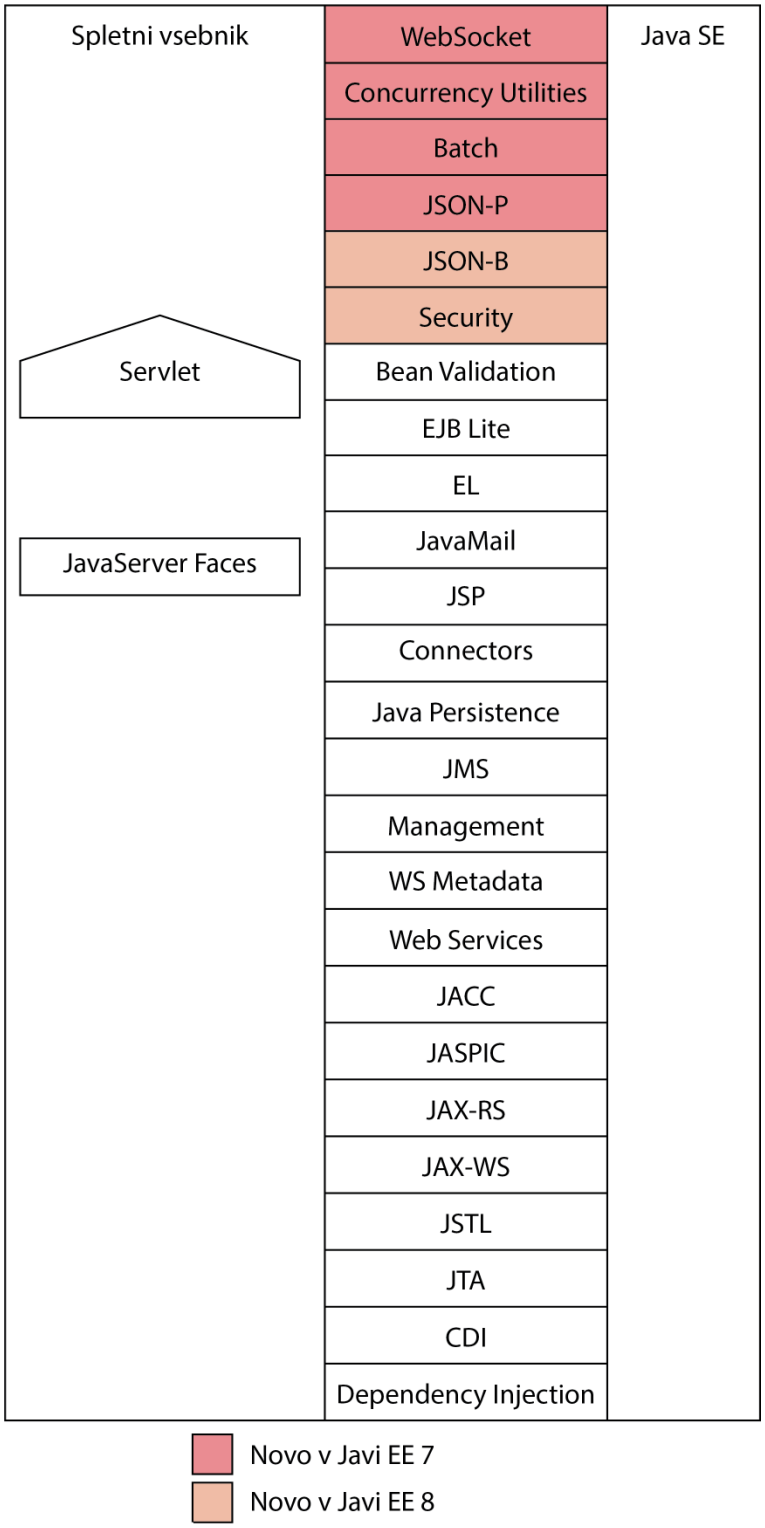
3.2.1 Spletni vsebnik

Spletni vsebnik *(Web Container)*, včasih imenovan tudi servletni vsebnik, je vmesnik med spletnimi komponentami in spletnim strežnikom ter je sestavljen iz komponent, ki obravnavajo interakcijo med odjemalci in poslovim nivojem oziroma vsebnikom EJB. Njegove primarne naloge so naslednje:

- Dinamično ustvari vsebino za odjemalce, v različnih formatih.
- Zbere podatke od uporabnikov vmesnika odjemalca in vrne ustrezne rezultate iz komponent poslovnega nivoja.
- Nadzira pretok zaslonov ali strani na odjemalcu.
- Ohranja stanje podatkov za uporabniško sejo.
- Izvede nekaj osnovne poslovne logike in začasno shrani nekatere podatke v komponente JavaBean. Slednje so objekti, ki delujejo kot začasne shrambe *(data stores)* za strani aplikacije [1].

Kot je že omenjeno, se spletne komponente in njihov vsebnik izvajajo na strežniku Java EE [1].

Slika 3.4 prikazuje zahtevane tehnologije v spletnem vsebniku. Z rozasto barvo so obarvana tiste tehnologije, ki so na novo dodane v Javi EE 7, z bež barvo pa tiste, ki so na novo dodane v Javi EE 8. Opcijski tehnologiji, ki nista prikazani na sliki, a tudi spada tukaj sta JAX-RPC in JAXR.

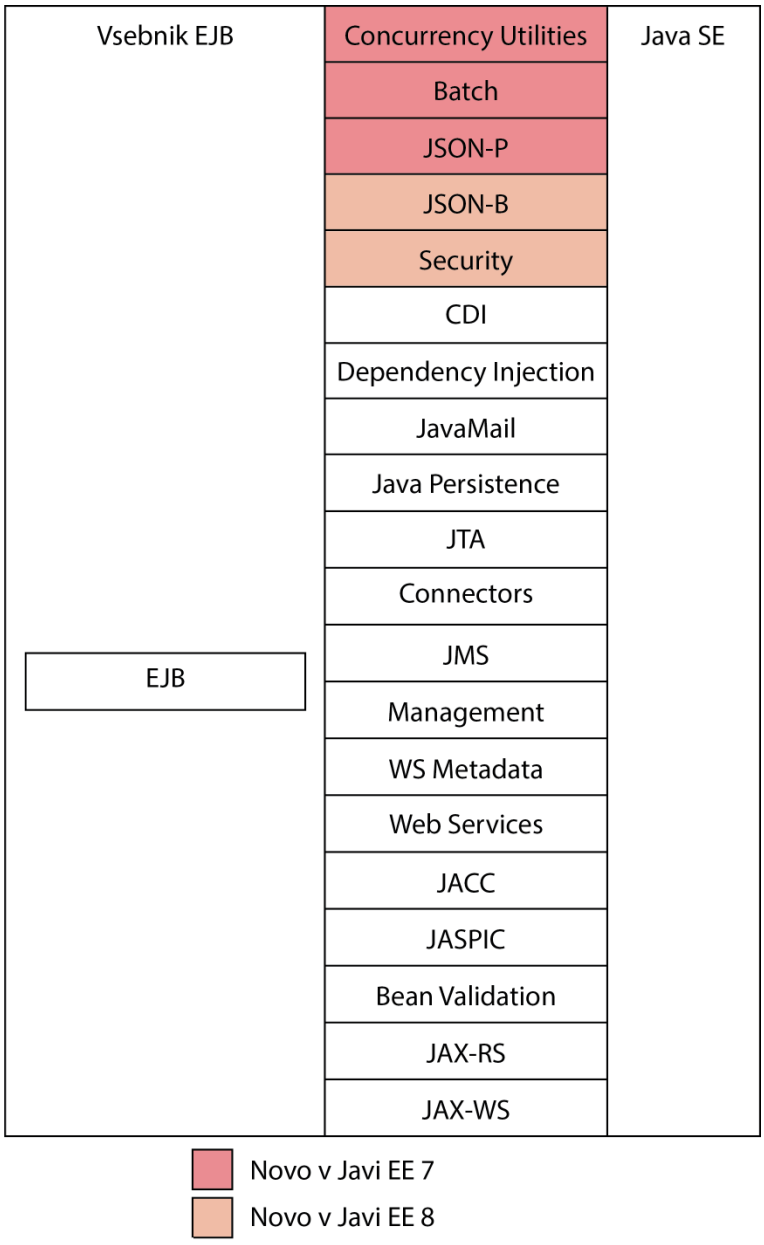


Slika 3.4: API-ji Java EE v spletnem vsebniku [5]

3.2.2 Vsebnik EJB

Vsebnik EJB *(EJB Container)* je vmesnik med poslovnimi zrni *(enterprise beans)* in strežnikom Jave EE. Poslovna zrna zagotavljajo poslovno logiko aplikacije. Vsebnik EJB upravlja izvajanje poslovnih zrn aplikacije Jave EE. Kot je že omenjeno, se EJB in njihov vsebnik izvajajo na strežniku Jave EE [1].

Slika 3.5 prikazuje tehnologije v vsebniku EJB. Z rozasto barvo so obarvana tiste tehnologije, ki so na novo dodane v Javi EE 7, z bež barvo pa tiste, ki so na novo dodane v Javi EE 8.



Slika 3.5: API-ji Java EE v vsebniku EJB [5]

3.3 Podatkovni nivo

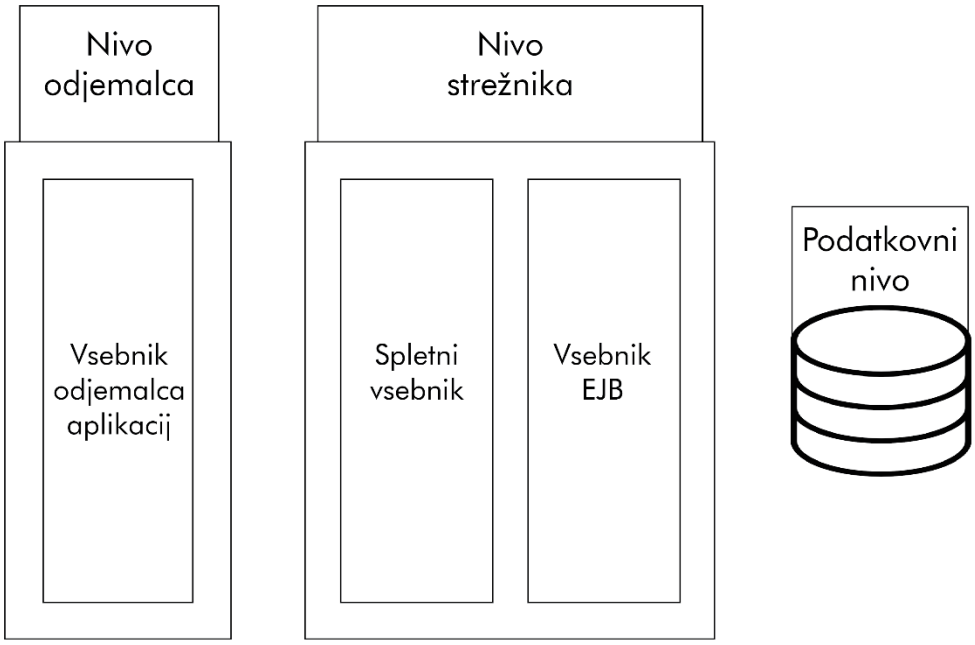
Na podatkovnem nivoju se nahaja podatkovna baza. Običajno se ne nahajajo na enakem stroju kot strežnik Jave EE. Storitve na tem nivoju so zaščitene pred neposrednim dostopom komponent odjemalca. Interakcija med komponentami odjemalca in podatkovne baze mora potekati preko preko procesov nivoja poslovne logike [5].

V naslednjem poglavju bomo predstavili nove in posodobljene specifikacije Jave EE 7 in 8 ter za vsako označili, na katerem nivoju več-nivojskih aplikacij se uporablja.

4 Nove in posodobljene tehnologije v Java Platform, Enterprise Edition

V tem poglavju bodo podrobneje opisane vse nove in posodobljene tehnologije Jave EE 7 in 8.

V vsakem podpoglavju bo najprej opisano na katerih nivojih in vsebnikih več-nivojske aplikacije je tehnologija zahtevana. Za lažjo predstavo bo podana slika 4.1, na kateri bodo z bledo rdečo barvo obarvani vsebniki, v katerih je tehnologija zahtevana.

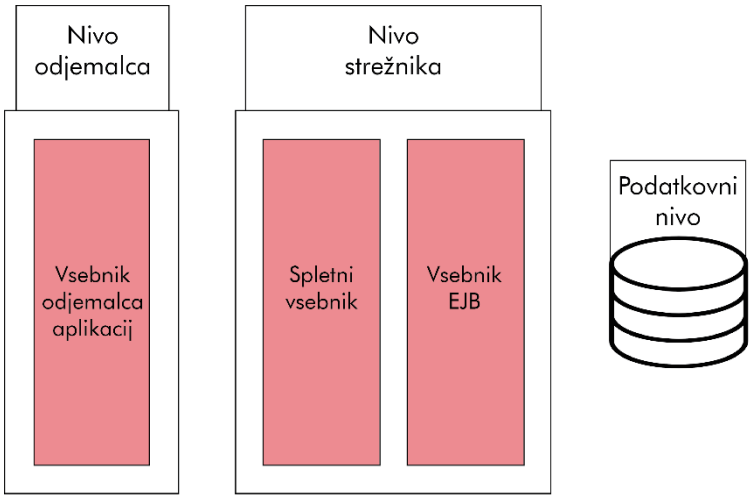


Slika 4.1: Prikaz več-nivojskega aplikacijskega modela Jave EE

Nato bodo podane vse informacije in opisani vsi pojmi, ki so potrebni za razumevanje tehnologije. Potem pri na novo dodanih tehnologijah sledi njihova razlaga. Na koncu vsakega podpoglavja, ki opisuje posodobljeno tehnologijo, sledijo še naštete in na kratko opisane novosti te tehnologije v Javi EE 7 in/ali 8.

4.1 Java API for JSON Processing 1.0 in 1.1

Java API for JSON Processing (JSON-P) 1.0 je nova tehnologija v Javi EE 7. Vsebuje specifikacijo programskega vmesnika JSON-P verzije 1.0, Java EE 8 pa JSON-P verzije 1.1. Ta tehnologija je zahtevana na nivoju odjemalca in na nivoju strežnika. Na nivoju odjemalca je zahtevana v vsebniku odjemalca aplikacij. Na nivoju strežnika pa v spletnem vsebniku in v vsebniku EJB. Pravkar opisano je prikazano na sliki 4.2. Z rozasto barvo so na sliki obarvani vsebniki, ki zahtevajo JSON-P.



Slika 4.2: Prikaz vsebnika odjemalskih aplikacij ter spletenega vsebnika in vsebnika EJB

JSON je besedilna oblika izmenjave podatkov, ki izhaja iz JavaScripta. Slednji se uporablja v spletnih storitvah in drugih povezanih aplikacijah [5]. JSON format je postal zelo popularen, posledično pa se je pojavila potreba po standardizaciji API-ja za manipulacijo JSON formata. JSON-P predstavlja prvi del standardnega vmesnika [19].

JSON-P omogoča Java EE aplikacijam razčlenjevanje, spreminjanje in poizvedovanje po podatkih JSON. To lahko naredijo z uporabo objektnega *(Object)* ali pretočnega *(Streaming)* modela [5]. Prvi, JSON Object Model, je visokonivojski API, ki ponuja nespremenljive modele objektov za strukture JSON objektov in JSON matrik. Drugi, JSON Streaming Model, je nizkonivojski API, ki je zasnovan za učinkovito obdelavo velikih količin podatkov JSON [19].

Streaming API ponuja način za razčlenjevanje in generiranje JSON-a v pretočni obliki [19].

Object Model API ustvari naključno dostopno drevesno strukturo, ki predstavlja podatke JSON v pomnilniku [19].

Če povzamemo zgoraj napisano lahko rečemo, da je JSON-P nova specifikacija, ki za platformo prilagaja branje in pisanje JSON vsebine [16].

V JSON-P 1.1 so vključene naslednje novosti:

- JSON Pointer, ki definira sintakso nizov za sklicevanje na določeno vrednost znotraj dokumenta JSON. Kazalec *(Pointer)* vključuje tudi API-je za pridobivanje vrednost iz ciljnega dokumenta in preoblikovanje teh tako, da se ustvari nov dokument JSON:
- JSON Patch, ki določa obliko za izražanje zaporedja operacij, ki se porabljajo uporabljajo za dokument JSON.
- JSON Merge Patch, ki določa pravila o formatu in procesiranju za uporabo operacij na dokumentu JSON. Te temeljijo na določeni vsebini ciljnega dokumenta.
- V osnovno obdelavo dokumentov JSON sta dodani funkciji urejanja in preoblikovanja.
- Dodani razredi in metode pomoči, ki se imenujejo JSON Collectors. Te uporabljajo funkcije pretočnega API-ja *(Stream API)*, ki je bil dodan v Javi SE 8 [20].

