

Uniwersytet Śląski

Wydział Informatyki i Nauki o Materialach

Vitalii Semkulych

Numer albumu:306050

Analiza metod wykorzystania języka Java w tworzeniu aplikacji  
internetowych

Praca dyplomowa  
Magisterska

Promotor:  
dr. Roman Simiński

Sosnowiec 2019

## Spis treści

1 Wstęp.....	5
2 Analiza rozpatrywanego problemu .....	7
3 Budowa aplikacji webowych .....	9
3.1 Java Enterprise Edition.....	9
3.1.1 Architektura komponentów JavaBeans.....	11
3.1.2 Komponenty WEB.....	11
3.1.3 Komponenty biznesowe.....	11
3.1.4 Poziom przechowywania danych.....	12
3.1.5 Kontener webowy .....	13
3.2 Protokoły HTTP .....	14
3.2.1 Wiadomości HTTP .....	14
3.3 Deskryptor wdrożenia.....	15
4 Rodzaje metod język Java do tworzenia aplikacji internetowych.....	17
4.1 Serwlety .....	17
4.1.1 Cykl życia serwletu.....	17
4.2 Java Servlet Page.....	20
4.2.1 Dyrektywa "JSP page" .....	20
4.2.2 Dyrektywa JSP taglib.....	22
4.2.3 Dyrektywa JSP "include" .....	22
4.2.4 Składnia deklaracji JSP .....	22
4.2.5 JSP Scriptlets .....	22
4.2.6 Wyrażenia JSP .....	23
4.2.7 Wykorzystanie JSTL w JSP.....	24
4.3 Java Servlet Facets.....	25
4.3.1 Sterowanie nawigacją .....	25
4.3.2 CDI bean .....	25
4.3.3 Podłączenia AJAX w JSF .....	26
4.4 Spring Framework .....	27
4.4.1 Architektura Spring Framework .....	28

4.4.2 Obiekty bean w Spring Framework .....	29
4.4.3 Spring Boo .....	31
4.4.4 Spring Security.....	32
4.4.5 Dialekt Thymeleaf .....	33
5 Wykorzystywane narzędzia oraz technologie .....	35
5.1 GlassFish Application Server .....	35
5.2 Serwer MySQL.....	36
5.3 NetBeans IDE .....	39
5.4 Przeglądarka Google Chrome oraz wtyczka "Performance page analyzer" .....	40
5.4 Bootstrap Framework .....	42
5.6 Technologia Ajax .....	43
5.7 Technologia Maven .....	44
6 Charakterystyka aplikacji testowych.....	47
7 Badania i eksperymenty .....	50
7.1 Cel oraz metodyka badań .....	50
7.2 Przebieg badań i omówienie wyników .....	51
7.3 Dyskusja wyników. ....	58
8. Podsumowanie i wnioski końcowe .....	60
9. Bibliografia.....	61
10. Spis rysunków i tabel. ....	62

Słowa kluczowe: Java, Servlet, JSF, JSP, Spring Framework.

### **Oświadczenie autora pracy**

Ja, niżej podpisany/a:

imię (imiona) i nazwisko: Vitalii Semkyłych

autor pracy dyplomowej pt. Zastosowanie sztucznej inteligencji w grach fabularnych

Numer albumu: 306050

Student Wydziału Informatyki i Nauki o Materiałach Uniwersytetu Śląskiego w Katowicach

kierunku studiów: Informatyka magisterska — studia stacjonarne II stopnia

specjalności: Inteligentne systemy informatyczne

Oświadczam, że ww. praca dyplomowa:

- została przygotowana przeze mnie samodzielnie <sup>1</sup>,
- nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (tekst jednolity Dz. U. z 2006 r. Nr 90, poz. 631, z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
- nie zawiera danych i informacji, które uzyskałem/łam w sposób niedozwolony,
- nie była podstawą nadania dyplomu uczelni wyższej lub tytułu zawodowego ani mnie, ani innej osobie.

Oświadczam również, że treść pracy dyplomowej zamieszczonej przeze mnie w Archiwum Prac Dyplomowych jest identyczna z treścią zawartą w wydrukowanej wersji pracy.

**Jestem świadoma/y odpowiedzialności karnej za złożenie fałszywego oświadczenia.**

.....  
Data

.....  
Podpis autora pracy

<sup>1</sup> uwzględniając merytoryczny wkład promotora (w ramach prowadzonego seminarium dyplomowego)

# 1 Wstęp

Ze względu na postęp techniczny, który się nastąpił pod koniec dwudziestego oraz na początku dwudziestego pierwszego wieku, aplikacje webowe zaczęły być szeroko stosowane w zastosowaniach osobistych oraz w zawodowych. Popularność Internetu przyczyniła się do szybkiego rozwoju technologii internetowych, zmiany koncepcyjnych i strukturalnych aspektów stron internetowych. Nastąpił podział aplikacji webowych na dwie warstwy: kliencką oraz serwerową, gdzie warstwą kliencką jest warstwa aplikacji która wysyła zapytania do serwera, a warstwa serwerowa udostępnia wybrane usługi innym programom (klientom).

Celem badawczym tej pracy jest analiza możliwości wykorzystania języka Java (wieloplatformowy, obiektowy język programowania wydany w 1995 roku przez Sun Microsystems) do tworzenia aplikacji internetowych. W ramach pracy zostanie stworzona kilka aplikacji webowych z wykorzystaniem następnych technologii: Java Servlet Pages (JSP), Java Servlet Faces (JSF) oraz Spring Framework. Eksperymenty obejmować będą przetestowanie szybkości, zużycia procesora oraz zapotrzebowania na pamięć operacyjną przy wykonywaniu różnych funkcji serwerowych w opracowanych projektach. Zostanie opisane wykorzystanie serwetów, ponieważ wszystkie wymienione powyżej technologie zbudowane w oparciu o ten mechanizm.

Wszystkie wyżej wymienione technologie zostaną opisane, przeanalizowane oraz porównane pod względem łatwości wymiany lub dołączenia modułów z których składa się aplikacja webowa takich jak bezpieczeństwo, uwierzytelnianie oraz łatwości korzystania z innych technologii takich jak na przykład AJAX. Czyli technologie zostaną porównane pod względem łatwości pisania aplikacji.

Wyjaśnione także zostaną dotyczące pojęcie budowy aplikacji webowych, pojęcia JEE (Java Enterprise Edition - Platforma Java, zapewniająca interfejs API i środowisko operacyjne do tworzenia i uruchamiania oprogramowania dla przedsiębiorstw, w tym usług webowych i sieciowych oraz innych skalowalnych aplikacji rozproszonych) oraz protokołów HTTP (HyperText Transfer Protocol - protokół transferu danych aplikacji w postaci dowolnych danych, oparty na technologii klient-serwer).

Zostaną podane oraz opisane narzędzia stosowane do tworzenia oraz testowania aplikacji webowych dołączonych z załączniku do pracy magisterskiej. Także będą przedstawione funkcjonalne możliwości serwera webowego GlassFish, dzięki któremu istnieje możliwość łatwego połączenia aplikacji webowych z innymi zasobami, z których te aplikacje mogą korzystać, na przykład bazy danych lub skrzynek pocztowych. Inną przyczyną dlaczego został wybrany GlassFish dla aplikacji napisanych w oparciu o technologie JSP i JSF jest to, że wspiera on w pełni Java Enterprise Edition, ponadto został uznany jako wzorcowy serwer aplikacji dla JEE.

W przypadku Spring Boot został użyty wbudowany serwer WWW. Dzięki któremu można tworzyć samodzielną aplikację webową, co nie tylko jest wygodne przy tworzeniu aplikacji, ale jest także akceptowalnym rozwiązaniem dla aplikacji na poziomie korporacyjnym i staje się coraz bardziej przydatne w świecie mikro usług.

## 2 Analiza rozpatrywanego problemu

W dzisiejszych czasach Internetu używamy na codzień, w związku z czym, coraz więcej powstaje nowych technologii do tworzenia aplikacji webowych. Tworzyć aplikację webową można za pomocą różnych języków programowania, ale w tej pracy skupię się na analizie dostępnych technologii dla tworzenia aplikacji webowych w jednym z najpopularniejszych w dzisiejszych czasach, języku Java.

Są następujące, obecnie znane technologie wykorzystywane do tworzenia aplikacji webowych w języku Java:

- 1) Servlet
- 2) JSP (Java Servlet Pages)
- 3) JSF (Java Servlet Faces)
- 4) Spring Framework

Wszystkie technologie webowe obecnie istniejące w języku Java są oparte o wykorzystanie Servletów. Servlet - implementacja, która rozszerza funkcjonalność serwera. On współdziała z klientami udostępniając odpowiedzi na zapytania.

Celem pracy jest w przeprowadzeniu badań nad technologiami używanymi do tworzenia web-aplikacji w języku Java oraz wyznaczenie zalet i wad każdej z nich, weryfikacja w jakich wypadkach warto używać tej lub innej technologii.

W tej pracy zostaną przeanalizowane oraz porównane następujące technologie: JSP, JSF, Spring Framework. Tworzenie aplikacji z wykorzystaniem każdej z tych technologii istotnie się różni. Każda technologia ma własne wady i zalety, zarówno w sensie funkcjonalnym jak i względem szybkości działania lub obciążenia komputera przez aplikacje napisaną w jednej z powyżej wymienionych technologii.

JSP (Java Servlet Pages) jest najstarszą z badanych technologii. Jej wykorzystanie polega na wstawianiu funkcjonalnych części z kodem Java w kod HTML. Ponieważ nie ma pełnej rozdzielności pomiędzy klientem i serwerem, powstają często problemy w przypadku konieczności wprowadzenia zmian do wcześniej napisanego dużego projektu. W JSP nie ma automatycznego przechowywania stanu funkcjonalnych elementów strony w przypadku jej aktualizacji. Na przykład, możemy przedstawić, że mamy formę do wyszukiwania produktów w sklepie internetowym napisanym w JSP, który składa się z pola dla wprowadzenia nazwy szukanego towaru oraz przycisku szukaj. Po wprowadzeniu nazwy towaru w pole i naciśnięciu przyciska zostanie zaktualizowana strona po czym informacja o nazwie szukanego towaru w polu zostanie stracona. Żeby naprawić to trzeba ręcznie przechowywać wszystkie dane w atrybutach sesji oraz wczytywać je w odpowiedni sposób po ładowaniu strony. Ten problem został uwzględniony w innej

technologiach takich jak: Spring, JSF. Dużą zaletą JSP jest to, że aplikacje napisane z jej użyciem działają dość szybko.

W JSF (Java Servlet Facets) Framework warstwa serwerowa oraz warstwa kliencka jest całkiem rozdzielona, co jest dużą zaletą tej technologii w porównaniu do JSP. Część kliencka była oparta na XHTML do wersji 2.3, co było jedną z największych wad. Część serwerową tworzy się za pomocą CDI beanów oraz ich właściwości zwanej wstrzykiwaniem zależności (wstrzykiwanie jednego beana w drugi). Projekty JSF są dość łatwo konfigurowane, także jest możliwość łatwego podłączenia AJAX dzięki załącznikowi "<f:ajax>", co jest dużą zaletą tej technologii w porównaniu do innych badanych.