4.2 Java Persistence API 2.1 in 2.2

Platforma Java EE 7 vsebuje specifikacijo programskega vmesnika Java Persistence API (JPA) verzije 2.1, Java EE 8 pa JPA verzije 2.2. Ta tehnologija je zahtevana na nivoju odjemalca in na nivoju strežnika. Na nivoju odjemalca je zahtevana v vsebniku odjemalca aplikacij. Na nivoju strežnika pa v spletnem vsebniku in v vsebniku EJB. Pravkar opisano je prikazano na sliki 4.2. Z rozasto barvo so na sliki obarvani vsebniki, ki zahtevajo JPA. Torej je zahtevana na istih nivojih in vsebnikih kot JSON-P.

JPA je rešitev za trajnost podatkov, ki temelji na standardih Java. Uporablja pristop objektno-relacijskega preslikovanja. Slednji je uporabljen zato, da se na ta način premosti razkorak med objektno orientiranim modelom in relacijsko podatkovno bazo. JPA se uporablja tudi v aplikacijah Java SE izven okolja Jave EE [5].

Java Persistence oziroma trajnost sestavljajo naslednja tri področja:

- API za izvajanje osnovnih CRUD operacij. Omogoča torej: shranjevanje, posodabljanje, pridobivanje in odstranjevanje objektov iz podatkovne baze
- Povpraševalni jezik za pridobivanje objektov iz podatkovne baze na ustrezen način (optimizacijski), brez potrebe po pisanju SQL stavkov.
- Deklarativni način za izvedbo preslikave med objekti in relacijskimi podatkovnimi bazami [21].

V JPA 2.1 so vključene naslednje novosti:

- Z uporabo lastnosti *java.persistence.schema-generation.**, so lahko generirane tabele in shema iz podatkovne baze.
- Unsynchronized Persistence Context pomeni, da konteksta obstojnosti ni potrebno vpisati v transakcijo. Takšen kontekst obstojnosti se lahko eksplicitno vključi v transakcijo.
- Preko *Criteria API* sta podprta posodobitev brisanje v velikem obsegu.
- Z uporabo *FUNCTION* so lahko uporabljene vnaprej določene in uporabniško določene funkcije.
- Sedaj se lahko z uporabo *StoredProcedureQuery* in *@NamedProcedureQuery* prikličejo shranjeni postopki [19].

V JPA 2.2 pa so vključene naslednje novosti:

- Dodana podpora za Java SE 8 API datuma in časa *(Date and Time API)*.
- Dodana *@Repeatable* anotacija, ki omogoča uporabo iste anoatcije za razred ali atribut, brez uporabe vsebnika anotacij. Kar pomeni, da se lahko razred entitete anotira z večkratnimi anotacijami *@NamedQuery* brez, da bi jih ovili v anotacijo *@NamedQueries*.
- Vmesnika JPA *Query* in *TypedQuery* sta dobila novo metodo *getResultStream*, ki vrne pretok Java SE 8 *(Java SE 8 Stream)* rezultata poizvedbe.
- Dodana podpora za vstavljanje CDI *(CDI Injection)* v *AtributeConverter*-ju.
- Spremenjen mehanizem odkrivanja obstojnosti ponudnika *(Persistence Provider Discovery Mechanism)* za module Jave SE [15].

4.3 Java Message Service API 2.0

Platformi Java EE 7 in 8 vsebujeta specifikacijo programskega vmesnika Java Message Service (JMS) verzije 2.0. Ta tehnologija je zahtevana na nivoju odjemalca in na nivoju strežnika. Na nivoju odjemalca je zahtevana v vsebniku odjemalca aplikacij. Na nivoju strežnika pa v spletnem vsebniku in v vsebniku EJB. Pravkar opisano je prikazano na sliki 4.2. Z rozasto barvo so na sliki obarvani vsebniki, ki zahtevajo JMS. Torej je zahtevana na istih nivojih in vsebnikih kot JSON-P in JPA.

JMS API definira standard za pošiljanje sporočil. Ta omogoča aplikacijskim komponentam Jave EE ustvarjanje, pošiljanje, prejemanje in branje sporočil, kar omogoča porazdeljeno komuniciranje, ki je šibko sklopljeno, zanesljivo in asinhrono [5].

Da bi bil API enostavnejši in lažji za uporabo, je v JMS 2.0 vključeno kar nekaj novosti:

- Nov in poenostavljen API ponuja preprostejšo alternativo prejšnjemu API-ju. Ta API vključuje na primer objekt *JMSContext*, ki združuje funkcije *Connection (povezave)* in *Session (seje)*. [5]
- *Connection*, *Session* in drugi objekti s *close* metodo sedaj implementirajo *java.lang.Autocloseable* vmesnik. Na ta način jih je mogoče uporabiti v Java SE 7 z blokom try-with-resources.
- Dodane so bile nove metode za ustvarjenje seje. Pri njih ni potrebno zagotoviti odvečnih argumentov.
- Objektu tipa *Message* je bila dodana nova metode *getBody*, ki poenostavlja pridobivanje vsebine sporočila, saj ni potrebe po pretvorbi tipa.
- Proizvajalec sporočil lahko sedaj določi, da bo sporočilo dostavljeno z zakasnitvijo *(delivery delay)*.
- Dodane so bile nove metode za pošiljanje. In sicer iz tega razloga, da bi bilo aplikaciji dovoljeno asinhrono pošiljanje sporočil. Asinhrono pošiljanje je na voljo za aplikacije, ki se izvajajo v Javi SE ali na vsebniku odjemalca aplikacije Jave EE. Ni pa na voljo za aplikacije, ki se izvajajo na spletnem vsebniku ali vsebniku EJB Jave EE.
- Več porabnikov je lahko naročenih na isto temo.
- Ponudniki JMS morajo sedaj obvezo nastaviti lastnost *JMSXDeliveryCount* sporočila. *JMSXDeliveryCount* je lastnost, s pomočjo katere izvemo število dostav sporočil [15], [19].

4.4 Bean Validation 1.1 in 2.0

Platforma Java EE 7 vsebuje specifikacijo programskega vmesnika Bean Validation verzije 1.1, Java EE 8 pa Bean Validation 2.0. Ta tehnologija je zahtevana na nivoju odjemalca in na nivoju strežnika. Na nivoju odjemalca je zahtevana v vsebniku odjemalca aplikacij. Na nivoju strežnika pa v spletnem vsebniku in v vsebniku EJB. Pravkar opisano je prikazano na sliki 4.2. Z rozasto barvo so na sliki obarvani vsebniki, ki zahtevajo Bean Validation. Torej je zahtevana na istih nivojih in vsebnikih kot JSON-P, JPA in JMS.

Specifikacija Bean Validation definira model metapodatkov in API za validacijo podatkov v komponentah JavaBeans. Namesto porazdelitve validacije podatkov po več nivojih, kot sta brskalnik in strežniška stran, se lahko validacijo omejitev določi na enem mesu in se jih deli po različnih nivojih [5].

V Bean Validation 1.1 so vključene naslednje novosti:

- Validacije omejitev se lahko uporabijo za parametre, povratne vrednosti poljubnih metod in konstrukte.
- Integracijske točke s CDI so bile povečane in predelane. To omogoča bolj naravno in standardno integracijo v Javi EE in Javi SE. Ob tem pa zajema tudi vstavljanje odvisnosti, upravljanje življenjskega cikla komponent in prestrezanje za validacijo metod.
- Ko poteka kaskadiranje veljavnosti, je mogoče spreminjati ciljno skupino [19].

V Bean Validation 2.0 pa so vključene naslednje novosti:

- Podpora za nove funkcije v Javi SE 8, kot je API za datum in čas.
- Dodane nove vgrajene omejitve BeanValidaion [20].

4.5 JavaMail 1.5

Platformi Java EE 7 in 8 vsebujta specifikacijo programskega vmesnika JavaMail verzije 1.5. Ta tehnologija je zahtevana na nivoju odjemalca in na nivoju strežnika. Na nivoju odjemalca je zahtevana v vsebniku odjemalca aplikacij. Na nivoju strežnika pa v spletnem vsebniku in v vsebniku EJB. Pravkar opisano je prikazano na sliki 4.2. Z rozasto barvo so na sliki obarvani vsebniki, ki zahtevajo Java Mail. Torej je zahtevana na istih nivojih in vsebnikih kot JSON-P, JPA, JMS in Bean Validation.

Aplikacije Java EE uporabljajo API JavaMail za pošiljanje e-poštnih sporočil.

API ima dva dela:

- Vmesnik na ravni aplikacij. Komponente aplikacij ga uporabljajo za pošiljanje sporočil pošte.
- Vmesnik ponudnika storitev.

Platforma Java EE vključuje API JavaMail s ponudnikom storitev, ki omogoča komponentam aplikacije pošiljanje internetne pošte [5].

V JavaMail 1.5 so vključene naslednje novosti:

- *@MailSessionDefinition* in *@MailSessionDefinitions* določata, da je *MailSession* registriran z JNDI [19].

4.6 Context and Dependency Injection for Java 1.1 in 2.0

Platforma Java EE 7 vsebuje specifikacijo programskega vmesnika Context and Dependency Injection (CDI) verzije 1.1, Java EE 8 pa CDI 2.0. Ta tehnologija je zahtevana na nivoju odjemalca in na nivoju strežnika. Na nivoju odjemalca je zahtevana v vsebniku odjemalca aplikacij. Na nivoju strežnika pa v spletnem vsebniku in v vsebniku EJB. Pravkar opisano je prikazano na sliki 4.2. Z rozasto barvo so na sliki obarvani vsebniki, ki zahtevajo CDI. Torej je zahtevana na istih nivojih in vsebnikih kot JSON-P, JPA, JMS, Bean Validation in JavaMail.

CDI za Javo EE definira množico kontekstualnih storitev, ki jih ponujajo vsebniki Jave EE. Te storitve omogoča razvijalcem, da v spletnih aplikacijah zlahka uporabljajo strežniška zrna skupaj s tehnologijo JSF. Ker je CDI konstruiran za uporabo z objekti, ki imajo stanje, ima CDI tudi širšo uporabo, kar omogoča razvijalcem veliko prilagodljivost pri integraciji različnih vrst komponent na šibko sklopljen a varen način [5].

V CDI 1.1 so vključene naslednje novosti:

- Dovoljuje samodejno omogočanje CDI za zrna. To stori z anotiranjem dosega *(scope annotation)* in EJB-ji v Javi EE.
- Podpira globalno naročanje in omogočanje prestreznikov, dekoraterjev in alternativ z uporabo anotacije *@Priority*.
- Dodaja anotacijo *@Vetoed*, ki dovoljuje enostavno programersko onemogočanje razredov [19].

V CDI 2.0 pa so vključene naslednje novosti:

- API za zaganjanje vsebnika CDI v Javi SE 8
- Podpora za določanje vrstnega reda opazovalcev *(observer ordering)*, ki določa vrstni red v katerem se za določen dogodek sklicujejo metode opazovalca in asihrono podporo dogodkov za proženje.
- Vmesniki konfiguratorja *(Configurators interfaces)*, ki se uporabljajo za dinamično definiranje in spreminjanje objektov CDI.
- V Javi je literar *(literal)* zaporedje znakov, ki predstavljajo konstantno vrednost in jo je potrebno shraniti v spremenljivko. Tako je v izrazu *String niz = "beseda"* literar niz znakov *"beseda"*, ki se shrani v spremenljivko tipa *String* z imenom *niz*. V Javi SE 8 so vgrajeni literari anotacij. Ti so priročni za ustvarjanje primerkov anotacij in še veliko več. [20].

4.7 Java API for JSON Binding

Java API for JSON Binding (JSON-B) 1.0 je nova tehnologija v Javi EE 8. Slednja vsebuje specifikacijo programskega vmesnika JSON-B 1.0. Ta tehnologija je zahtevana na nivoju odjemalca in na nivoju strežnika. Na nivoju odjemalca je zahtevana v vsebniku odjemalca aplikacij. Na nivoju strežnika pa v spletnem vsebniku in v vsebniku EJB. Pravkar opisano je prikazano na sliki 4.2. Z rozasto barvo so na sliki obarvani vsebniki, ki zahtevajo JSON-B. Torej je zahtevana na istih nivojih in vsebnikih kot JSON-P, JPA, JMS, Bean Validation, JavaMail in CDI.

JSON-B podira serializacijo in deserializacijo med objekti Java in z RFC 7159 združljivimi objekti JSON. Hkrati pa ohranja skladnost z JAXB 2.0. Zagotavlja tudi privzete preslikave instanc in razredov Java v dokumente JSON, ki ustrezajo sprejetim konvencijam [17].

JSON-B prav tako dovoljuje razvijalcem prilagoditev serializacije in deserializacije. Prilagoditev teh dveh procesov za posamezne razrede, je možna z uporabo anotacij (prvi pristop). Za razvoj prilagojenih strategij pa se lahko uporabi graditelj konfiguracije izvajalnega časa *(runtime configuration builder)* (drugi pristop). Drugi pristop vključuje uporabo adapterjev za podporo prilagoditev, ki temeljijo na uporabniku. Ob vsem zgoraj naštetem pa se JSON-B tudi dobro integrira z JSON-P 1.1 [17].

Dva vmesnika omogočata dostop do novega JSON-B:

- *JsonBinding* omogoča JSON-B-ju dostopno točko odjemalca. Sledenje stori z izgradnjo instanc *Jsonb*, ki temeljijo na nastavljenih konfiguracijah in parametrih.
- *Jsonb* zagotavlja postopke serializacije in deserializacije preko metod *toJson()* in *fromJson()* [17].

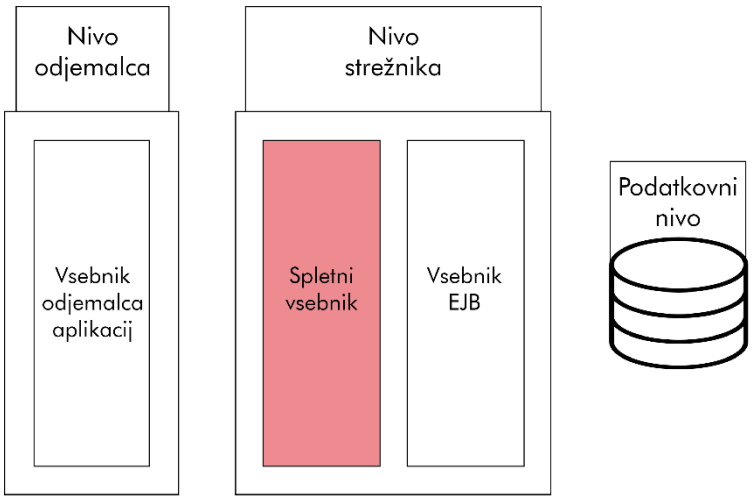
JSON-B opisuje tudi funkcionalnost za priključitev zunanjih ponudnikov JSON-B. Kar pomeni, da razvijalec ni omejen na logično vezavo, ki je priložena API-ju [17].

Sedaj, ko smo predelali JSON-P in JSON-B si lahko podrobneje ogledamo, kako se razlikujeta. V poglavju 4.1 smo povedali, da JSON-P ponuja prenosno standardno rešitev za razčlenjevanje, generiranje, spreminjanje in poizvedovanje po podatkih JSON, kar je razvijalcem v veliko pomoč. A se je bilo potrebno pri pretvorbi objektov Java iz/v predstavitve JSON še vedno zanašati na ogrodja *(frameworks)*, kot so EclipseLink MOXy in Google GSON. Ta ogrodja pa so si med seboj različna, saj ni nobenega standarda. Posledično je bila pri razvijalcih Jave EE (pred nastankom JSON-B-ja) to najbolj zaželena novost. Zato je Oracle v Javi EE 8 rešil ta problem z JSON-B, ki definira standardno vezno plast za pretvorbo objektov Java v sporočila

JSON in pretvorbo sporočil JSON v objekte Java. Tako JSON-B izkorišča in dopolnjuje JSON-P ter izboljšuje enostavnost razvoja [22].

4.8 Java API for WebSocket 1.0 in 1.1

WebSocket 1.0 je nova tehnologija v Javi EE 7. Slednja vsebuje specifikacijo programskega vmesnika WebSocket verzije 1.0, Java EE 8 pa WebSocket 1.1. Ta tehnologija je zahtevana na nivoju strežnika v spletnem vsebniku. Pravkar opisano je prikazano na sliki 4.3. Z rozasto barvo je obarvan vsebnik, ki zahteva WebSocket.