Jeszcze jedną analizowaną oraz badaną technologią w tej pracy jest Spring. Wielką zaletą, w porównaniu do poprzednich technologii jest wykorzystywanie aspektów. Używanie aspektów nadaje możliwość łatwego dodawania dodatkowych funkcjonalnych możliwości takich jak zabezpieczenia, buforowanie, logowanie w punktach połączenia (miejsce gdzie należy zastosować aspekt). Także jak w JSF, w Spring Framework aplikacje budowane są z wykorzystaniem beanów, które są odpowiednio opisane dzięki klasom konfiguracji, adnotacjom albo konfiguracji XML. Także trzeba dodać, że w porównaniu do JSF w Spring Framework nie ma wbudowanych opcji do wykorzystania AJAX, co zmusza dewelopera do napisania wiele dodatkowego kodu w przypadku konieczności użycia AJAX w projekcie Spring. Tworzyć aplikacje na Spring Framework można wykorzystując wbudowany kontener webowy, dzięki któremu deweloperzy nie muszą się już martwić konfiguracją kontenera Serwletu i wdrożeniem na nim aplikacji. Teraz aplikację można uruchomić przez plik wykonywalny jar przy użyciu wbudowanego serwera.



### 3 Budowa aplikacji webowych

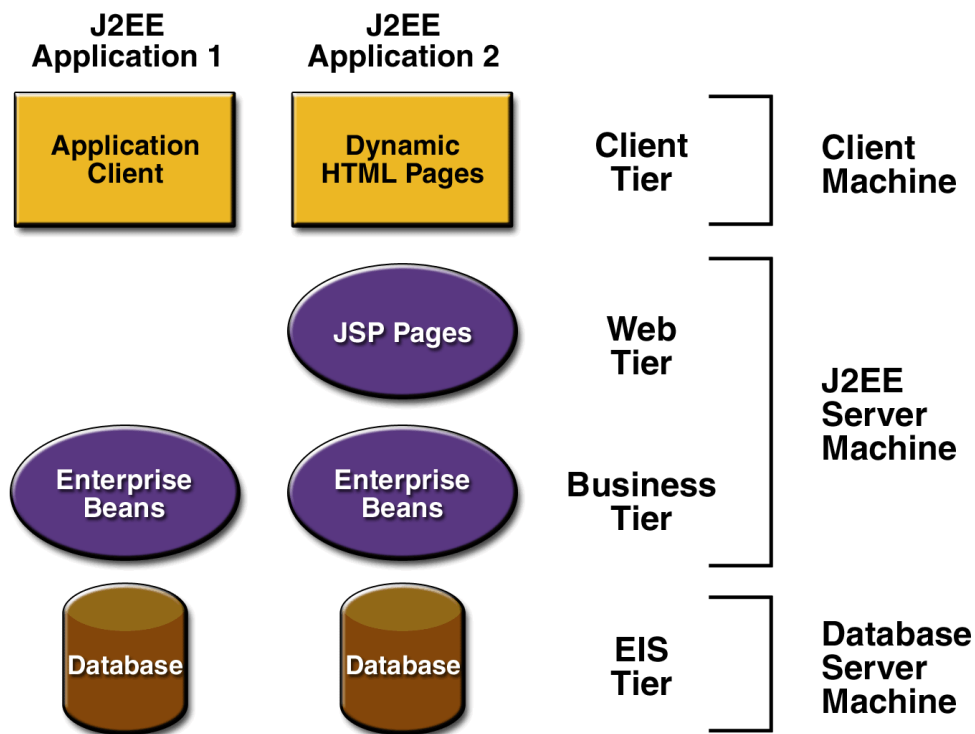
Dzisiaj programiści dążą do tworzenia rozproszonych transakcyjnych aplikacji biznesowych i chcą korzystać z szybkości, bezpieczeństwa i niezawodności zapewnianej przez technologie serwerowe.

Aby zmniejszyć koszty i przyspieszyć projektowanie i rozwój aplikacji korporacyjnych, platforma JEE oferuje podejście komponentowe do projektowania, rozwoju, integracji i wdrażania aplikacji korporacyjnych. Platforma JEE oferuje wielopoziomowy rozproszony model aplikacji, możliwość ponownego wykorzystania komponentów, zintegrowaną wymianę danych w oparciu o XML, ujednolicony model bezpieczeństwa i elastyczne zarządzanie transakcjami. Producenci i użytkownicy mają swobodę wyboru produktów i komponentów, które najlepiej spełniają ich wymagania biznesowe i technologiczne.

#### 3.1 Java Enterprise Edition.

Platforma JEE używa wielopoziomowy rozproszony model aplikacji. Aplikacja jest podzielona logicznie na komponenty względem funkcjonalności, które one reprezentują. Komponenty z których składa się JEE aplikacja instalowane na różnych komputerach zgodnie z poziomem do którego należą one w wielowarstwowym środowisku JEE. Części aplikacji JEE przedstawione na rysunku 1:

- Komponenty na poziomie klienta działają na komputerze klienta.
- Komponenty na poziomie warstwy sieci Web działają na serwerze JEE.
- Komponenty warstwy biznesowej działają na serwerze JEE.
- Oprogramowanie systemu informacyjnego na poziomie korporacyjnym (EIS) działa na serwerze EIS.



**Rysunek 1: Aplikacje wielowarstwowe.**

Aplikacje JEE składają się z komponentów (kompletny funkcjonalny moduł oprogramowania osadzony w aplikacji JEE z powiązanymi klasami i plikami). W Java Enterprise Edition zdefiniowane następujące komponenty:

- Aplikacje klienckie i aplety są komponentami działającymi na komputerze klienckim.
- Komponenty technologii Java Servlet i JavaServer Pages (JSP) - składniki sieci działające na serwerze.
- Komponenty Enterprise - komponenty biznesowe działające na serwerze.

Klientem JEE może być klient sieciowy lub klient aplikacji [1]:

- Klienci sieciowi - składa się z dynamicznych stron internetowych (na przykład strony HTML, XML) oraz generowanych komponentów webowych na poziomie sieci. Zazwyczaj nie wykonują takich funkcji, jak wysyłanie zapytań do bazy danych, realizacją złożonych reguł biznesowych lub komunikowanie się z aplikacjami serwerowymi. Podczas korzystania z klienta webowego takie operacje są przenoszone na komponenty biznesowe działające na serwerze JEE i wykorzystujące bezpieczeństwo, szybkość, usługi i niezawodność technologii serwerowych JEE.
- Aplety - to małe aplikacje klienckie napisana w języku Java i działająca na wirtualnej maszynie Java zainstalowanej w przeglądarce internetowej. Jednak

systemy klienckie mogą wymagać wtyczki Java, aby aplet mógł pomyślnie działać w przeglądarce internetowej.

- c) Klienci aplikacji webowych - działają na komputerze klienckim i zapewniają użytkownikom możliwość pracy z zadaniami, które wymagają bogatszego interfejsu użytkownika od interfejsu dostarczanego przez języki znaczników. Zapełniają one graficzny interfejs użytkownika utworzony przy użyciu JavaFX, Swing, AWT API. Klienci aplikacji webowych mają bezpośredni dostęp do komponentów korporacyjnych działających na poziomie biznesowym. Jednak klient aplikacji JEE może otworzyć połączenie HTTP do komunikacji z serwletem działającym na poziomie sieci, jeśli takie wymagania aplikacji istnieją.

### **3.1.1 Architektura komponentów JavaBeans**

Poziomy serwerów i klientów także zawierają komponenty oparte na architekturze komponentów JavaBeans. Są one używane do kontrolowania przepływu danych między klientem aplikacji lub apletem, a komponentami działającymi na serwerze JEE lub komponentami serwera i bazą danych.

Komponenty JavaBeans zawierają zmienne obiektowe i metody pobierające (ang. getObject) i ustawiające (ang. setObject). Komponenty JavaBeans używane w ten sposób są zazwyczaj proste w projektowaniu i implementacji, ale muszą być zgodne z regułami nazewnictwa i projektowania zdefiniowanymi w architekturze komponentów JavaBeans.

### **3.1.2 Komponenty WEB**

Składnikami JEE Web mogą być serwlety lub strony JSP. Serwlety są klasami języka Java, które po otrzymaniu żądania od klienta zarządzają nimi i budują odpowiedzi. Strony JSP to dokumenty tekstowe, które wykonują się w taki sam sposób jak serwlety, ale oferują bardziej naturalne podejście do tworzenia statycznej zawartości.

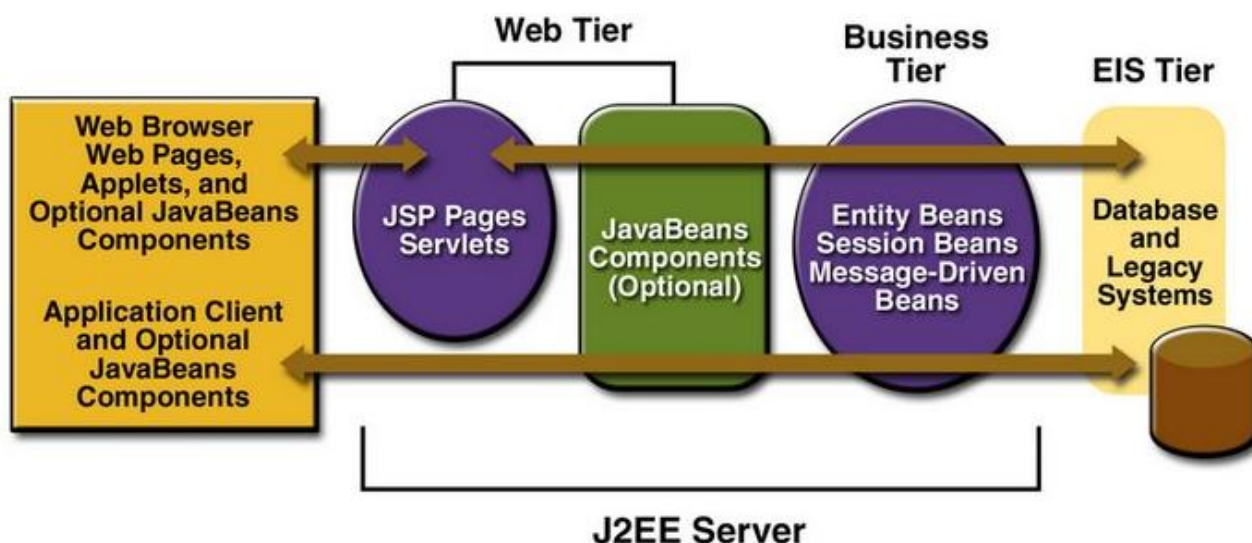
Podobnie jak warstwa klienta, warstwa sieci webowej pokazana na rysunku 2 może zawierać komponent JavaBeans do zarządzania wprowadzaniem danych przez użytkowników i kierowania tych danych do komponentu działającego w warstwie biznesowej w celu odpowiedniego przetwarzania.

### **3.1.3 Komponenty biznesowe**

Kod biznesowy jest logiką rozwiązywanego problemu bezpośrednio w obszarze biznesowym (na przykład bank, finanse). Komponenty biznesowe są zarządzane przez składniki korporacyjne działające na poziomie biznesowym. Rysunek 2 pokazuje, jak komponent biznesowy odbiera dane z programu klienckiego, przetwarza je (w razie potrzeby) i wysyła do korporacyjnego

systemu informacyjnego w celu przechowywania. Komponent biznesowe pobiera dane z repozytorium i/lub procesów i wysyła je z powrotem do programu klienta.

Istnieją trzy typy komponentów biznesowych: komponenty sesji, komponenty zarządzania danymi i komponenty sterowane komunikatami. Komponenty sesji reprezentują krótkoterminową komunikację z klientem. Po zakończeniu pracy komponent sesji i jego dane znikają. Natomiast komponenty zarządzania danymi reprezentują trwałe dane przechowywane w pojedynczym wierszu tabeli bazy danych. Jeśli klient się wyłącza lub serwer się wyłącza, wbudowana usługa zapewnia, że dane takiego komponentu zostaną zapisane. Komponenty sterowane komunikatami łączą funkcje komponentów sesji i odbiornika komunikatów JMS (Java Message Service), umożliwiając komponentowi biznesowemu asynchroniczne odbieranie komunikatów JMS [1].