Slika 4.3: Prikaz spletnega vsebnika

V tradicionalnem modelu zahtevek-odziv, ki se uporablja v HTTP, odjemalec zahteva vire, strežnik pa vrne odgovore. Odjemalec je vedno sprožilec za zamenjavo, kar pomeni, da strežnik ne more poslati nobenih podatkov brez, da za njih najprej zaprosi odjemalec. Ta model je deloval dobro za svetovni splet *(World Wide Web)*. Slednje je bilo mogoče zato, ker so odjemalci občasno zahtevali dokumente, ki so se redko spreminjali. Omejitve tega pristopa pa so vedno bolj pomembne, ker se vsebina hitro spreminja, uporabniki pa pričakujejo vedno bolj interaktivno vsebino na spletu. Protokol WebSocket obravnava prej naštete omejitve na način, da zagotavlja dvosmerni komunikacijski kanal med odjemalcem in strežnikom preko protokola TCP. To pomeni, da lahko oba istočasno sprejemata in oddajata podatke. TCP je protokol, ki je ustvarjen za pošiljanje podatkovnih paketov preko interneta. TCP je eden izmed najbolj uporabljanih protokolov. V kombinaciji z drugimi odjemalskimi tehnologijam, kot sta JavaScript in HTML5, pa WebSocket omogoča spletnim aplikacijam, da zagotovijo bogatejšo uporabniško izkušnjo [5].

Java API for WebSocket je nova specifikacija v platformi Java EE 7. Razvijalci jo uporabijo, ko želijo integrirati WebSocket v svoje aplikacije. Java API for WebSocket krajše imenujemo kar WebSocket [5].

Java API for WebSocket:

- Omogoča, da se končni točki odjemalca in strežnika določita deklarativno preko anotacij na POJO ali programersko preko implementacije vmesnika [19]. Na ta način se določijo konfiguracijski parametri končnih točk in njihove metode povratnega klica življenjskega cikla [5].
- Zagotavlja konfiguracijo specifično za strežnik, kot so:
    - preslikava, ki identificira končno točko WebSocket-a v prostoru URI vsebnika,
    - sub-protokoli, ki jih podpira končna točka in
    - razširitve, ki jih zahteva aplikacija.
- Ponuja konfiguracijo specifično za odjemalce, kot so po meri konfigurirani algoritmi.
- Omogoča pakiranje in namestitve na JDK ali spletne vsebnike.
- Omogoča integracijo z obstoječimi tehnologijami Java EE [19].

V tej verziji WebSocket vsebuje metodo *addMessageHandler*, ki se zanaša na pridobivanje generičnih parametrov med izvajanjem, kar pa ni vedno možno. Ker metoda deluje za nekatere običajne primere, ta napaka ni bila ugotovljena v prvi verziji. Ta pomanjkljivost pa je zelo vidna, ko se lamda izrazi uporabljajo kot MessageHandlers. Iz tega razloga sta v WebSocket 1.1 sta vključeni dve novi metodi:

- *MessageHandler.Whole* in
- *MessageHandler.Partial.*

4.9 Java Servlet 3.1 in 4.0

Platforma Java EE 7 vsebuje specifikacijo programskega vmesnika Java Servlet verzije 3.1, Java EE 8 pa Java Servlet verzije 4.0. Ta tehnologija je zahtevana na nivoju strežnika v spletnem vsebniku. Pravkar opisano je prikazano na sliki 4.3. Z rozasto barvo je obarvan vsebnik, ki zahteva Java Servlet. Torej je zahtevana na istih nivojih in vsebnikih kot WebSocket.

Tehnologija Java Servlet omogoča določitev servletnih razredov specifičnih za HTTP [5]. Servlet je del kode v Javi, ki se naloži v vsebnik in se izvede kot reakcija na nek HTTP ukaz. Torej dinamično obdela zahtevke in oblikuje odzive. Servletni razred razširja zmožnosti strežnikov, ki gostijo aplikacije do katerih dostopajo programski modeli zahtevek-odziv.

Čeprav se lahko servleti odzovejo na katero koli vrsto zahtevka, se pogosto uporabljajo za razširitev aplikacij, ki jih gostijo spletni strežniki [5].

V Java Servlet 3.1 so vključene naslednje novosti:

- Ponuja proces za neblokirajočo zahtevo in neblokirajoč odgovor za asinhrone servlete.
- Določa standardni mehanizem za nadgradnjo obstoječe HTTP povezave z drugim protokolom. To stori z uporabo *HttpUpgradeHandler*.
- Določa pravila, za katere HTTP metode pokrivajo <*security-constraint*> [19].

V Java Servlet 4.0 so vključene naslednje novosti:

- Server Push, ki je zmožnost strežnika, da predvidi kaj bo odjemalec potreboval pred zahtevkom odjemalca.
- HTTP Trailer, ki je zbirka posebnih tipov HTTP glav *(HTTP headers)*, ki je na vrsti po telesu odgovora *(response body)*. Kar pomeni, da pošiljatelju omogoča, da na koncu sporočil vključi dodatna polja. Na ta način dostavi metapodatke, ki se lahko dinamično generirajo med pošiljanjem sporočila. Takšni metapodatki sta na primer: preverjanje celovitosti sporočila in digitalni podpis [20].

4.10 JavaServer Faces 2.2 in 2.3

Platforma Java EE 7 vsebuje specifikacijo programskega vmesnika JavaServer Faces (JSF) verzije 2.2, Java EE 8 pa JSF verzije 2.3. Ta tehnologija je zahtevana na nivoju strežnika v spletnem vsebniku. Pravkar opisano je prikazano na sliki 4.3. Z rozasto barvo je obarvan vsebnik, ki zahteva JSF. Torej je zahtevana na istih nivojih in vsebnikih kot WebSocket in Java Servlets.

Tehnologija JSF je delovno okolje za gradnjo uporabniškega vmesnika spletne aplikacije [5].

V JSF 2.2 so vključene naslednje novosti:

- Faces Flow zagotavlja enkapsulacijo povezanih pogledov/strani z aplikacijsko definiranimi vhodnimi in izhodnimi točkami.
- Resource Library Contracts omogočajo razvijalcem uporabo predlogov obrazcev v celotni aplikaciji. Slednje lahko storijo na večkratni in izmenični način.
- Oznaka omogoča skoraj popolno kontrolo nad uporabniško izkušnjo vsakega posameznega HTML5 elementa v pogledu.

- Stateless Views pomeni, da razvijalcem ni potrebno več shranjevati stanja *UIComponent*. To omogoča aplikacijam z JavaScript komponentami, da upravljajo stanja, namesto da to za njih dela JSF [19].

V JSF 2.3 pa so vključene naslednje novosti:

- Neposredna podpora za WebSocket, preko nove oznake: *<f:websocket>*
- Validacija zrn *(bean validation)* na ravni razreda preko nove oznake: *<f:validateWholeBean>*
- S CDI združljiva anotacija *@ManagedProperty* [20].

4.11 Expression Language 3.0

Platformi Java EE 7 in 8 vsebujeta specifikacijo programskega vmesnika Expression Language (EL) verzije 3.0. Ta tehnologija je zahtevana na nivoju strežnika v spletnem vsebniku. Pravkar opisano je prikazano na sliki 4.3. Z rozasto barvo je obarvan vsebnik, ki zahteva EL. Torej je zahtevana na istih nivojih in vsebnikih kot WebSocket, Java Servlets in JSF.

EL zagotavlja pomemben mehanizem, ki omogoča komuniciranje predstavitvene plasti (spletnih strani) z aplikacijsko logiko (upravljalnimi zrni). Uporablja ga več tehnologij Java EE, kot so JSF, JSP in CDI [5].

V EL 3.0 so vključene naslednje novosti:

- EL je posebna specifikacija. Z uporabo razreda *ELProcessor* je lahko EL konfiguriran in uporabljen zunaj vsebnika Jave EE.
- V EL je po novem vključena sintaksa lambda. Z uporabo lambda je sedaj podprt celoten nabor postopkov zbiranja, kot je na primer filter.
- Poleg navadnih aritmetičnih in primerjalnih operatorjev so zaradi večje izrazitosti EL dodani novi operatorji, kot je operator za zadolžitev in operator za spajanje nizov [19].

4.12 JavaServer Pages 2.3

Platformi Java EE 7 in 8 vsebujeta specifikacijo programskega vmesnika JavaServer Pages (JSP) verzije 2.3. Ta je zahtevana na nivoju strežnika v spletnem vsebniku, kar je prikazano na sliki 4.3. Z rozasto barvo je obarvan vsebnik, ki zahteva JSP. Torej je zahtevana na istih nivojih in vsebnikih kot WebSocket, Java Servlets, JSF in EL.

JSP omogoča poenostavljen in hiter način ustvarjanja spletnih strani, ki prikazujejo dinamično ustvarjeno vsebino. Specifikacija JSF določa interakcijo med strežnikom in stranjo JSP ter opisuje obliko in sintakso strani [23]. Ob tem dopušča tudi, da se delčki kode servleta neposredno vstavijo v besedilni dokument [5].

Stran JSP je besedilni dokument, ki vsebuje dve vrsti besedila:

- Statične podatke, ki se lahko izrazijo v kakršnem koli besedilnem formatu. Kot sta na primer HTML ali XML.
- Elemente JSP, ki določajo, kako stran ustvari dinamično vsebino [5].

V tehnologij JSP 2.3 so vključene naslednje, ki so v primerjavi z JSP 2.2 manjše novosti:

- Podpora za EL 3.0.
- JSP-ju je razpoložljiv API Servlet 3.1 [24].

Spletne komponente (servelti, spletne strani) se lahko ustvarijo s pravkar opisano tehnologijo JSP (strani JSP) in/ali tehnologijo JSF. Strani JSP so besedilni dokumenti, ki se izvedejo kot servleti, a omogočajo naravnejši pristop ustvarjanja statične vsebine Tehnologija JSF temelji na servletih in tehnologiji JSP ter kot je že omenjeno v poglavju 4.10 zagotavlja ogrodje *(framework)* za uporabniški vmesnik spletnih strani. Zato lahko sedaj, ko smo predelali obe tehnologiji pojasnimo razliko med njima [5].

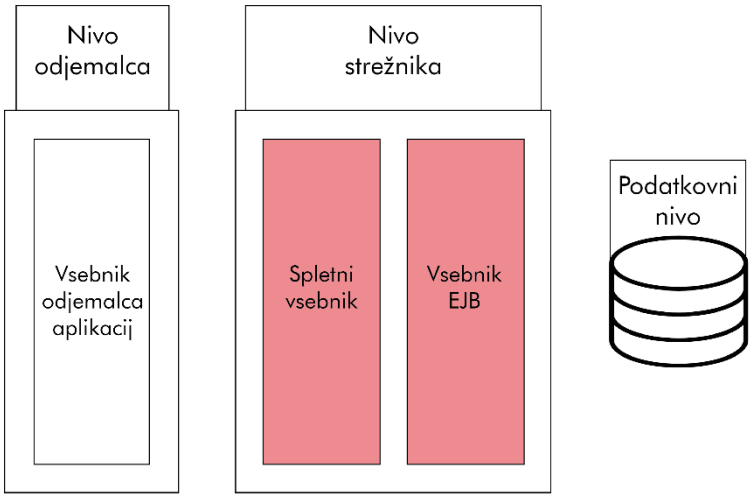
JSP je bil primarno razvit za ustvarjanje dinamičnih spletnih strani za majhne aplikacije. Zato ga je težko uporabljati za obsežne aplikacije, saj so te razvite z določenim okoljem in sistemom, ki temelji na komponentah. JSF pa je ravno sistem, ki temelji na komponentah in je zelo koristen za obsežne projekte. JSF prav tako uporablja MVC ogrodje *(Model-View-Controller)* in so iz tega razloga uporabniški vmesnik in njegove komponente lahko ponovno uporabljeni na določeni spletni strani. Medtem, ko je JSP precej starejši in zato popolnoma razvit, je JSF še vedno v fazi razvijanja. Slednje pomeni, da so z njegovo vsako različico dodane novosti [25].

Zato je varno reči, da je JSP preteklost in JSF prihodnost. A je vseeno zelo priporočljivo, da se razvijalec pred uporabo JSF nauči JSP. Razlog za to je ta, da se s pomočjo JSP lahko razumejo celotni procesi ozadja, kar posledično razvijalcu omogoča razvoj naprednih aplikacij [25].

4.13 Concurrency Utilities for Java EE

Concurrency Utilities 1.0 je nova tehnologija v platformi Java EE 7. Slednja platforma in platforma Java EE 8 vsebujeta specifikacijo programskega vmesnika Concurrency Utilities

verzije 1.0. Ta tehnologija je zahtevana na nivoju strežnika v spletnem vsebniku in vsebniku EJB. Pravkar opisano je prikazano na sliki 4.4. Z rozasto barvo sta obarvana vsebnika, ki zahteva Concurrency Utilities.



Slika 4.4: Prikaz spletnega vsebnika in vsebnika EJB

Vzporednost oziroma hkratnost *(Concurrency)* je koncept izvajanja dveh ali več nalog hkrati. Te naloge lahko vsebujejo metode/funkcije, dele programa ali celo druge programe. V trenutni računalniški arhitekturi je podpora več jeder in več procesorjev v eni CPE zelo pogosta [5].

Platforma Java je vedno ponujala podporo za sočasno programiranje. To je bilo osnovno za izvajanje številnih storitev, ki jih ponujajo vsebniki Jave EE. V Javi SE 5 je bil dodaten visokonivojski API za vzporednost, zagotovljen s paketom *java.util.concurrent* [5].

Pred Javo EE 7 pa ni bilo nobenih specifičnih API-jev, ki bi omogočali razvijalcem v podjetjih uporabo pripomočkov za vzporednost na varen standardni način. Spletni vsebniki in vsebniki EJB Jave EE instancirajo objekte z uporabo bazenov niti. Te upravljajo vsebniki. Iz tega razloga je bila uporaba Java SE API-jev za vzporednost za navajanje objektov virov močno odsvetovana. Razlog za to je, da če razvijalec ustvari nov (ne-upravljan) *Thread* objekt, vsebnik ne more garantirati, da bodo bile druge storitve platforme Java EE (na primer transakcije in varnost) del tega *Thread* objekta [5].

Novo nastal API Concurrency Utilities rešuje zgoraj opisan dolgo obstajajoč problem poslovne Jave. Ta se glasi: kako ustvariti Java nitne procese brez znanja in kontrole aplikacijskega strežnika. Concurrency Utilities izboljšuje produktivnost razvijalcev z upravljanim nitnim bazenom in izvajalskimi viri [16].

Concurrency Utilities:

- Zagotavlja zmožnost vzporednosti komponentam Java EE aplikacij, ne da bi ogrozili celovitost vsebnika.
- Definira upravljane objekte: *ManagedExecutorService*, *ManagedScheduledExecutorService*, *ContextService* in *ManagedThreadFactory* [19].

Zgornji alineji lahko z besedami napišemo tudi: Concurrency Utilities je standardni API, za zagotavljanje asinhronih zmožnosti aplikacijskih komponent Jave EE. Slednje stori s pomočjo naslednjih vrst objektov: managed executor service, managed scheduled executor services, managed thread factory in context service [5].

4.14 Batch Processing API for the Java Platform

Batch Processing 1.0 je nova tehnologija v platformi Java EE 7. Slednja platforma in platforma Java EE 8 vsebujeta specifikacijo programskega vmesnika Batch Processing verzije 1.0. Ta tehnologija je zahtevana na nivoju strežnika v spletnem vsebniku in vsebniku EJB. Pravkar opisano je prikazano na sliki 4.4. Z rozasto barvo sta obarvana vsebnika, ki zahteva Batch Processing. Torej je zahtevana na istih nivojih in vsebnikih kot Concurrency Utilities.

Nekatere poslovne aplikacije vsebujejo naloge, ki jih je mogoče izvajati brez interakcije uporabnika. Takšne naloge imenujemo serijska *(batch)* dela in se običajno izvajajo periodično ali ko je poraba virov nizka. Serijska dela pogosto obdelujejo velike količine informacij, kot so dnevniške datoteke, zapisi iz podatkovne baze ali slike. Ti primeri vključujejo generiranje poročil, pretvorbo oblike podatkov in obdelavo slik [5].

Torej se serijska obdelava *(Batch Processing)*, nanaša na tekoča serijska opravila v računalniškem sistemu, ki jih je mogoče izvesti brez interakcije uporabnikov [5].

Java EE vsebuje delovno okolje, ki zagotavlja infrastrukturo za serijsko izvajanje, ki je skupna vsem serijskim aplikacijam. Slednje omogoča razvijalcem, da se osredotočijo na poslovno logiko njihove serijske aplikacije [5].

Serijsko delovno okolje je sestavljeno iz:

- za delo specificiranega jezika, ki temelji na XML,
- množice serijskih anotacij in vmesnikov za aplikacijske razrede, ki izvajajo poslovno logiko,
- serijskih vsebnikov, ki upravljajo izvajanje serijskih del in

- podpornih razredov ter vmesnikov za interakcijo s serijskim vsebnikom [5].

Serijska obdelava:

- Za opis serijskih del, omogoča uporabo za delo specificiranega jezika. Specificiran jezik določa shema XML. Ta definira celotno zaporedje izvajanja nalog.
- Predstavlja modul serijskega programiranja z uporabo vmesnikov, abstraktnih razredov in anotacij na polju.
- Ponuja stile za obdelavo serijskih in paketnih del [19].

4.15 Java EE Connector Architecture 1.7

Platformi Java EE 7 in 8 vsebujeta specifikacijo programskega vmesnika Java EE Connector Architecture (JCA) verzije 1.7. Ta tehnologija je zahtevana na nivoju strežnika v spletnem vsebniku in vsebniku EJB. Pravkar opisano je prikazano na sliki 4.4. Z rozasto barvo sta obarvana vsebnika, ki zahteva JCA. Torej je zahtevana na istem nivoju in vsebniku kot Concurrency Utilities in Batch Processing.

JCA uporabljajo proizvajalci orodij in sistemski integratorji za ustvarjanje adapterjev virov. Ti podpirajo dostop do poslovnih informacijskih sistemov, ki so lahko priključeni v kateri koli izdelek Jave EE. Adapter vira je programska komponenta, ki omogoča komponentam aplikacije Jave EE dostop in interakcijo z globje ležečim vodjo virov EIS-a. Ker je adapter virov specifičen za svojega vodjo virov, za vsak tip podatkovne baze ali poslovni informacijski sistem običajno obstaja drug adapter vira [5].

JCA prav tako zagotavlja na zmogljivost orientirano, varno, razširljivo in na sporočilih temeljno transakcijsko integracijo spletne storitve. Ta spletna storitev temelji na platformi Jave EE in ima obstoječe EIS-e, ki so lahko sinhroni ali asinhroni. Obstoječe aplikacije in EIS-i, ki so v platformo Java EE integrirani preko JCA, so lahko z uporabo JAX-WS in modelov komponent Jave EE, izpostavljeni kot spletne storitve na osnovi XML-a. Tako sta JAX-WS in JCA dopolnilni oziroma komplementarni tehnologiji za integracijo poslovnih aplikacij *(EAI)* in za popolno *(end-to-end)* poslovno integracijo [5].

V JCA 1.7 so vključene naslednje novosti:

- Omogoča uporabo anotacij *@AdministeredObjectDefinition*, *@AdministeredObjectDefinitions*, *@ConnectorFactoryDefinition* in *@ConnectorFactoryDefinitions*. Na ta način določi objekt in tovarno, ki ju upravlja priključek za registracijo v JNDI [19].

4.16 Java Authorization Contract for Containers 1.5

Platformi Java EE 7 in 8 vsebujeta specifikacijo programskega vmesnika Java Authorization Contract for Containers (JACC) verzije 1.5. Ta tehnologija je zahtevana na nivoju strežnika v spletnem vsebniku in vsebniku EJB. Pravkar opisano je prikazano na sliki 4.4. Z rozasto barvo sta obarvana vsebnika, ki zahteva JACC. Torej je zahtevana na istem nivoju in vsebniku kot Concurrency Utilities, Batch Processing in JCA.

JACC specifikacija opredeljuje pogodbo med aplikacijskim strežnikom Jave EE in ponudnikom avtorizacijske politike. Vsi Java EE vsebniki podpirajo to pogodbo.

Specifikacija JACC opredeljuje razrede *java.security.Permission*, ki zadovoljujejo model avtorizacije Jave EE. Ob tem določa tudi vezavo odločitev dostopa vsebnika do operacij na primerih teh razredov dovoljenj. Specifikacija opredeljuje tudi semantike ponudnikov politik, ki za obravnavo avtorizacijskih zahtev platforme Java EE uporabljajo nove razrede dovoljenj, vključno z opredelitvijo in uporabo vlog [5].

4.17 Java Authentication Service Provider Interface for Containers 1.1

Platformi Java EE 7 in 8 vsebujeta specifikacijo programskega vmesnika Java Authentication Service Provider Interface for Containers (JASPIC) verzije 1.1. Ta tehnologija je zahtevana na nivoju strežnika v spletnem vsebniku in vsebniku EJB. Pravkar opisano je prikazano na sliki 4.4. Z rozasto barvo sta obarvana vsebnika, ki zahteva JASPIC. Torej je zahtevana na istem nivoju in vsebniku kot Concurrency Utilities, Batch Processing, JCA in JACC.

Specifikacija JASPIC definira vmesnik ponudnika storitev *(SPI)*. Z njim so lahko ponudniki avtentikacije, ki izvajajo mehanizme avtentikacije sporočil, vključeni v vsebnike ali izvajalna okolja za obdelavo odjemalskih ali strežniških sporočil. Ponudniki avtentikacije integrirani preko tega vmesnika, delujejo na omrežnih sporočilih. Ta sporočila jim priskrbijo njihovi klicni vsebniki. Ponudniki avtentikacije preoblikujejo odhodna sporočila tako, da je lahko vir vsakega sporočila avtenticiran od sprejemnega vsebnika in da je lahko prejemnik sporočila avtenticiran od pošiljatelja sporočila. Ponudniki avtentikacije avtenticirajo vsa dohodna sporočila in svojemu klicnemu vsebniku vrnejo identiteto, ki je ugotovljena kot rezultat avtentikacije sporočila [5].

4.18 Java API for RESTful Web Services 2.0 in 2.1

Platforma Java EE 7 vsebuje specifikacijo programskega vmesnika Java API for RESTful Web Services (JAX-RS) verzije 2.0, Java EE 8 pa JAX-RS verzije 2.1. Ta tehnologija je zahtevana na nivoju strežnika v spletnem vsebniku in vsebniku EJB. Pravkar opisano je prikazano na sliki 4.4. Z rozasto barvo sta obarvana vsebnika, ki zahteva JAX-RS. Torej je zahtevana na istem nivoju in vsebniku kot Concurrency Utilities, Batch Processing, JCA, JACC in JASPIC.

Representational State Transfer (REST) je arhitekturni stil, ki določa omejitve, kot je enotni vmesnik *(uniform interface)*. Ta ob uporabi za spletno storitev sproži želene lastnosti, kot so zmogljivost, skalabilnost in prilagodljivost. Prav te lastnosti pa omogočajo storitvam, da delujejo na spletu najbolje. V arhitekturnem stilu REST se podatki in funkcionalnosti štejejo za vire. Do teh virov pa se dostopa z enoličnimi indikatorji vira *(Uniform Resource Identifiers – URIs)*, ki so običajno povezave na spletu. Viri so obravnavani z uporabo množice natančno definiranih operacij. Arhitekturni stil REST omejuje arhitekturo na arhitekturo odjemalec-strežnik in je namenjen uporabi komunikacijskega protokola brez stanja, običajno kar HTTP. V arhitekturnem stilu REST odjemalci in strežniki z uporabo standardiziranega vmesnika in protokola izmenjujejo predstavitev virov. Prav ta načela pa spodbujajo RESTful aplikacije, da so preproste, lahke in imajo visoko zmogljivost. [26].

JAX-RS definira API-je za razvoj spletnih storitev. Te so zgrajene v skladu z REST arhitekturnim stilom. Aplikacija JAX-RS je spletna aplikacija sestavljena iz razredov, ki so zapakirani kot servlet v datoteki WAR skupaj z zahtevanimi knjižnicam [5].

V JAX-RS 2.0 so vključene naslednje novosti:

- Ponuja nov API odjemalca *(Client API)*, ki se lahko uporablja za dostop do spletnih virov in omogoča integracijo s ponudniki JAX-RS.
- Podpira asinhrono obdelavo v Client API in API-ju strežnika *(Server API)*.
- Določa filtre sporočila *(Message Filters)* in prestreznike entitet *(Entity Interceptors)* kot razširitveni točki za prilagoditev obdelave zahtevka/odziva na strani odjemalca in strežnika.
- Podpira nove pogajalske vsebine na strani strežnika, z uporabo faktorja *qs*.
- Omogoča deklarativno validacijo polj, lastnosti in parametrov. Ti so injicirani z uporabo *@HeaderParam*, *@QueryParam* in drugimi. Razrede virov je mogoče označiti z anotacijami omejitev [19].

V JAX-RS 2.1 pa so vključene naslednje novosti:

- Reactive Client API. Sedaj izboljšani API-ji v končni fazi v Javi SE omogočajo, da je ob prejemu rezultata klica na ciljnem viru, zaporedje teh rezultatov specificirano, razvrščeno po prioriteti, združeno ali spojeno in določeno, kako je mogoče obravnavati izjeme.
- Izboljšave v podpori za dogodke, ki so poslani s strežnika.
- Odjemalci se lahko z uporabo dolgotrajne povezave naročijo na obvestila o dogodkih, ki jih izda strežnik.
- Podpora za objekte JSON-B in izboljšana integracija s tehnologijami CDI; Servlet in Bean Validation [20].

4.19 Java Transaction API 1.2

Platformi Java EE 7 in 8 vsebujeta specifikacijo programskega vmesnika Java Transaction API (JTA) verzije 1.2. Ta tehnologija je zahtevana na nivoju strežnika v spletnem vsebniku in vsebniku EJB. Pravkar opisano je prikazano na sliki 4.4. Z rozasto barvo sta obarvana vsebnika, ki zahteva JTA. Torej je zahtevana na istem nivoju in vsebniku kot Concurrency Utilities, Batch Processing, JCA, JACC, JASPIC in JAX-RS.

JTA zagotavlja standardni vmesnik za določanje transakcij. Arhitektura Java EE omogoča privzeto samodejno potrjevanje *(auto commit)*, ki obravnava potrditve transakcije in povrnitve prejšnjega stanja transakcije. *Auto commit* pomeni, da katere koli druge aplikacije, ki si ogledujejo podatke, vidijo posodobljen podatke po vsaki operaciji branja ali pisanja v podatkovno bazo. V primeru, ko aplikacija izvede dve ločeni operaciji dostopa do podatkovne baze, pa je potrebna uporaba JTA API-ja. Slednji API moramo uporabiti za razmejitev, kjer se celotna transakcija skupaj z operacijama začne, povrne v prejšnje stanje in namesti [5].

V JTA 1.2 so vključene naslednji novosti:

- *@Transactional* zagotavlja aplikaciji deklarativno omejitev transakcij na CDI upravljanih zrnih. Kot tudi na razredih, ki so definirani kot upravljalna zrna po specifikaciji Java EE. Kar lahko stori tako na nivoju razreda, kot tudi metode. Anotacije na nivoju metod prevladajo tiste na nivoju razreda.
- *@TransactionScoped* je nov CDI obseg, ki definira primerke zrna, katerih življenjski cikel obsega trenutno aktivno transakcijo JTA [19].

4.20 Java EE Security API

Java Security 1.0 je nova tehnologija v Javi EE 8. Ta tehnologija je zahtevana na nivoju strežnika v spletnem vsebniku in vsebniku EJB. Pravkar opisano je prikazano na sliki 4.4. Z rozasto barvo sta obarvana vsebnika, ki zahteva Java Security. Torej je zahtevana na istem nivoju in vsebniku kot Concurrency Utilities, Batch Processing, JCA, JACC, JASPIC, JAX-RS in JTA

Novi Java EE Security API je bil uveden iz razloga, da bi bile odpravljene neskladnosti v zvezi z izvajanem skrbi za varnost preko servletnih vsebnikov. Ta težava je bila posebej opazna v spletnem profilu Jave EE. Opazna je bila predvsem zato, ker Java EE določa samo, kako se morajo izvajati standardni API-ji polnega profila Jave EE [17].

Nova specifikacija uvaja tudi sodobne zmogljivosti, kot je podpora za CDI, ki pa na obstoječe API-je ne vplivajo [17].

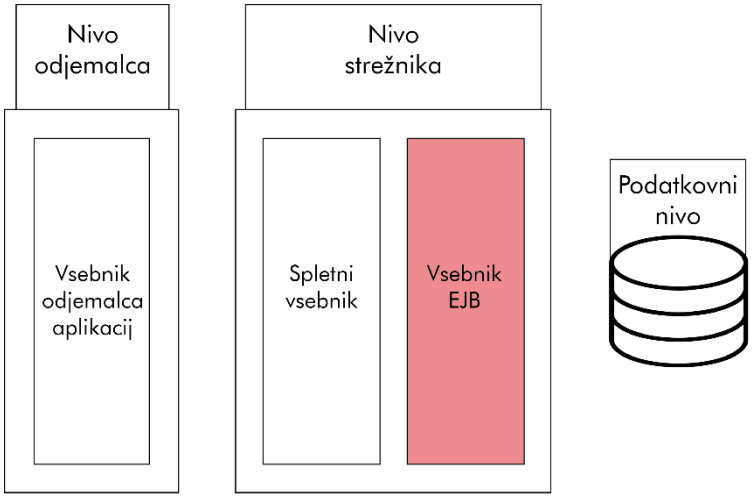
Lepota tega API-ja je, da ponuja alternativen način konfiguracije trgovin identitete *(identity stores)* in mehanizmov za potrditev verodostojnosti oziroma avtentikcije. Pri tem pa ne nadomešča obstoječih varnostnih mehanizmov [17].

Specifikacija Java EE Security API-ja obravnava tri glavne zadeve:

- *HttpAuthenticationMechanism* podpira potrditev verodostojnosti za servletni vsebnik.
- *IdentityStore* standardizira JAAS *LoginModule*.
- *SecurityContext* zagotavlja dostopno točko za programsko varnost [17].

4.21 Enterprise JavaBeans 3.2

Platformi Java EE 7 in 8 vsebujeta specifikacijo programskih vmesnikov Enterprise JavaBeans (EJB) verzije 3.2 in Interceptors verzije 1.2. Specifikacija slednjih je del specifikacije EJB in bo podrobneje opisana v naslednjem podpoglavju [5]. Ta tehnologija je zahtevana na nivoju strežnika v vsebniku EJB. Pravkar opisano je prikazano na sliki 4.5. Z rozasto barvo je obarvan vsebnik, ki zahteva EJB.



Slika 4.5: Prikaz vsebnika EJB

Komponenta EJB ali strežniško zrno je telo kode, ima polja in metode, s katerimi se implementirajo moduli poslovne logike. Na poslovno zrno lahko gledamo, kot na gradbeni blok, ki se lahko uporablja samostojno ali pa z drugim poslovnim zrnom za izvajanje poslovne logike na strežniku Java EE [5].

Java definira naslednje tipe javanskih strežniških zrn:

- Session beans ali sejna zrna, ki opravljajo poslovne operacije, orkestrirajo transakcije ter upravljajo z dostopi [5].
- Message-driven beans ali sporočilna zrna združujejo značilnosti sejnega zrna in poslušalca sporočil. Kar omogoča, da poslovna komponenta sprejema sporočila asinhrono. Običajno so to JMS sporočila [19].

Pomembno je tudi povedati, da celoten nabor funkcij EJB morda ni potreben za vse poslovne aplikacije. Takšne so na primer aplikacije, ki so usmerjene v spletni profil. Zato je bil ustvarjen EJB Lite, ki je minimalna množica celotne EJB funkcionalnosti. V EJB Lite ni bila definirana nobena nova funkcionalnost, sestavljena je le iz ustrezne podmnožice vseh EJB funkcionalnosti [19]. EJB Lite je zahtevana na istem nivoju in vsebniku kot WebSocket, Java Servlet, JSF, EL in JSP. Na sliki 4.3 je z rdečo obarvan vsebnik, ki zahteva EJB Lite.

V EJB 3.2 so vključene naslednje novosti:

- Izbirna je postala podpora za EJB 2.1, EJB QL, končne točke spletne storitve, ki temeljijo na JAX-RPC in pogled odjemalca.

- Izboljšana pogodba sporočilnih zrn z vmesnikom za poslušanje sporočil brez metod. Na ta način se vse javne metode izpostavijo kot metode za poslušanje sporočil. To bo omogočilo prilagojene adapterje vira za prihodnja sporočilna zrna.
- Skupine EJB API-ja so bile določene z jasnimi pravili za EJB Lite vsebnike. In sicer da morajo podpirati druge skupine API-jev. To bo pomagalo določiti, kako se lahko uradno dodajo značilnosti EJB, ki presegajo EJB Lite v izdelke, ki ne podpirajo polnega profil Jave EE.
- V EJB Lite so vključene invokacije asinhronega sejnega zrna in ne dolgotrajne EJB Timer storitve.
- Dodana je možnost, da se lahko onemogoči pasiviranje sejnih zrn s stanjem [19].

4.22 Interceptos 1.2

Platformi Java EE 7 in 8 vsebujeta specifikacijo programskega vmesnika Interceptors verzije 1.2. Specifikacija slednjih je del specifikacij EJB 3.2 in CDI 1.2 [5]. Ta tehnologija je zahtevana na nivoju strežnika v vsebniku EJB. Pravkar opisano je prikazano na sliki 4.5. Z rozasto barvo je obarvan vsebnik, ki zahteva Interceptors. Torej je zahtevana na istem nivoju in vsebniku kot EJB.