**Rysunek 2: Komunikacja z serwerem na różnych poziomach aplikacji webowej JEE**

### 3.1.4 Poziom przechowywania danych

Powiązanie warstwy "DataSource" i "Persistence Layer" zostało zaprojektowane dla pracy z danymi. Ponadto dane mogą być przechowywane nie tylko w bazie danych. Może to być plik XML, plik tekstowy, serwer pocztowy lub plik Excel - może to być wszystko, o ile jest to mniej lub bardziej stały repozytorium.

Aby móc pobierać dane z konkretnego źródła, jest używane pojęcie "DataSource" - źródło danych, za którym zwykle ukrywa się coś mniej lub bardziej konkretnego - baza danych, plik lub coś innego. Koncepcja "DataSource" działa jak rodzaj pomostu między "Persistence Layer", a na przykład bazą danych. W rzeczywistości istnieje nawet specjalna klasa - "javax.sql.DataSource", która służy do tworzenia połączenia z bazą danych. Ale oprócz tej abstrakcji nadal istnieje bardzo ważny wzorzec projektowy - Data Access Object (DAO), co wynika z kilku powodów:

- Źródła danych mogą się różnić.
- Różne API mogą być używane do bezpośredniego dostępu do określonego repozytorium.

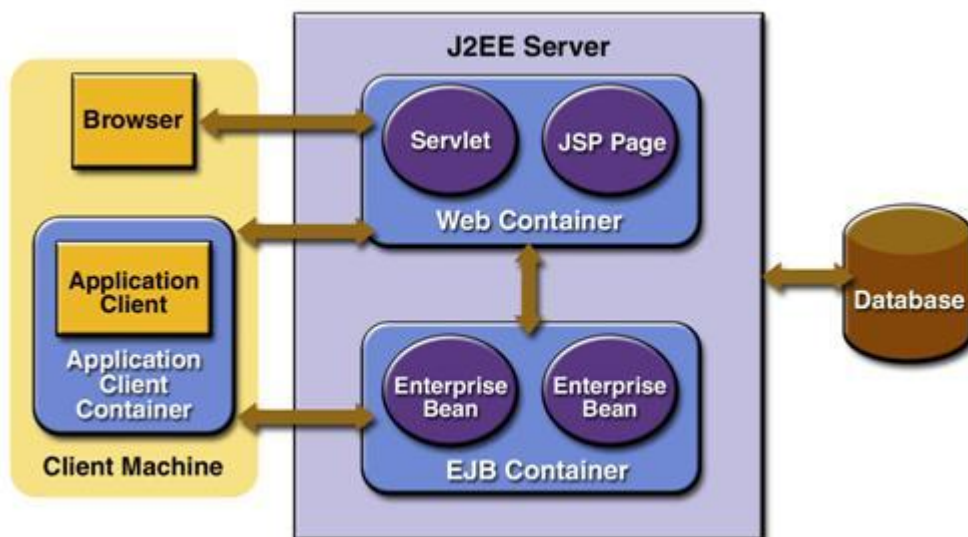
- Włączenie kodu do logiki ograniczającej zmianę pamięci nie jest dobrym pomysłem.

### 3.1.5 Kontener webowy

Kontener webowy jest odpowiedzialny za zarządzanie cyklem życia serwletów, mapowanie adresu URL (Uniform Resource Locator) na konkretny serwlet i zapewnienie, że przesyłający żądanie URL ma odpowiednie prawa dostępu. Kontener webowy obsługuje żądania do serwletów, plików stron serwera java (JSP) i innych typów plików, które zawierają kod po stronie serwera. Kontener WWW tworzy instancje serwletów, ładuje i rozładowuje Serwlety, tworzy i zarządza obiektami żądań i odpowiedzi oraz wykonuje inne zadania zarządzania Serwletami [1].

Proces wdrażania ustawia komponenty aplikacji JEE w kontenerach JEE, jak pokazano na rysunku 3.

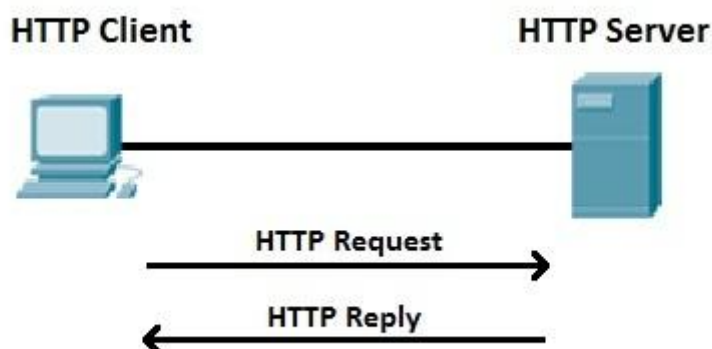
- - Serwer JEE - jest częścią środowiska wykonawczego aplikacji JEE. Serwer JEE zapewnia EJB i kontenery internetowe.
- Komponenty JEE serwera:
- - Kontener EJB zarządza wykonaniem komponentów korporacyjnych dla aplikacji JEE. Komponenty Enterprise działają na serwerze JEE.
- - Kontener sieciowy - kontroluje wykonanie strony JSP i serwletów dla aplikacji JEE. Składniki sieci webowej działają na serwerze JEE.
- - Kontener aplikacji klienta - zarządza wykonaniem komponentów aplikacji klienta (aplikacje klienckie wykonywane na kliencie).
- - Aplety kontenerowe - zarządzają wykonywaniem apletów (składa się z przeglądarki internetowej i wtyczki Java uruchomionej na kliencie).