Interceptors ali prestrezniki prestrezajo izvajanje poslovnih metod zrna. Uporabljamo jih za opravljanje opravil pred ali po izvajanju poslovne metode zrna. Lahko jih implementiramo v obliki metode ali v obliki razreda [5].

V Interceptors 1.2 so vključene naslednje novosti:

- Povezovanje prestreznikov s pomočjo *InterceptorBinding* je zdaj del te specifikacije namesto CDI-ja.
- Dodan je prestreznik *@AroundConstruct*, ki označuje metodo prestreznika, prejme povratni klic, ko se kliče konstruktor ciljnega razreda. Drugače povedano prestreza kreiranje ciljnega razreda.
- Prestrezniki na ravni metode se lahko razširijo na življenjski cikel povratnih metod. To se stori tako, da se dodajo prestrezniki na ravni konstrukta.
- Prednostni obsegi so lahko namenjeni naročanju prestreznikov, in sicer z uporabo povezovanja prestreznika [19].

V tem poglavju smo predstavili vse nove in posodobljene tehnologije v Javi EE 7 in 8. V naslednjem poglavju bomo opisali, kako se tehnologije Jave EE združujejo v različne profile.

5 Profili v Javi EE

Profili so konfiguracije platforme Java EE, ki so usmerjene v določen razred aplikacij. Dodani so bili že v Javi EE 6 z namenom, da bi z njimi definirali podmnožico tehnologij Jave EE za specifične namene [27].

Java EE 7 definira dva profila. Prvi je poln profil Jave EE 7 *(Full profile)*, ki zajema vse tehnologije Java EE specifikacije. Drugi profil je spletni profil Jave EE 7 *(Web profile)*, ki zajema tehnologije za izdelavo spletnih aplikacij [27].

Obstaja še tretji profil, ki se imenuje MicroProfile, katerega cilj je optimizacija Jave EE za arhitekturo mikrostoritev [28].

Vsak profil je določen v skladu s pravili Java Community Process (JCP). Običajno je predlog za ustvarjanje novega profila ali revizijo obstoječega predložen kot JSR. Pomen JSR-ja je razložen v poglavju 2.2. Ko je enkrat JSR odobren, se oblikuje strokovna skupina, ki bo v prihodnosti izvajala delo, katerega narekuje postopek. V JSR-ju profila mora biti navedena različica platforme Java EE, na kateri se profil gradi. Če se slučajno gradi na že obstoječem profilu, mora biti omenjena tudi ta informacija [27].

Profili se lahko ustvarijo in razvijejo neodvisno od platforme Java EE. A je vseeno predvideno, da bodo profili ohranili razumno raven usklajenosti s platformo. Slednje je predvideno zato, da bi se izognili razdrobljenosti razvojnega prostora v postopoma nezdružljive dele. Iz tega razloga mora biti profil zgrajen na najnovejši različici Java EE platforme, ki je na voljo v času odobritve JSR-ja. Ob tem je priporočljivo tudi to, da strokovne skupine, ki izvajajo delo, presežejo prej omenjeno zahtevo in koliko je praktično, zagotovijo, da se profil gradi na tisti različici Java EE platforme, ki bo najnovejša ob času zaključka profila [27].

Profili določajo nabor storitev, ki jih mora strežnik skladen s profilom ponujati.

Lahko so podmnožica vseh tehnologij Java EE:

- To je mogoče, pod pogojem, da si vsa pravila v specifikaciji, ki se nanašajo na vključeno tehnologijo (bodisi samostojno ali v kombinaciji z drugo tehnologijo), sledijo. Ker lahko profil vsebuje ustrezno podmnožico tehnologij, ki so vsebovane v platformi,

lahko na ta način učinkovito odvrže tehnologije, ki jih platforma podpira, a niso splošno uporabne na določenem področju [27].

Lahko vključujejo tehnologije izven nabora Java EE.

- To pomeni, da lahko profil doda eno ali več tehnologij, ki niso prisotne v platformi [27].

Poleg tega lahko profil označi določeno tehnologijo, kot izbirno. To pomeni, da produkt, ki implementira profil, lahko, ali pa tudi ne, vsebuje tehnologijo, ki je označena kot izbirna. Če to tehnologijo uporabi, mora seveda spoštovati vse relevantne zahtevke, ki so določeni s specifikacijo profila [27].

Produkt lahko vsebuje:

- Dva ali več profilov Jave EE ali
- Full platformo in enega ali več profilov Jave EE

vse dokler njihove združene zahteve niso v konfliktu [27].

Platforma Java SE 7 je potreben temelj za vse profile Jave EE 7. Slednje sklepamo iz tega, ker specifikacija Jave EE 7 zahteva vsaj Javo SE 7.

Naslednje tehnologije morajo biti podprte v vseh profilih Java EE:

- Anotacije vira in življenjskega cikla, ki so definirane s specifikacijo Common Annotations (*Resource*, *Resources*, *PostContruct*, *PreDestroy*) [27].

Naslednje funkcionalnosti morajo biti podprte v vseh profilih Java EE:

- JNDI Naming Context,
- JTA [27].

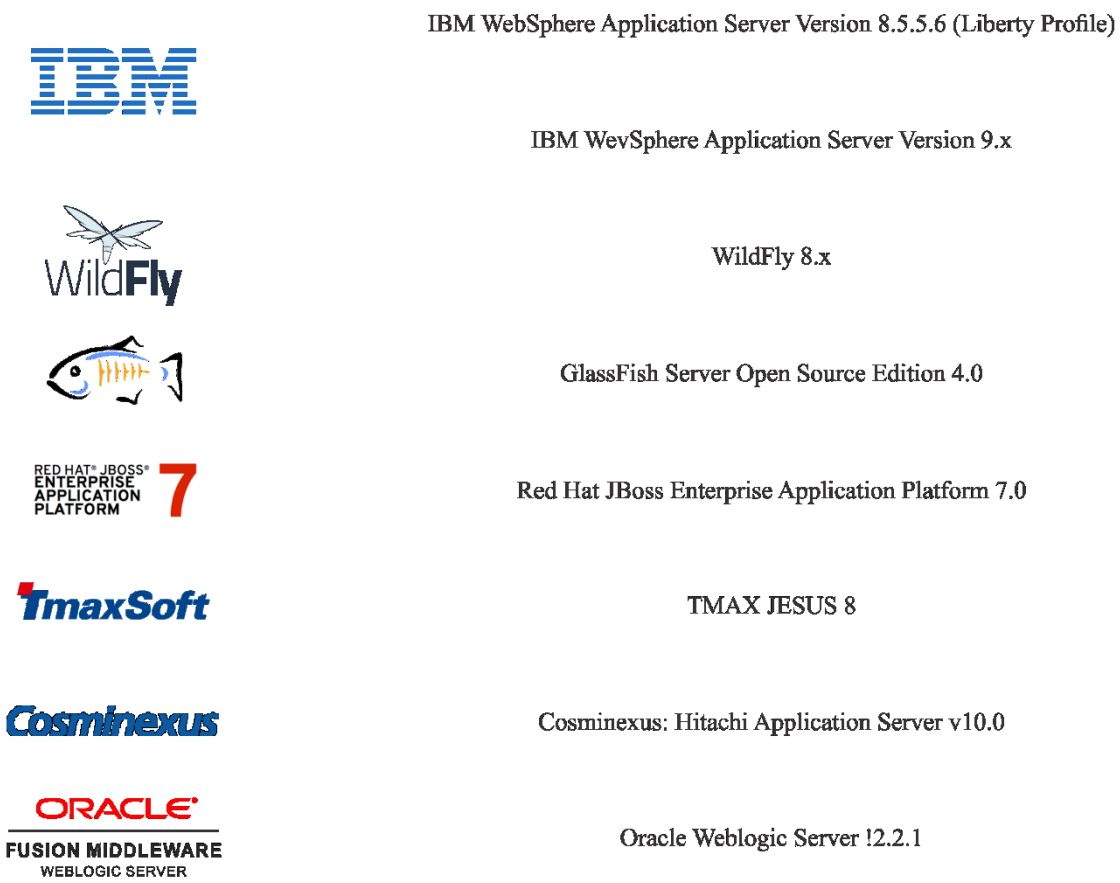
Vsi profili Jave EE si delijo množico skupnih značilnosti. Te so na primer: imenovanje, vstavljanje virov in varnostne zahteve. Ta lastnost profilov zagotavlja stopnjo poenotenja vseh izdelkov in posredno aplikacij, ki spadajo pod dežnik platforme Jave EE. Prav tako zagotavlja tudi to, da razvijalci seznanjeni z določenim profilom ali celotno platformo, enostavno preidejo na druge profile, brez pretiranega znanja in izkušenj [27].

V naslednjih treh podpoglavjih bodo podrobneje opisano vsi zgoraj našteti profili.

5.1 Full Profile

Poln profil Jave EE zajema vse tehnologije specifikacije Java EE [27]. Slednje so naštete v tabeli 2.2. V zadnjem stolpcu so s kljukico označen vse obvezne tehnologije, z o pa izbirne. Z novimi verzijami platform se bodo spreminjale tudi te zahteve.

Slika 5.1 prikazuje nekatere izmed najpopularnejših Java EE 7 aplikacijskih strežnikov. To so tisti strežniki, ki podpirajo produkte polnega profila.



Slika 5.1: Kompatibilne implementacije polnega profila Jave EE 7

Poln profil uporabimo za tiste aplikacije, ki uporabljajo oddaljeni vmesnik. Razlog za to je ta, da poln profil zajema vse tehnologije Jave EE, torej tudi tiste, ki so napredne.

Tehnologije v polnem profilu so prevelike za spletne aplikacije. Ob tem pa tudi velikokrat vse tehnologije niso uporabljene. Iz teh dveh razlogov se je razvil profil, ki zajema podmnožico tehnologij polnega profila in je namenjen samo spletnim aplikacijam ter je v primerjavi s polnim

profilom bolj omejen, hitrejši, manjši in osnovnejši. Ta profil se imenuje spletni profil in bo podrobneje opisan v naslednjem podpoglavju.

5.2 Web Profile

Spletni profil Jave EE 7 je profil, ki zajema tehnologije platforme Java EE za izdelavo spletnih aplikacij in je prvi profil platforme Java EE, ki je bil predstavljen že v Javi EE 6 [27].

Do ideje za izdelavo spletnega profila so prišli zato, ker ima večina spletnih aplikacij pomembne zahteve na področjih upravljanja transakcij, varnosti in trajnosti podatkov *(persistence)*. Takšne zahteve pa lahko brez težav rešijo vzpostavljene tehnologije Java EE, kot so EJB Lite, JPA, JTA. Teh pa samostojni servletni vsebniki ne podpirajo. Tako lahko spletni profil z vključevanjem številnih API-jev izboljša raven razvoja spletnih aplikacij, ki uporabljajo platformo Java z vnaprej nameščenimi, vnaprej integriranimi in popolnoma preizkušenimi funkcijami spletne infrastrukture [29].

Glede vprašanja, katere in koliko tehnologij bi bilo dobro uporabiti, pa obstajata dva pogleda.

Po eni strani si za razvoj »moderne« spletne aplikacije želimo razumno popoln sklad, sestavljen iz standardnih API-jev. Ta naj bi bil zmožen objektivnega obravnavanja potreb velikega števila spletnih aplikacij. Sklad bi se prav tako moral enostavno nadgrajevati ter znati obravnavati tudi vse preostale potrebe razvijalca [30].

Po drugi strani pa si želimo, da bi uravnotežili željo glede omejitve odtisa spletnih vsebnikov, tako v fizični kot v pojmovni ravni. Za razvijalca, ki se uči spletnega profila, je bolje, če imajo majhen, osredotočen profil s čim manj prekrivanji med tehnologijami [30].

Zato si je strokovna skupina pri definiciji spletnega profila prizadevala, da bi našla kompromisno rešitev med zgornjima dvema sklopoma zahtev. V smislu popolnosti mora spletni profil ponujati celovit sklad s tehnologijami. Kar se tiče enostavnosti pa izpusti mnoge poslovne zaledne API-je, ki so del platforme Jave EE [30].

Zato si kljub temu, da je spletni profil funkcijsko bogat, razvojna skupina profila še vedno prizadeva za preprostost. Slednje je razvidno iz tega, da profil izpušča številne API-je, ki so del polnega profila Jave EE. Če so izpuščeni API-ji kasneje potrebni, pa lahko razvijalci preprosto premestijo svoje aplikacije na poln profil Jave EE [29].

Kot je že zgoraj omenjeno je spletni profil sestavljen iz celotnega sklopa Java EE API-jev, ki so potrebni za razvoj modernih spletnih aplikacij. Vsi tisti, ki so vključeni v spletni profil Jave EE 7, so obkljukani v predzadnjem stolpcu tabele 2.2.

Vsaka različica platforme Java EE prinaša številne novosti. Ob tem pa se spreminjajo tudi tehnologije vključene v spletni profil. Tabela 5.1 prikazuje katere verzije tehnologij in tehnologije so bile vključene v spletni profilu v različnih verzijah platforme Jave EE [30].

Tehnologije	**Verzije tehnologij spletnega profila Jave EE 6**	**Verzije tehnologij spletnega profila Jave EE 7**	**Verzije tehnologij spletnega profila Jave EE 8**
Servlet	3.0	3.1	4.0
JSF	2.0	2.2	2.3
JSP	2.2	2.3	2.3
EL	2.2	3.0	3.0
JSTL	1.2	1.2	1.2
CDI	1.0	1.1	2.0
DI	1.0	1.0	1.0
Managed Beans	1.0	1.0	1.0
Interceptors	1.1	1.2	1.2
EJB Lite	3.1	3.2	3.2
JPA	2.0	2.1	2.2
JTA	1.1	1.2	1.2
Bean Validation	1.0	1.1	2.0
Common Annotations	1.1	1.2	1.3
Debugging support	1.0	1.0	1.0
WebSocket	×	1.0	1.1
JAX-RS	×	2.0	2.1
JSON-P	×	1.0	1.1
JSON-B	×	×	1.0
Java EE Security API	×	×	1.0
JASPIC	×	×	1.1

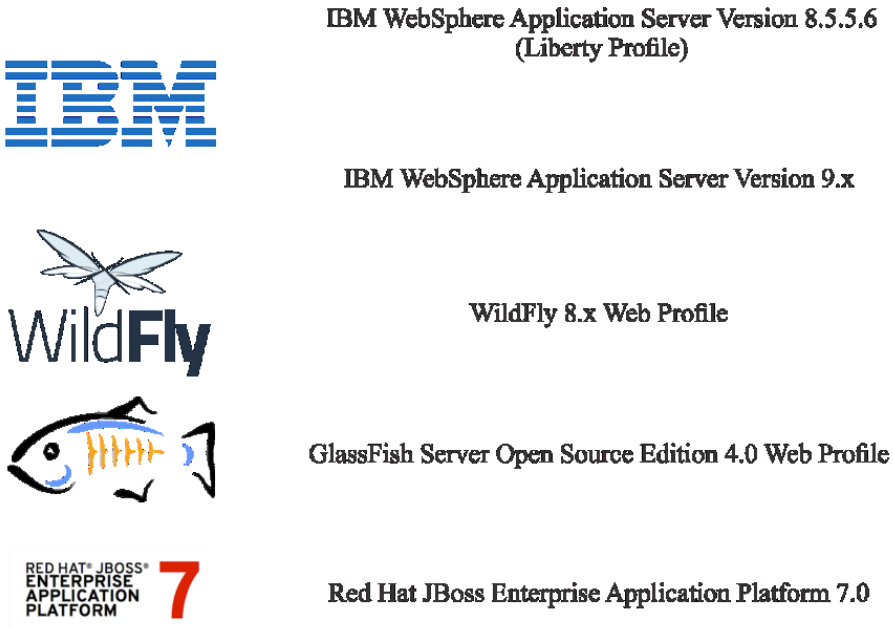
Tabela 5.1: API-ji iz Java EE 7 in 8, ki so vključeni v spletnem profil. [30], [31].

Iz tabele je razvidno, da je spletni profil Jave EE 7 dodal podporo za HTML5 z WebSocket, JSON, JAX-RS 2.0 in številne druge tehnologije [30], [31].

S prihodom platforme Java EE 8, pa so v spletnem profilu posodobljene tehnologije: WebSocket, JSON-P, Servlet, JSF, CDI, Bean validation, JPA, Common Annotations in JAX-RS. Ob tem so dodane tudi tri tehnologije: JSON-B, Java EE Security API in JASPIC [30], [31].

V nobeni različici spletnega profila ni izbirnih komponent. Produkti spletnega profila lahko vsebujejo tehnologije iz platforme Java EE, ki niso naštete v tabeli 5.1. A morajo biti skladne z njihovimi zahtevami združljivosti. Ob tem morajo podpirati tudi module spletnih aplikacij (.war datoteke). Ostali moduli ne rabijo biti podprti [30].

Slika 5.2 prikazuje nekatere izmed najpopularnejši aplikacijskih strežnikov spletnega profila Jave EE 7. To so tisti strežniki, ki podpirajo produkte spletnega profila.



Slika 5.2: Kompatibilne implementacije spletnega profila Jave EE 7

Spletni profil Jave EE je namenjen spletnim aplikacijam, ki ne zahtevajo naprednih tehnologij Jave EE. Primer teh so podpora za oddaljeni vmesnik, celotna specifikacija EJB API-ja in JMS-API. Primera aplikacij spletnega profila sta aplikacija grafikona in aplikacija klepeta v realnem času.

S časom pa so mikrostoritve postale vse bolj popularne, zato je naraščala potreba po profilu, ki bi bil namenjen samo njim. Iz tega razloga se je razvil v tem trenutku najmlajši MicroProfile, ki po podrobneje opisan v naslednjem poglavju.

5.3 Eclipse MicroProfile

Eclipse MicroProfile je osnovna definicija platforme, ki optimizira Javo EE za arhitekturo mikrostoritev. MicroProfile ni standarden profil glede na specifikacijo Java EE, ampak ga

razvija skupnost. Torej je sodelovanje pri razvoju odprto za katerokoli skupnost, podjetje, skupine ali posameznike. Slednji imajo aktivno vlogo pri definiciji in načrtovanju profila. Omeniti je potrebno tudi, da veliko ljudi, ki sodeluje pri razvoju tega profila, niso le člani skupnosti Jave EE, ampak so tudi člani strokovnih skupin pri razvoju Jave EE 8 [28].

Skupnost uporablja tehnologije poslovne Jave, kot je Java EE skupaj z arhitekturo mikrostoritev že kar nekaj časa. Inovacija MicroProfile-a, pa je privedla do še več različnih pristopov, tako v implementaciji produktov, kot tudi pri oblikovanju vzorcev. Člani skupnosti, ki razvijajo profil, bodo še naprej samostojno razvijali inovacije. MicroProfile jim bo omogočal sodelovanje na tistih področjih, kjer je kaj skupnega. Če se izhodiščna platforma določi na takšen način, bodo imeli razvijalci stopnjo prenosljivosti z več izbirami za implementacijo. MicroProfile je trenutno namenjen spodbujanju inovativnosti, v prihodnosti pa lahko napreduje tudi v standard [28].

Prvotno načrtovane osnovne tehnologije zanj so: JAX-RS, CDI in JSON-P. Te so izbrane zato, ker so najbolj pogosto uporabljene tehnologije v razvoju mikrostoritev. MicroProfile pa želi obstoječa orodja združiti z novimi in tako ustvariti osnovno platformo, ki bo optimizirana za arhitekturo mikrostoritev [28].

V času pisanja diplomske naloge, so bile razvite štiri različice MicroProfile-a, ki so spodaj tudi podrobneje opisane.

Prva različica MicroProfile 1.0 je bila izdan preden je MicroProfile postal Eclipse-ov projekt. Sestavljen pa je bil samo iz specifikacij Java EE, ki so bile prvotno načrtovane. To so JAX-RS 2.0, CDI 1.2 in JSON-P 1.0 [32].

Slika 5.3 prikazuje aplikacijske strežnike, ki podpirajo produkte MicroProfile-a 1.0.

Slika 5.3: Aplikacijski strežniki, ki podpirajo produkte MicroProfile-a 1.0.

Druga različica je MicroProfile 1.1, ki je prva izdaja profila, odkar je postal projekt Eclipse-a. Zaradi časovne omejenosti, je sestavljen iz celotnega MicroProfile-a 1.0 in MicroProfile Config 1.0 [32].

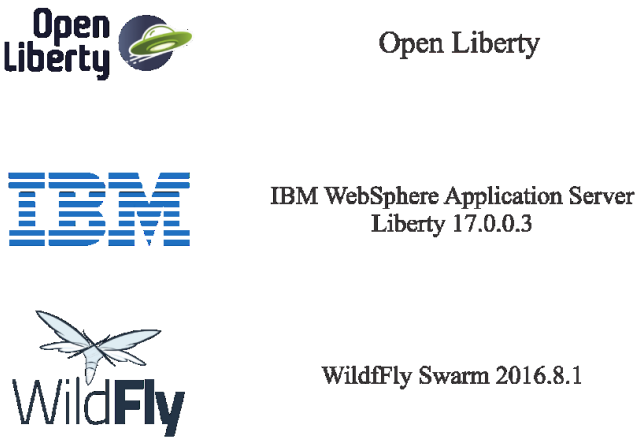
Slika 5.4 prikazuje aplikacijske strežnike, ki podpirajo produkte MicroProfile-a 1.1.

Slika 5.4: Aplikacijski strežniki, ki podpirajo produkte MicroProfile-a 1.1.

Tretja različica je MicroProfile 1.2, ki je druga izdaja profila od kar je projekt Eclipse-a. Ker je tudi ta izdaja bila časovno omejena vsebuje celoten MicroProfile 1.1 in spodaj naštete tehnologije.

- MicroProfile Config 1.1, ki je nadomestil MicroProfile Config 1.0
- MicroProfile Fault Tolerance 1.0
- MicroProfile Health Check 1.0
- MicroProfile Metrics 1.0
- MicroProfile JWT Authentication 1.0 [32]

Slika 5.5 prikazuje aplikacijske strežnike, ki podpirajo produkte MicroProfile-a 1.2 in MicroProfile-a 1.3, ki bo podrobneje razložen v nadaljevanju.



Slika 5.5: Aplikacijski strežniki, ki podpirajo produkte MicroProfile-a 1.2 in 1.3.

Četrta različica je MicroProfile 1.3, ki je tretja izdaja profila od kar je projekt Eclipse-a. Ker je tudi ta izdaja bila časovno omejena vsebuje celoten MicroProfile 1.2 in spodaj naštete tehnologije.

- MicroProfile Config 1.2, ki je nadomestil MicroProfile Config 1.1
- MicroProfile Metrics 1.1, ki je nadomestil MicroProfile Metrics 1.0
- MicroProfile OpenAPI 1.0
- MicroProfile OpenTracing 1.0
- MicroProfile Rest Client 1.0 [32]

Zaradi lažjega razumevanja so v tabeli 5.2 prikazane katere tehnologije in verzije tehnologij vsebujejo različice profila.

Tehnologije	**MicroProfile 1.0**	**MicroProfile 1.1**	**MicroProfile 1.2**	**MicroProfile 1.3**
JAX-RS	2.0	2.0	2.0.1	2.0.1
CDI	1.2	1.2	1.2	1.2
JSON-P	1.0	1.0	1.0	1.0
MicroProfile Config	×	1.0	1.1	1.2
MicroProfile Fault Tolerance	×	×	1.0	1.0
MicroProfile Health Check	×	×	1.0	1.0
MicroProfile Metrics	×	×	1.0	1.1

Tehnologije	MicroProfile 1.0	MicroProfile 1.1	MicroProfile 1.2	MicroProfile 1.3
MicroProfile JWT Authentication	×	×	1.0	1.0
MicroProfile OpenAPI	×	×	×	1.0
MicroProfile OpenTracing	×	×	×	1.0
MicroProfile Rest Client	×	×	×	1.0

Tabela 5.2: Tehnologije vključene v različnih verzijah MicroProfile-a [32]

Sedaj bo podrobneje opisana četrta različica profila. Ustrezne implementacije Eclipse MicroProfile-a 1.3 mora vsebovati najmanj spodaj naštete API-je. Slednji pogoj mora biti izpolnjen, zato ker razvijalcem aplikacij jamči minimalen nabor API-jev. Ta nabor pa vsekakor ni namenjen, da bi jih omejevali. Kot je že bilo omenjeno, implementacije profila lahko vsebujejo API-je, ki spodaj niso našteti in tudi novejše verzije spodaj naštetih. A morajo razvijalci v tem primeru biti pozorni na to, da te aplikacije niso nujno prenosljive.

API-ji in njihove minimalne različice, ki jih morajo vsebovati ustrezne implementacije Eclipse MicroProfile-a 1.3:

- Java SE 8.
- CDI 1.2, ki ponuja osnovo za vse večje število API-jev vključenih v profil.
- JAX-RS 2.0.1, ki zagotavlja standardne API-je za odjemalce in strežnike za RESTful komunikacijo aplikacij MicroProfile-a.
- JSON-P 1.0, ki zagotavlja standardne API-je za obdelavo JSON dokumentov.
- Common Annotations for the Java Platform 1.2, ki zagotavljajo anotacije za skupne semantične koncepte v različnih individualnih tehnologijah na platformama Java SE in Java EE.
- MicroProfile Config 1.2, ki zagotavlja aplikacijam in mikrostoritvam sredstva za pridobitev konfiguracijskih lastnosti preko več okolijskih virov, tako zunanjih kot notranjih aplikacij. Ob tem jih preko injekcije odvisnosti ali iskanja naredi dosegljive.
- MicroProfile Fault Tolerance, ki omogoča ločevanje logike izvajanja od poslovne logike.
- MicroProfile Health Check, ki omogoča zmožnost preiskovanja stanja računalniških vozlišč iz drugega stroja.

- MicroProfile Metrics 1.1, ki ponuja strežnikom MicroProfile-a enoten način izvoza metrik k upravljalcem. Ob tem bo Metrics zagotovil tudi skupni Java API za razkrivanje njihovih podatkov o telemetriji.
- MicroProfile JWT Authentication, ki zagotavlja na vlogi temeljeno kontrolo dostopnosti (RBAC) končne točke mikrostoritve z uporabo OpenID Connect (OIDC) in JSON Web Tokens (JWT).
- MicroProfile OpenAPI ponuja enoten Java API za specifikacijo OpenAPI verzije 3, ki ga lahko vsi razvijalci aplikacij uporabljajo za razkritje dokumentacija API-ja.
- MicroProfile OpenTracing definira API in z njim povezana vedenja, ki storitvam omogočajo enostavno sodelovanje v porazdeljenem okolju sledenja.
- MicroProfile Rest Client ponuja varen pristop za klicanje storitev RESTful preko HTTP-ja [32].

Zelo pomembno je povedati tudi nekaj o lahkem odprtokodnem ogrodju KumuluzEE, ki ga razvija slovensko zagonsko podjetje in je bilo nagrajeno s prestižno nagrado Java Duke's Choice Award. Namenjeno je za:

- razvoj mikrostoritev, z uporabo standardnih tehnologij Jave EE in
- migracijo obstoječih aplikacij Java EE na mikrostoritve [33].

KumuluzEE omenjamo v tem podpoglavju, zato ker je popolnoma skladen s specifikacijo Eclipse MicroProfile 1.2. Zagotavlja izvedbe vseh API-jev MicroProfile: Config 1.1, Health Check 1.0, Metrics 1.0, Fault Tolerance 1.0 in JWT Authentication 1.0. Kmalu bo KumuluzEE popolnoma skladen tudi s specifikacijo MicroProfile 1.3 [33].

Profile, predstavljene v tem poglavju, podpirajo aplikacijski strežniki, ki bodo predstavljeni v naslednjem poglavju.

6 Aplikacijski strežniki Jave EE

Aplikacijski strežnik Jave EE je strežniška aplikacija, ki izvaja API-je platforme Jave EE in ponuja standardne storitve Jave EE. Aplikacijski strežnik Jave EE večinoma imenujemo kar strežnik Jave EE. Aplikacijski strežnik pa mu rečemo zato, ker omogoča posredovanje aplikacijskih podatkov odjemalcem. Tako kot spletni strežniki posredujejo spletne strani spletnim brskalnikom [1].

Aplikacijski strežniki Jave EE gostijo več tipov komponent aplikacij, ki ustrezajo nivojem v več-nivojski aplikaciji. Strežnik Java EE ponuja storitve tem komponentam v obliki vsebnika [1]. Če povežemo pravkar napisano z znanjem iz tretjega poglavja, ugotovimo da je aplikacijski strežnik Java EE v bistvu strežniški nivo in ponuja dva vsebnika in sicer spletni in vsebnik EJB. Vsak izmed njiju ima zahtevane tehnologije Jave EE, ki pa temeljijo na Javi SE. Slednje je tudi prikazano na slikah 3.4 in 3.5.

Obstaja več komercialnih in odprtokodnih aplikacijskih strežnikov. Ti omogočajo Java EE razvijalcem, da razvijejo in naložijo Java EE skladne aplikacije [34].

Primeri odprtokodnih aplikacijskih strežnikov so:

- Oracle-ov GlassFish,
- Red Hat-ov WildFly in
- Appache-ov Tomcat in Geronimo [34].

Primeri komercialnih aplikacijskih strežnikov sta:

- Oracle-ov WebLogic in
- IBM-ov WebSphere [34].

Tabela 6.1 prikazuje od katere verzije nekaterih izmed zgoraj naštetih aplikacijskih strežnikov sta podprti Java EE 7 in 8. V oklepaji je napisano po kolikšnem času po njuni izdaji so dodali podporo z vsako izmed njiju.

Aplikacijski strežnik	Java EE 7	Java EE 8
GlassFish	Od verzije 4.0: 12.06.2013 (dva meseca po izdaji Jave EE 7)	Od verzije 5.0: 21.09.2017 (isti dan, kot je bila izdana Java EE)
WildFly	Od verzije 8.0: 11.02.2014 (skoraj osem mesecev po izdaji	Ni še v celoti podprta, a na Red Hat-u aktivno delajo na tem, da bi bila.
Tomcat	Od verzije 8.0: 25.06.2014 (malo več kot eno leto po izdaji Jave EE 7)	Od verzije 9.0: 18.01.2018 (skoraj štiri mesece po izdaji Jave EE 8)
WebLogic	Od 12.2.1: 26.10.2015 (malo več kot dve leti po izdaji Jave EE 7)	Ni podprta, a imajo v načrtu, da bo bila še letos.
WebSphere Traditional	Od verzije 9.0	Ni še podprta
WebSphere Liberty profil	Od verzije 8.5.5.6	Od november 2017 podprta na WebSphere Liberty beta (skoraj dva meseca po izdaji Jave EE 8)

Tabela 6.1: Podpora aplikacijskih strežnikov za Javo EE 7 in 8

Preden se podjetje odloči, kateri aplikacijski strežnik bo uporabljalo, mora podrobno preučiti kakšne aplikacije bodo razvijali in kateri strežniki podpirajo njihove zahteve.

Aplikacijski strežniki imajo za vsak Java EE API, ki ga potrebujejo referenčno implementacijo. Tako je na primer v aplikacijskem strežniku GlassFish:

- Jersey referenčna implementacija za JAX-RS,
- Mojarra za JSF in
- Open Message Queue (Open MQ) za JMS.

EclipseLink je na primer odprtokodna referenčna implementacija za JPA, referenčna implementacija za novo dodani Security API v Javi EE, pa se imenuje Soteria.

IBM-ov WebSphere Application Server je vodilna fundacija programske opreme za storitveno usmerjeno arhitekturo aplikacij in poslovnih storitev [35]. Zato bo prav ta aplikacijski strežnik podrobneje opisan v naslednjem podpoglavju.

6.1 WebSphere Application Server

WebSphere Application Server (WAS) ponuja okolje za implementacijo rešitve vmesne opreme. Jedrna komponenta je izvajalno okolje aplikacijskega strežnika. To okolje zagotavlja infrastrukturo za izvajanje poslovnih aplikacij. Aplikacijski strežnik ponuja množico storitev,

ki jih lahko poslovna aplikacija uporabi. Ob tem pa tudi služi tudi kot platforma za razvoj in namestitev teh aplikacij. Ker se poslovne potrebe razvijajo, se posledično razvija tudi WAS. Tako je WAS od leta 1998 zrasel in sprejel številne nove tehnologije in standarde [35].

V sklopu arhitekture več-nivojskih aplikacij se na WAS-u lahko opravljajo naslednje funkcije:

- Hiter in enostaven razvoj in namestitev storitev.
- Zagon storitev na varnem, razširljivem in zelo dostopnem okolju.
- Povezavo sredstev programske opreme in razširitev njihovega obsega.
- Upravljanje aplikacij brez napora.
- Razširitev, ko se potrebe razvijejo ter ponovno uporabo ključnih spretnosti in sredstev [35].