Rysunek 3: Serwer JEE i kontenery

## 3.2 Protokoły HTTP

HTTP - to protokół, który umożliwia otrzymywanie różnych zasobów, takich jak dokumenty HTML. HTTP jest podstawą wymiany danych w Internecie. HTTP jest protokołem interakcji klient-serwer (rysunek 4), co oznacza inicjowanie żądań do serwera przez odbiorcę, zwykle przeglądarkę internetową. Wynikowy dokument będzie składał się z różnych pod dokumentów, które są częścią ostatecznego dokumentu, na przykład z oddzielnie uzyskanego tekstu, opisu struktury dokumentu, obrazów, plików wideo, skryptów i wielu innych [10].



**Rysunek 4: Przykład wymiany danych w Internecie za pomoc HTTP protokołu**

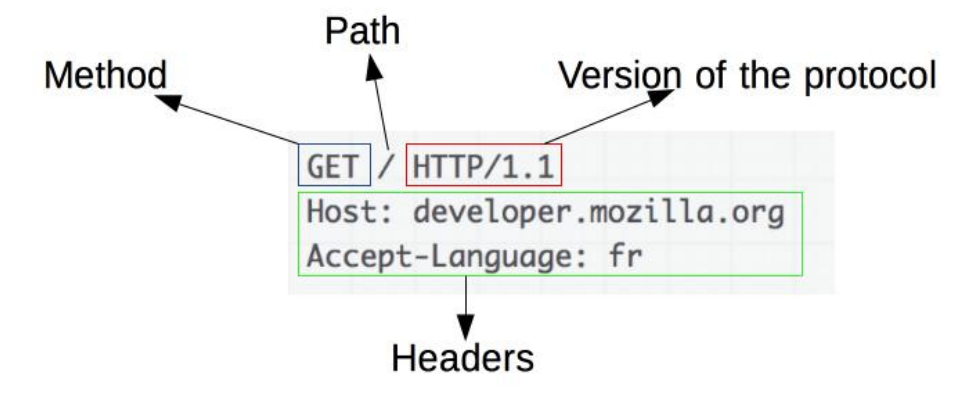
Klienci i serwery współdziałają, wymieniając pojedyncze wiadomości (a nie strumień danych). Wiadomości wysyłane przez klienta są zwykle określane przez przeglądarkę jako żądania, a wiadomości wysyłane przez serwer są nazywane odpowiedziami. Ze względu na swoją rozszerzalność jest on używany nie tylko do odbierania dokumentów hipertekstowych, obrazów i wideo przez klienta, ale także do przesyłania treści na serwery, na przykład za pomocą formularzy HTML. HTTP może być również używany do pobierania tylko części dokumentu w celu aktualizacji strony internetowej na żądanie (na przykład za pośrednictwem żądania AJAX).

### 3.2.1 Wiadomości HTTP.

Istnieją dwa typy komunikatów HTTP: żądania i odpowiedzi.

Żądania określone przez następujące elementy (rysunek 5):

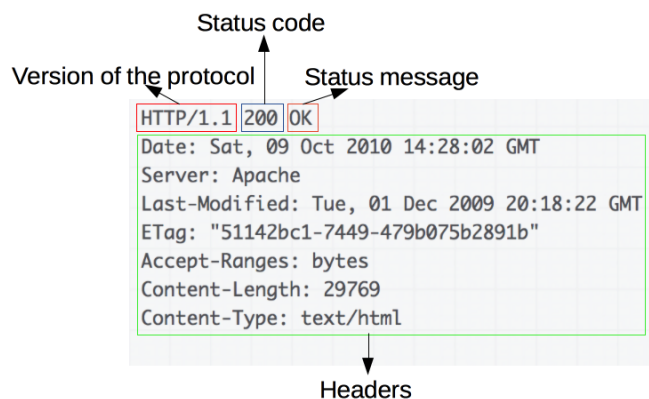
- Metoda HTTP GET, POST. GET używa się dla uzyskania zasobów klientem, POST dla przesyłania formularza HTML.
- Ścieżka do zasobu: URL są pozbawione elementów, które są oczywiste z kontekstu.
- Wersja protokołu HTTP.
- Nagłówki (opcjonalne), które dostarczają dodatkowych informacji dla serwera.
- Ciało, dla niektórych metod, takich jak POST, który zawiera wysłany zasób.



**Rysunek 5: Przykład HTTP żądania.**

Odpowiedzi zawierają następujące elementy(rysunek 6):

- Wersja protokołu HTTP.
- Kod statusu HTTP wskazujący powód sukcesu lub niepowodzenia.
- Komunikat o statusie - krótki opis kodu statusu.
- Nagłówki HTTP, jak nagłówki w żądaniach.



**Rysunek 6: Przykład odpowiedzi HTTP**

### 3.3 Deskryptor wdrożenia

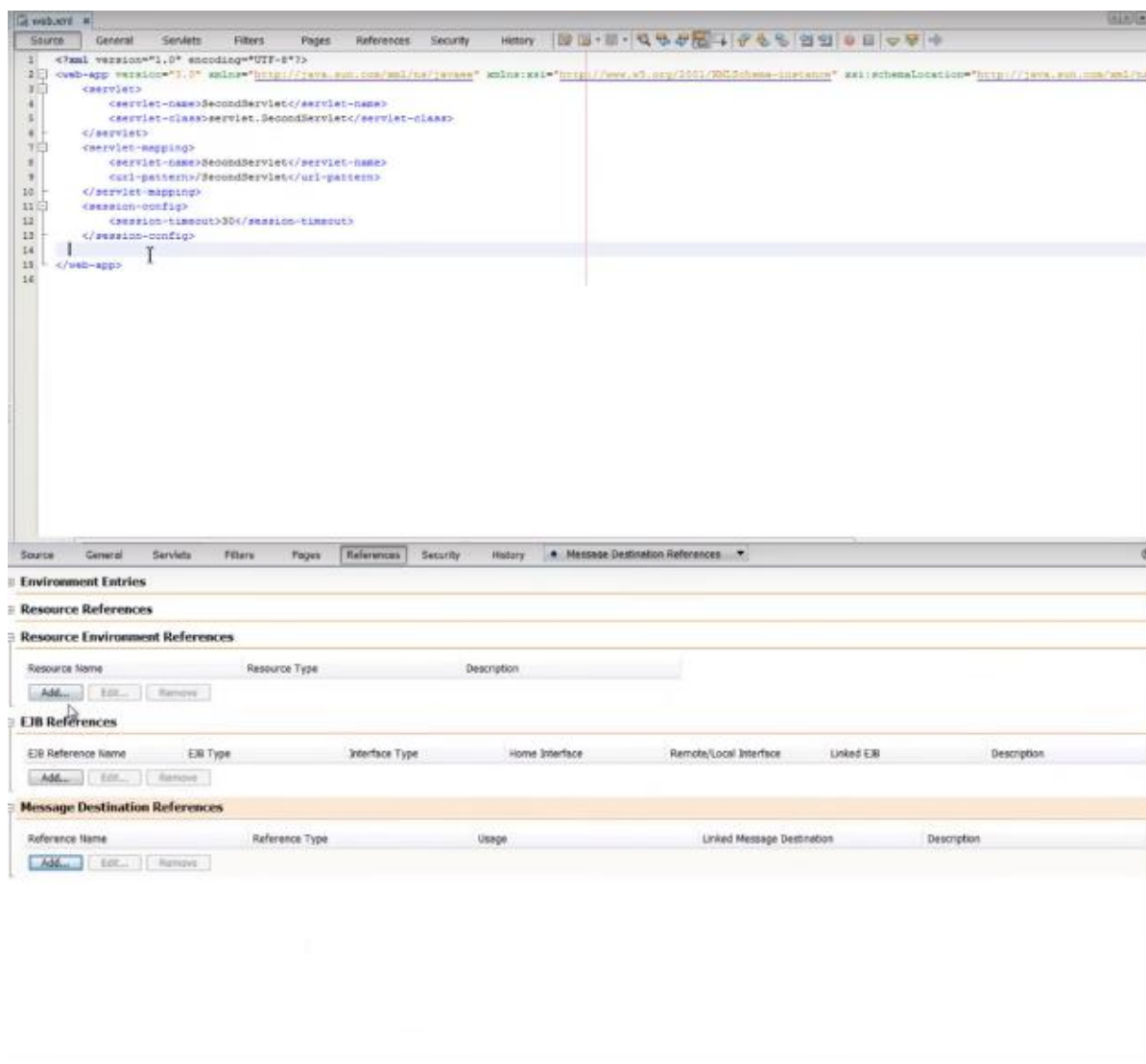
Deskryptor wdrożenia (ang. Deployment descriptor) jest plikiem konfiguracyjnym artefaktu, który może być wdrożony w kontenerze Servlet. W specyfikacji Java Platform Enterprise Edition deskryptor wdrażania opisuje sposób wdrożenia komponentu, modułu lub aplikacji (takiej jak aplikacja internetowa lub aplikacja korporacyjna).

Ten plik konfiguracyjny określa opcje wdrożenia modułu lub aplikacji z określonymi ustawieniami zabezpieczeń i opisuje określone wymagania konfiguracyjne. Deskryptor wdrożenia opisywany przez składnie XML(rysunek 7).

W przypadku aplikacji internetowych deskryptor wdrożenia powinien mieć nazwę "web.xml" i znajdować się w katalogu "WEB-INF"( katalogu głównym aplikacji internetowej).



Ten plik jest standardowym deskryptorem wdrożenia zdefiniowanym w specyfikacji Servlet. Istnieją również inne typy deskryptorów, takie jak plik deskryptora wdrożenia "sun-web.xml", który zawiera dane wdrożenia specyficzne dla serwera Sun GlassFish Enterprise Server.



**Rysunek 7: Przykład deskryptora wdrożenia.**

Za pomocą deskryptora wdrożenia można:

- ustawić czas sesji dla pojedynczego użytkownika,
- stronę powitalną dla projektu, zdefiniować role,
- które użytkownicy będą mieli w aplikacji,
- zadeklarować linki na resursy zewnętrzne,
- dodać strony błędów dla różnych rodzajów błędów,
- dodać nowe filtry do łańcuchu filtrów, co rozszerza funkcjonalne możliwości przy tworzeniu żądań oraz odpowiedzi.



## 4 Rodzaje metod język Java do tworzenia aplikacji internetowych

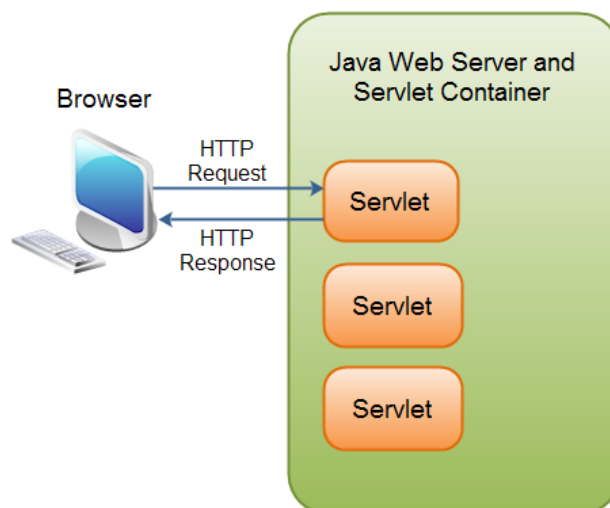
W dzisiejszych czasach są dojrzałe technologie za pomocą których można tworzyć aplikacje webowe w oparciu o język Java, takich jak: JSP( Java Servlet Pages), JSF (Java Servlet Faces), Spring. Każda z tych technologii jest zbudowana w oparciu na Servlet, co jest interfejsem Java, którego implementacja rozszerza funkcjonalność serwera. Servlet współdziała z klientami za pomocą zasady żądania-odpowiedzi.