Različni tipi aplikacij zahtevajo različne stopnje zmogljivosti aplikacijskih strežnikov, zato je WAS na voljo v več možnih pakiranjih. Vsa pakiranja si delijo skupno podlago, vsako pa ponuja edinstvene prednosti za izpolnitev potreb aplikacij in infrastrukture, ki jih podpira. A prav zagotovo vsaj eden izmed WAS produktov izpolnjuje zahteve vsakega posameznega projekta in podpira njegovo infrastrukturo. Ob rasti podjetja pa družina WAS zagotavlja tudi migracijsko pot do bolj zapletenih konfiguracij. Na voljo so naslednji paketi:

- WAS Express,
- WAS Base,
- WAS Network Deployment,
- WAS for z/OS,
- WAS for Developers,
- WAS Hypervisor Edition,
- WAS Liberty Core in
- WAS Community Edition [35].

WAS je od verzije 8.5 sestavljen iz dveh okolij za izvajanje strežnika aplikacij. Imenujemo ju tudi profila. Vsak paket aplikacijskega strežnik WebSphere vsebuje obe vrsti profila [36].

Prvi je Traditional WAS. Zanj se uporabljata tudi imeni klasični ali full profil WAS-a [36]. Je izvirna arhitektura WAS-a, ki se je začela leta 1998 in se je razvila v industrijsko močen vsebnik Jave EE. Podrobneje bo opisan v poglavju 6.1.2.

Drugi je Liberty. Zanj se uporabljata tudi imeni Liberty WAS ali Liberty profil. Je vsebnik Jave EE nove generacije. Bil je predstavljen v WAS 8.5 in je na voljo tudi kot samostojna

ponudba, imenovana WAS Liberty Core, ki pa je bila predstavljena v WAS 8.5.5 [36]. Podrobneje bo opisan v poglavju 6.1.3.

Vsaka verzija WAS-a podpira različne specifikacije ali API-je. Primerjava teh za oba profila se nahaja v podpoglavjema 6.1.2.1 in 6.1.3.1.

Naslednje podpoglavje pa bo namenjeno primerjavi Liberty in klasičnega WAS profila. To poglavje bo še posebej v pomoč tistim razvijalcev, ki imajo težavo pri izbiri profila.

6.1.1 Primerjava klasičnega in Liberty profila

Obstajajo številni vidiki, ki jih je potrebno upoštevati, ko izbiramo med klasičnim in Liberty profilom. Če ima razvijalec zahtevo, ki jo podpira le eden izmed njiju (na primer poseben API), je izbira preprosta. A ker Liberty profil vse bolj raste in ponuja vse več tehnologij, je izbira postala veliko bolj zapletena in je potrebno zelo natančno pretehtati med vsemi prednostmi in slabostmi, ter se komaj nato dokončno odločiti [37].

Klasični profil je še vedno v celoti podprt in strateški. Zato v primeru, ko je aplikacija že nameščena na tem okolju in ustreza razvijalčevim potrebam, ni razloga za premikanje aplikacij iz obstoječega okolja. Če pa razvijalec začenja z novim projektom, ali pa potrebuje lažje, bolj fleksibilno okolje ali pa želi uporabiti novo okolje v oblaku, je lahko Liberty profil dobra izbira zanj [37].

Najbolj očitna razlika med profiloma je izvajalno okolje arhitekture. Klasični profil ima precej fiksno množico aplikacijskih storitev, ki se naložijo in inicializirajo ob zagonu. Med njih spada tudi celotna platforma Jave EE z nekaj razširitvami, ki pa so prilagodljive. Kar pomeni, da ta profil zagotavlja izvajanje s celotnim programskim modelom, ki je na voljo privzeto. Pri tem so vse storitve, aplikacije in viri v celoti inicializirni, ko se zagon strežnika dokonča [37].

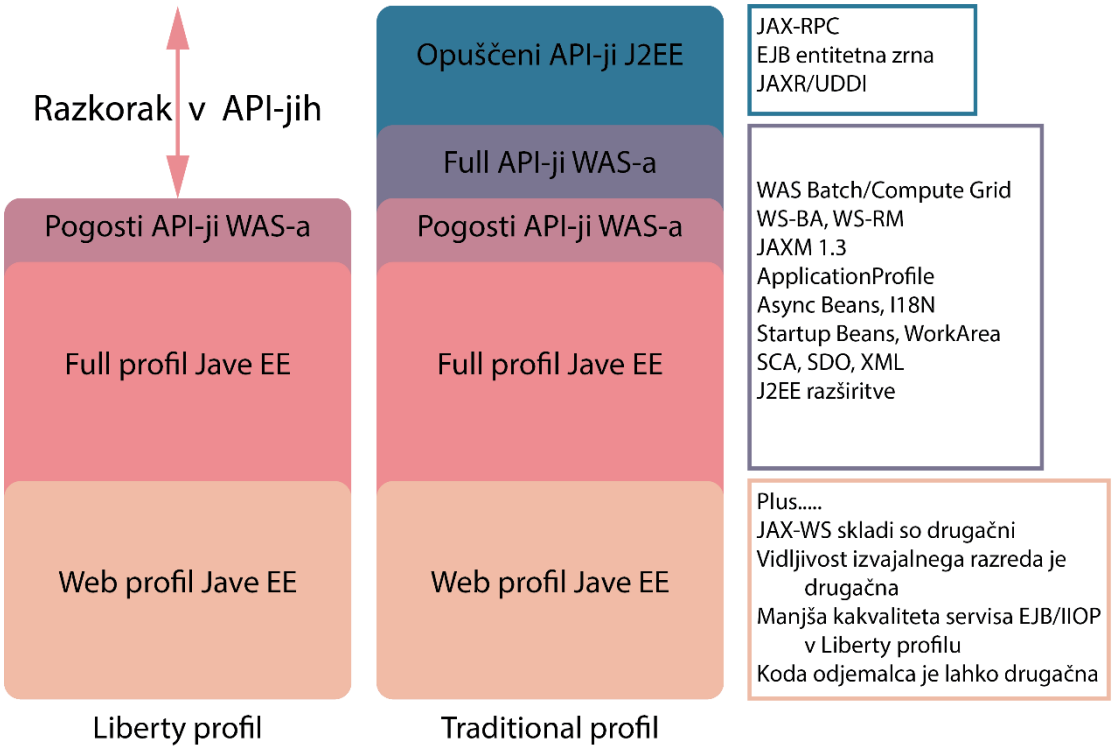
Po drugi strani pa ima Liberty profil majno jedro, ki je edino, ki se naloži in inicializira ob zagonu. Uporabnik pa določi na drobnozrnatem nivoju katere storitve so potrebne. Namen slednjega je ta, da vsaka instanca strežnika tesno ustreza potrebam aplikacije, ki se izvaja. Takšen model je primer za okolje deljenih virov, kot je oblak. Inicializacija storitev, aplikacij in virov je v splošnem zakasnjena dokler se ne uporabljajo. Takšen model strežnika zagotavlja hitrejši zagon strežnika in zmanjša odtis pomnilnika [37].

Pravkar omenjene osnovne razlike v izvajalnem okolju arhitekture so pogosto najpomembnejši dejavniki pri izbiri. Še posebej zato, ker se vse več aplikacij premika v oblak, kar pa pomeni, da delitev virov postaja vedno bolj pomembna [37].

Obstajajo še številna področja na katerih lahko primerjamo profila, a se bomo v diplomski nalogi osredotočili samo še na eno, ki je najbolj kritično. To je področje tehnologij, ki ju profila ponujata.

Zelo pomembno je, da razvijalec preveri, ali so tehnologije, ki jih uporablja aplikacija, na voljo v profilu, ki ga nameravamo uporabiti. Če na primer aplikacija uporablja tehnologijo, ki v Liberty profilu ni na voljo, potem se lahko razvijalec enostavno odloči za uporabo klasičnega profila. Razvijalec pa vsekakor mora biti pozoren pri razvoju novih aplikacij, saj ima Liberty profil celoten nabor API-jev Jave, ki jih posodablja hitreje kot klasičen profil. Tako, da je morda za novejše aplikacije Liberty profil boljša izbira [37].

Oba profila podpirata vse API-je platforme Java EE. Razlikujeta se v tem, da lahko razvijalec v Liberty profilu izbira kateri izmed teh API-jev bodo vključeni v vsaki izmed instanc strežnika. V klasičnem profilu pa so vsi API-ji prisotni v vseh instancah. Ob tem pa starejši API-ji, ki so bili preseženi skozi leta zaradi izboljšav specifikacije Jave niso bili replicirani na Liberty profil. Torej obstaja zelo majhna možnost, da bodo API-ji, ki so bili opuščeni v Java EE specifikaciji ali klasičnem profilu, del Liberty profila. Izjema so le tisti API-ji, po katerih obstaja veliko povpraševanje. Slika 6.1 prikazuje visoko nivojski pogled razkoraka med API-ji klasičnega in Liberty profila [37].



Slika 6.1: Razkorak med API-ji klasičnega in Liberty profila

6.1.2 WAS – klasični profil

Klasični profil je tradicionalno na voljo z WAS paketi. Aplikacijski strežnik, ki ima ta profil, je sestavljen iz širokega spektra izvajalnih komponent, ki so na voljo vedno, ko je strežnik zagnan. Ob aplikacijskih strežnikih pa ta profil podpira tudi ustvarjanje definicije generičnega strežnika. Ta se uporabi za konfiguriranje drugih strežnikov ali procesov, ki so potrebni za podporo okolja aplikacijskega strežnika. Druge zmogljivosti, kot so združevanje aplikacijskih strežnikov za uravnoteževanje obremenitve in visoke razpoložljivosti, pa se razlikujejo glede na paket WAS-a [35].

V naslednjem podpoglavju bo podana primerjava zadnjih dveh večjih verzij klasičnega profila. Pri čemer bomo videli tudi katere tehnologije in njihove verzije podpira klasični profil.

6.1.2.1 Primerjava verzije 8.5.5 in verzije 9

Klasični profil verzije 8.5.5 in 9.0 podpira verzije Java EE in Java SE platforme prikazane v tabeli 6.2.

Specifikacija ali API	**Verzija 8.5.5**	**Verzija 9.0**
Java EE	Java EE 6	Java EE 7
Java SE	Java SE 6 in 7	Java SE 8

Tabela 6.2: Podpora Java EE in Java SE platform na klasičnem profilu verzije 8.5.5 in 9 [35], [38]

Tabela 6.3 pa prikazuje, katere specifikacije in njihove verzije podpira klasični profil verziji 8.5.5 in 9.0 za Javo EE. Vsekakor pa je je podpora za določeno specifikacijo kompatibilna tudi s starejšimi verzijami te specifikacije [35],[38].

Specifikacija ali API	**Verzija 8.5.5**	**Verzija 9.0**
JAX-RS	1.1	2.0
Java APIs for WSDL	1.2	✓
Implementing Enterprise Web Services	1.3	1.4
JAX-WS	2.2	2.2
JAXB	2.2	2.2
Web Services Metadata for the Java Platform	✓	✓
JAX-RPC	1.1	1.1 (izbirna)
Java APIs for XML Messaging (JAXM)	1.3	
JAXR	1.0	1.0 (izbirna)
SAAJ	1.3	1.3
Java Servlet	3.0	3.1

Specifikacija ali API	Verzija 8.5.5	Verzija 9.0
JSF	2.0	2.2
JSP	2.2	2.3
EL	2.2	3.0
Managed Beans	1.0	
JSTL	1.2	1.2
Debugging Support for Other Language	1.0	1.0
CDI	1.0	1.2
DI	1.0	1.0
Bean Validation	1.0	1.1
EJB	3.1	3.2
Interceptors	1.1	1.2
JCA	1.6	1.7
JPA	2.0	2.1
Common Annotations for the Java Platform	1.1	1.2
JMS	1.1	2.0
JTA	1.1	1.2
JavaMail	1.4	1.5
JASPIC	✓	1.1
JACC	1.3	1.5
Java EE Application Deployment	1.2	1.2 (izbirna)
J2EE Management	1.1	1.1
JAXP	1.3	1.4
JDBC	4.0	4.1
JMX	2.0	
JAF	1.1	1.1
StAX	1.0	1.0
JSON-P		1.0
WebSocket		1.0 in 1.1
Batch Processing		1.0
Concurrency Utilities		1.0

Tabela 6.3: Podprte specifikacije in API-ji na WAS klasičnem profilu verzije 8.5.5 in 9 [35], [38]

6.1.3 WAS – Liberty profil

IBM WAS Liberty je poenostavljeno, lahko-razvojno in samostojno izvajalno okolje aplikacij. To okolje ima spodnje značilnosti.

- Preprosto nastavljivo. Konfiguracija se bere iz datoteke XML s tekstovno urejevalnikom prijazno sintakso.

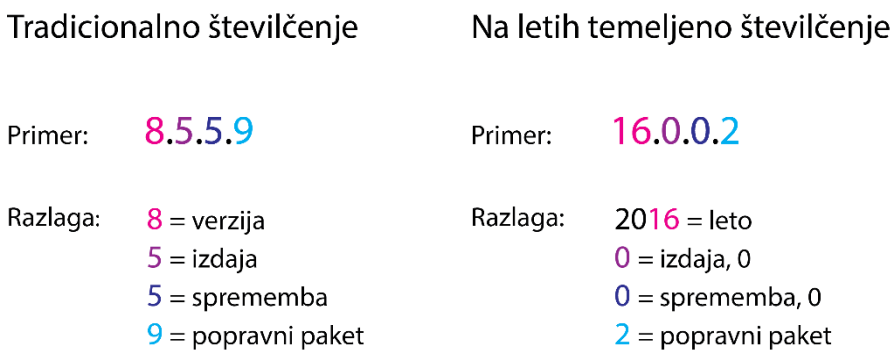
- Dinamično in zanesljivo. Izvajalno okolje naloži samo tisto, kar potrebuje aplikacija in ponovno sestavi izvajalno okolje, kot odgovor na konfiguracijske spremembe.
- Hitro. Strežnik z osnovno spletno aplikacijo starta v manj kot petih sekundah.
- Razširljivo. Liberty profil zagotavlja podporo za razširitve uporabnikov in produktov [39].

Liberty profil podpira podmnožico programskega modela, ki je na voljo s klasičnim profilom WAS-a. Vsaka aplikacija, ki se izvaja v Liberty profilu, se lahko izvaja tudi v klasičnem profilu WAS-a. Ta profil je zelo dobra izbira za razvijalce, ki razvijajo spletne aplikacije, a ne potrebujejo celotnega okolja Jave EE [36].

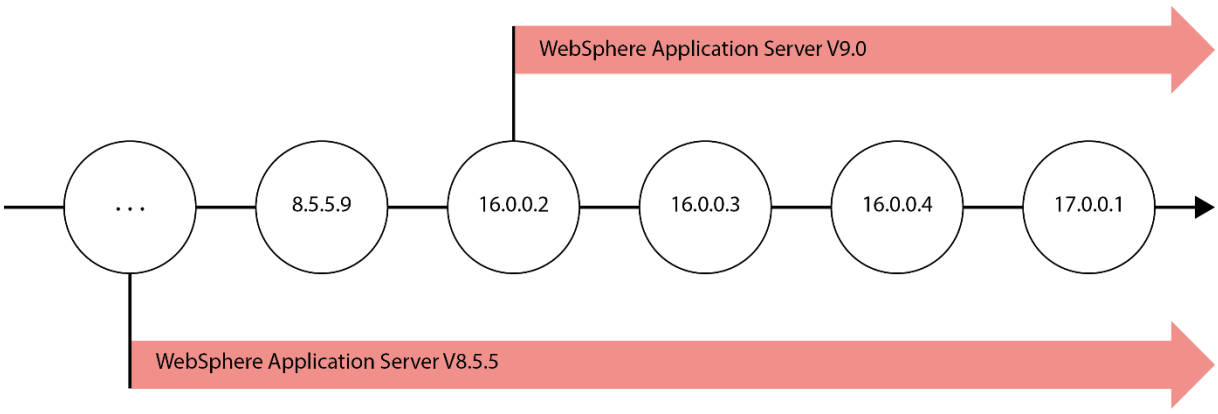
V naslednjem podpoglavju bo podana primerjava Liberty profila verzije 8.5.5 in 18.0.0.1. Zakaj je prišlo do takšnega razmika med verzijama bo razloženo v naslednjih odlomkih.

Avgusta 2016 je IBM spremenil številčenje verzij Liberty profila. Tradicionalno so nove funkcionalnosti izdane v večjih kosih kot nova verzija produkta. Manjše posodobitve in popravki pa so dostavljeni kot popravni paketi med verzijami To pomeni, da bodo razvijalci morda morali počakati nekaj let do nove verzije, da bodo lahko preizkusili novo tehnologijo. Z modelom Liberty pa so nove funkcionalnosti stalno dostavljene kot modularne opcijske funkcije. Kar pomeni, da ni potrebe po potrpežljivem čakanju na naslednjo veliko različico, ampak samo do naslednjega rednega posodabljanja. Na ta način lahko razvijalec sprejme funkcije, ki jih želi, ko je pripravljen. Takšna migracija se imenuje ničelna migracija in omogoča, da lahko razvijalec v celosti sprejme neprekinjeno dostavo in brez težav preide na novo raven [40].

Ker nove različice več ne prinašajo večjih količin novih informacij, je sedaj poudarek na paketnih popravkih. Pri takšni dostavi z enojnim tokom razvijalec dobi iste pakete popravkov za Liberty s posodobitvami v istem časovnem razporedu, ne glede na to ali je kupil WAS različice 8.5.5 ali 9.0. Paralelno s to novostjo pa se je uvedel tudi nov način številčenja različic. Ta se je začel s paketnim popravkom 16.0.0.2 in ga podrobneje opisuje slika 6.2. Za lažjo časovno predstavo začetka novega številčenja različic, je na sliki 6.3 prikazan časovni trak različic.



Slika 6.2: Razlaga tradicionalnega in na letih temelječega številčenja [40]



Slika 6.3: Prikaz začetka uporabe na letih temelječega številčenja v primerjavi z tradicionalnim številčenjem [40].

Trenutno najnovejši paketni popravek je 18.0.0.1. Iz tega razloga bo v naslednjem podpoglavju podana primerjava različic Liberty profila 8.5.5 po tradicionalnem načinu številčenja in najnovejše verzije 18.0.0.1.

6.1.3.1 Primerjava verzije 8.5.5 in verzije 18.0.0.1

Tabela 6.4 prikazuje katere verzije Jave EE in Jave SE podpira Liberty profil verzije 8.5.5 in 18.0.0.1

Specifikacija ali API	Verzija 8.5.5	Verzija 18.0.0.1
Java EE	Java EE 6 (Web profil) in Java EE 7 od 8.5.5.6	Java EE 6 (Web profil) in Java EE 7
Java SE	Java SE 6 ter 7 od 8.5.5.2 in Java SE 8 od 8.5.5.5	Java SE 6 do 17.0.0.2 in Java SE 7 in višje

Tabela 6.4: Podpora Java EE in Java SE platform na WAS Liberty profilu verzije 8.5.5 in 18.0.0.1 [41]

Tabela 6.5 pa prikazuje, katere specifikacije in njihove verzije podpira Liberty profil verzij 8.5.5 in 18.0.0.1 za Javo EE [35], [38].

Specifikacija ali API	Verzija 8.5.5	Verzija 18.0.0.1
JAX-RS	1.1	2.0
Java APIs for WSDL	1.2	✓
Implementing Enterprise Web Services	1.3	1.4
JAX-WS	2.2	2.2
JAXB	2.2	2.2
Web Services Metadata for the Java Platform	✓	✓
SAAJ	1.3	1.3
Java Servlet	3.0	3.1
JSF	2.0	2.2
JSP	2.2	2.3
EL	2.2	3.0
Managed Beans	1.0	
JSTL	1.2	1.2
Debugging Support for Other Language	1.0	1.0
CDI	1.0	1.2
DI	1.0	1.0
Bean Validation	1.0	1.1
EJB	EJB Lite 3.1	EJB 3.2
Interceptors	1.1	1.2
JPA	2.0	2.1
Common Annotations for the Java Platform	1.1	1.2
JMS	1.1	2.0
JTA	1.1	1.2
JAXP	1.3	1.4

Specifikacija ali API	Verzija 8.5.5	Verzija 18.0.0.1
JDBC	4.0	4.1
JMX	2.0	
JAF	1.1	1.1
StAX	1.0	1.0
J2EE Management		1.1
JASPIC		1.1
JACC		1.5
JCA		1.7
JavaMail		1.5
JSON-P		1.0
WebSocket		1.0 in 1.1
Batch Processing		1.0
Concurrency Utilities		1.0

Tabela 6.5: Podprte specifikacije in API-ji na WAS Liberty profilu verzije 8.5.5 in 18.0.0.1 [35], [38]

Aplikacijski strežnik Jave EE je torej strežniška aplikacija, ki izvaja API-je platforme Jave EE in ponuja standardne storitve Jave EE [1]. Poznamo odprtokodne in komercialne aplikacijske strežnike. Med slednjimi je IBM-ov WAS vodilna fundacija programske opreme za storitveno usmerjeno arhitekturo aplikacij in poslovnih storitev. Različni tipi aplikacij zahtevajo različne stopnje zmogljivosti aplikacijskih strežnikov, zato je WAS na voljo v več možnih pakiranjih [35]. WAS je od verzije 8.5 sestavljen iz dveh okolij za izvajanje strežnika aplikacij med katerimi lahko razvijalec izbira. Ti dve okolji se imenujta tudi profila, in sicer klasični in Liberty profil. Vsak paket WAS-a vsebuje obe vrsti profila [36].

7 Zaključek

V diplomski nalogi smo predstavili Javo EE in novosti njenih zadnjih dveh verzij. Ker Java EE temelji na Javi SE, smo zato pred tem predstavili še Javo SE in novosti zadnjih treh verzij. Z uporabo platforme Java EE razvijamo več-nivojske aplikacije, zato smo v nalogi opisali več-nivojske aplikacije in njihovo implementacijo z Javo EE. Do sedaj so se v Javi EE razvili trije profili, zato smo obravnavali tudi njih. Na koncu je sledil še opis aplikacijskih strežnikov, na katerih se izvaja drugi nivo več-nivojskih aplikacij, in sicer nivo poslovne logike. V tem poglavju smo podrobneje opisali tudi zelo priljubljen aplikacijski strežnik WebSphere.

Za razvoj več-nivojskih aplikacij obstajajo tudi druge platforme in delovna okolja, a je zaradi priljubljenosti programskega jezika Java platforma Java EE ena izmed najbolj uporabljenih. Posledično obstaja veliko literature in skupnosti, ki so v veliko pomoč tako novim, kot obstoječim razvijalcem. Pozorni moramo biti tudi na to, da se bo v prihodnosti Java EE preimenovala v Jakarta EE in bo znana pod tem imenom. Zaradi hitrega napredovanja v računalništvu bodo kmalu po končani diplomi prišle nove različice in druge novosti, zato mora uspešen razvijalec, ki želi razvijati čim modernejše aplikacije, ves čas spremljati nove izdaje in novosti.

Ob izbiri primerne platforme ali delovnega okolja za razvoj več-nivojskih aplikacij je tudi zelo pomembna izbira aplikacijskega strežnika. Ti se med sabo zelo razlikujejo, zato mora razvijalec pred končno izbiro dobro premisliti kakšne so njegove zahteve. WebSphere ves čas sledi trendom in je zato zelo uporabljan aplikacijski strežnik v podjetjih. Malo manj je primeren za samostojne razvijalce in študente, saj je plačljiv. A tudi za njih obstajajo odlični odprtokodni aplikacijski strežniki, kot je na primer zelo popularen aplikacijski strežnik GlassFish.

Kljub hitrem razvoju računalništva, pa bo ta diplomska naloga prišla zelo prav tistim, ki se prvič srečujejo z Javo in razvojem aplikacij v Javi EE in bi radi v kratkem času osvojili osnove. Prav tako pa bo prišla zelo prav tistim, ki že poznajo osnove in so že razvijali aplikacije, a bi radi prešli na novejšo različico in bi se želeli v kratkem času seznaniti z novostmi in podrobneje raziskati tiste, ki jih potrebujejo na svojem področju. Seveda pa bo ta diplomska naloga zelo zanimiva tudi čez nekaj let, ko bo kakšen radoveden študent ali razvijalec raziskoval zgodovino razvoja aplikacij.

Literatura

[1] Oracle, 2012. *Your First Cup: An Introduction to the Java EE Platform* [online]. Redwood City: Oracle USA. Dostopno na: https://docs.oracle.com/javaee/6/firstcup/doc/firstcup.pdf. [Datum dostopa: 3. 5. 2017].

[2] Tiobe, 2018. *The Java Programming Language* [online]. Dostopno na: https://www.tiobe.com/tiobe-index/java/. [Datum dostopa: 26. 6. 2018].

[3] Oracle, 2012. *Java Garbage Collection Basics* [online]. Dostopno na: http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html. [Datum dostopa: 24. 6. 2018]

[4] Oracle. *Java EE at Glance* [online]. Dostopno na: http://www.oracle.com/technetwork/java/javaee/overview/index.html. [Datum dostopa: 3. 5. 2017].

[5] Oracle, 2014. *Java Platform, Enterprise Edition: The Java EE Tutorial, Release 7* [online]. Redwood City: Oracle USA. Dostopno na: https://docs.oracle.com/javaee/7/JEETT.pdf. [Datum dostopa: 24. 10. 2017].

[6] Oracle. *What is the difference between the JRE and the JDK* [online]. Dostopno na: https://www.java.com/en/download/faq/techinfo.xml. [Datum dostopa: 15. 5. 2018].

[7] Oracle. *Java Platform Standard Edition 8 Documentation* [online]. Dostopno na: https://docs.oracle.com/javase/8/docs/. [Datum dostopa: 24. 10. 2017].

[8] Xuelei, F. , *55 New Features in Java SE 8* [online]. Redwood City: Oracle USA. Dostopno na: http://www.oracle.com/technetwork/cn/community/developer-day/2-55-new-features-java-se-8-2202551-zhs.pdf. [Datum dostopa: 28. 11. 2017].

[9] Oracle. *The Java Tutorials* [online]. Dostopno na: https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html. [Datum dostopa: 22. 5. 2018].

[10] Oracle, 2017. *Overview of What's New in JDK 9* [online]. Redwood City: Oracle USA. Dostopno na: https://docs.oracle.com/javase/9/whatsnew/JSNEW.pdf. [Datum dostopa: 28. 11. 2017].

[11] OpenJDK, 2018. *JDK 10* [online]. Dostopno na: http://openjdk.java.net/projects/jdk/10/. [Datum dostopa 21. 6. 2018].

[12] Oracle, 2017. *Java Platform, Standard Edition HotSpot Virtual Machine Garbage Collection Tuning Guide* [online]. Dostopno na:

https://docs.oracle.com/javase/9/gctuning/garbage-first-garbage-collector.htm#JSGCT-GUID-ED3AB6D3-FD9B-4447-9EDF-983ED2F7A573. [Datum dostopa: 24. 6. 2018].

[13] Oaks S. , 2014. *Java Performance: The Definitive Guide.* Sebastopol: O'Reilly Media.

[14] Eisele M. , 2016. *Modern Java EE Design Patterns.* Sebastopol: O'Reilly Media.

[15] Oracle, 2013. *Java EE 8 Technologies* [online]. Dostopno na: http://www.oracle.com/technetwork/java/javaee/tech/java-ee-8-3890673.html. [Datum dostopa: 19. 6. 2018].

[16] Pilgrim, P. A. , 2013. *Java EE 7 Developer handbook*. Birmingham; Mumbai: Packt Enterprise.

[17] Theedom, A. , 2017. *What's new in Java EE 8* [online]. Armonk: IBM Corporation. Dostopno na: https://www.ibm.com/developerworks/library/j-whats-new-in-javaee-8/. [Datum dostopa: 28. 11. 2017]

[18] IBM. *WebSphere Application Server traditional 9.0.0.x* [online]. Dostopno na: https://www.ibm.com/support/knowledgecenter/en/SSEQTP_9.0.0/com.ibm.websphere.base.doc/ae/welc_newdeveloper.html. [Datum dostopa: 4. 11. 2017].

[19] Gupta, A. , 2013. *Java EE 7 Essentials*. Beijing: O'Reilly.

[20] Oracle, 2017. *Java Platform, Enterprise Edition (Java EE) 8. The Java EE Tutorial* [online]. Dostopno na: https://javaee.github.io/tutorial/toc.html. [Datum dostopa: 20. 6. 2018]

[21] Goncalves A. , 2014. *Beginning Java EE 7.* New York, Apress.

[22] Oracle, 2014. *Java API for JSON Binding (JSON-B)* [online]. Dostopno na: https://blogs.oracle.com/theaquarium/java-api-for-json-binding-json-b. [Datum dostopa: 12. 7. 2018].

[23] Oracle, 2018. *JavaServer Pages Technology –Frequently Asked Questions* [online]. Dostopno na: http://www.oracle.com/technetwork/java/faq-137059.html. [Datum dostopa: 12. 7. 2018].

[24] IBM. *JavaServer Pages 2.3 feature functions* [online]. Dostopno na: https://www.ibm.com/support/knowledgecenter/en/SS7K4U_liberty/com.ibm.websphere.wlp.zseries.doc/ae/cwlp_jsp23.html. [Datum dostopa: 24. 10. 2017].

[25] FreelancingGig, 2018. *What is the difference between JSP in JSF* [online]. Dostopno na: https://www.freelancinggig.com/blog/2018/03/16/difference-jsp-jsf/. [Datum dostopa: 12. 7. 2018].

[26] Oracle, 2010. *The Java EE 6 Tutorial, Volume I.* [online]. Dostopno na: https://docs.oracle.com/cd/E19226-01/820-7627/gijqy/index.html. [Datum dostopa: 30. 5. 2018].

[27] Oracle, 2013. *Java Platform, Enterprise Edition 7 Specification* [online]. Redwood City: Oracle America. Dostopno na: http://download.oracle.com/otn-pub/jcp/java_ee-7-fr-spec/JavaEE_Platform_Spec.pdf. [Datum dostopa: 3. 5. 2017].

[28] Eclipse. *Frequently Asked Questions* [online]. Dostopno na: https://microprofile.io/faq. [Datum dostopa: 3. 5. 2017].

[29] Oracle, 2013. *Introduction to Java Platform, Enterprise Edition 7* [online]. Redwood Shores: Oracle Corporation. Dostopno na: http://www.oracle.com/technetwork/java/javaee/javaee7-whitepaper-1956203.pdf. [Datum dostopa: 3. 5. 2017].

[30] Oracle, 2013. *Java Platform, Enterprise Edition 7 Web Profile Specification* [online]. Redwood City: Oracle America. Dostopno na: http://download.oracle.com/otn-pub/jcp/java_ee-7-fr-eval-spec/WebProfile.pdf. [Datum dostopa: 3. 5. 2017].

[31] Oracle, 2017. *Java Platform, Enterprise Edition 8 Web Profile Specification* [online]. Redwood City: Oracle America. Dostopno na: http://download.oracle.com/otn-pub/jcp/java_ee_web_profile-8-final-eval-spec/WebProfile.pdf. [Datum dostopa: 28. 10. 2017].

[32] MicroProfile Team, 2018. *Eclipse MicroProfile* [online]. Dostopno na: https://microprofile.io/. [Datum dostopa: 31. 5. 2018].

[33] KumuluzEE, 2018. *KumuluzEE* [online]. Dostopno na: https://ee.kumuluz.com/. [Datum dostopa 30. 8. 2018].

[34] Heffelfinger R. D. , 2014. *Java EE 7 with GlassFish 4 Application Server.* Birmingham: Packt Publishing.

[35] Albertoni F. , Bajerski J. , Barillari D. , Cada L. , Hanson S. , Huang G. L. , Jain R. , Mendes G. K. , Mierlea C. , Narain S. , Pinto S. , Ricciuti J. , Sadtler C. , Steege C. , 2013. *WebSphere Application Server V8.5 Concepts, Planning, and Design Guide.* Armok: IBM Corporation.

[36] Albertoni F. , Baumann T. , Bhatia Y. , Fronza E. M. , Ros Gomes M. , Kapciak S. , Mierlea C. , Pinto S. , Ramachandra A. , Rui L. , Troncoso M. , 2013. *WebSphere Application Server V8.5 Administration and Configuration Guide for the Full Profile.* Armonk: IBM Corporation.

[37] Mulholland A. , 2015. *Choosing between traditional WebSphere and Liberty* [online]. Dostopno na: http://public.dhe.ibm.com/ibmdl/export/pub/software/websphere/wasdev/documentation/ChoosingTraditionalWASorLiberty-16.0.0.4.pdf. [Datum dostopa: 5. 6. 2018]

[38] IBM, 2018. *Java EE 7 programming model support* [online]. Dostopno na: https://www.ibm.com/support/knowledgecenter/SSEQTP_9.0.0/com.ibm.websphere.base.doc/ae/rovr_specs_javaee7.html. [Datum dostopa: 6. 6. 2018].

[39] Hachitani M. , Mierlea C. , Neergaard P. , Smolko G. , 2015. *IBM WebSphere Application Server Liberty Profile Guide for Developers.* Armonk: IBM Corporation.

[40] IBM, 2016. *To 16.0.0.2 and beyond: Liberty continuous, single-stream fix pack delivery* [online]. Dostopno na: https://developer.ibm.com/wasdev/blog/2016/06/24/16-0-0-2-beyond-liberty-continuous-single-stream-fix-pack-delivery/. [Datum dostopa: 4. 6. 2018]

[41] Esen A. , Iue T. , Patterson N. , Ricciuti J. , 2015. *IBM WebSphere Application Server V8.5 Administration and Configuration Guide for Liberty Profile.* Armonk: IBM Corporation.