W niniejszym rozdziale zostanie opisano większość technologii służących do tworzenia aplikacji webowych opartych o język Java.

### 4.1 Serwlety

Servlet to Java program, który działa po stronie serwera aplikacji internetowej. Podobnie jak applety dynamicznie rozszerzają funkcjonalność przeglądarki internetowej, Servlety dynamicznie rozszerzają funkcjonalność serwera webowego [3].

Gdy żądanie jest wysyłane przez klienta, serwer sieciowy może określić, który applet musi zostać wykonany przy użyciu specjalnego pliku konfiguracyjnego (Deployment descriptor). Następnie serwer sieciowy uruchamia maszynę "Java Virtual Machine", która z kolei uruchamia servlet, który przetwarza HTTP żądanie i wysyła zawartość do kontenera serwera sieciowego (prawdopodobnie w formie strony HTML). Serwer wysyła klientowi HTTP odpowiedź (stronę HTML wygenerowaną przez servlet) (rysunek 8)



**Rysunek 8: Serwlety wewnątrz kontenera Java Servlet.**

#### 4.1.1 Cykl życia serwletu

Servlety wykonują się na platformie serwera sieciowego (na przykład: GlassFish, TomCat), jako część tego samego procesu, co sam serwer sieciowy. Serwer sieciowy jest odpowiedzialny

za inicjowanie, wywoływanie i zniszczenie każdej instancji serwletu. Serwer sieciowy komunikuje się z serwletem za pomocą prostego interfejsu - "javax.servlet.Servlet".

Interfejs "javax.servlet.Servlet" zawiera trzy główne metody:

- Po pierwszym załadowaniu serwletu wywoływana jest metoda "init()". Pozwala to serwletowi na wykonywanie wszelkich prac instalacyjnych, takich jak otwieranie plików lub nawiązywanie połączeń z ich serwerami. Jeśli serwlet jest na stałe zainstalowany na serwerze, wtedy jest on ładowany podczas uruchamiania serwera. Gwarantuje się, że metoda "init()" zakończy się przed jakimkolwiek innym wywołaniem serwletu (na przykład wywołanie metody "service()"). Metod "init()" wywołuje się tylko raz dopóki serwlet nie zostanie rozładowany, a następnie ponownie załadowany przez serwer.
- Metoda "service()" jest sercem serwletu. Każde żądanie klienta powoduje jedno wywołanie metody "service()". Ta metoda odczytuje żądanie i tworzy komunikat odpowiedzi z dwoma argumentami, "ServletRequest" i "ServletResponse". Obiekt "ServletRequest" zawiera dane wysyłane klientem. Dane składają się z par nazwa/wartość i strumienia wejściowego. Obiekt "ServletResponse" zawiera odpowiedź serwletu wysyłanego klientowi. Metoda "service()" tworzy odpowiedź na każde żądanie klienta wysłane do niego z serwera. Należy jednak pamiętać, że może być kilka równoległych zapytań, które są przetwarzane w tym samym czasie. Jeśli metoda "service()" wymaga jakichkolwiek zasobów zewnętrznych, takich jak pliki, bazy danych, konieczne jest zapewnienie, że dostęp do zasobów jest bezpieczny dla wątków.
- Metoda "destroy()" jest wywoływana, żeby zwolnić wszystkie zasoby przed usunięciem serwletu. Ta metoda może być pusta, jeśli nie ma potrzeby wykonywania żadnych operacji końcowych. Przed wywołaniem metody "destroy()" serwer czeka na zakończenie wszystkich operacji obsługi lub wygaszanie określonego czasu.

Także serwlet zawiera dwie dodatkowe metody:

- Metoda "getServletConfig()" zwraca odwołanie do obiektu, który implementuje interfejs "ServletConfig". Ten obiekt zapewnia dostęp do informacji o konfiguracji serwletu.
- Metoda "getServletInfo()" jest definiowana przez programistę tworzącego Serwlet w celu zwrócenia ciągu zawierającego informacje o serwlecie, na przykład nazwę autora i wersję serwletu.

W serwlecie są również zdefiniowane metody "doGet()" oraz "doPost()" wykorzystywane dla przechwytywania żądań "HTTP GET" oraz "HTTP POST". Obie te metody wywołują metodę "processRequest()", która przyjmuje parametry "HttpRequest" i "HttpResponse".

"ServletRequest" dostarcza informacje o klientach dotyczące parametrów żądania HTTP (na przykład: atrybuty, strumień wejściowy) do serwletu. Metody żądanie "HttpRequest", przedstawione w poniższej tabeli 1.

**Tabela 1: Główne metody obiektu HttpRequest**

getAttribute ()	Zwraca wartość określonego atrybutu tego żądania.
getLength ()	Rozmiar żądania, jeśli jest znany.
getContentType ()	Zwraca typ MIME treści żądania.
getInputStream ()	Zwraca strumień wejściowy do odczytu danych binarnych z treści żądania.
getParameterNames ()	Zwraca tablicę ciągów z nazwami wszystkich parametrów.
getParameterValues ()	Zwraca tablicę wartości dla określonego parametru.
getProtocol ()	Zwraca protokół i wersję zapytania jako ciąg: <protocol>/<major version>.<minor version>.
getReader ()	Zwraca BufferedReader, aby pobrać tekst z treści żądania.
getRealPath ()	Zwraca rzeczywistą ścieżkę dla określonej ścieżki wirtualnej.
getRemoteAddr ()	Zwraca adres IP klienta, który wysłał żądanie.
getRemoteHost ()	Nazwa hosta komputera klienta, który wysłał to żądanie.
getScheme ()	Zwraca schemat użyty w adresie URL tego żądania (na przykład https, http, ftp itp.).
getServerName ()	Nazwa hosta serwera, który odebrał żądanie.
getServerPort ()	Zwraca numer portu użyty do odebrania tego żądania.
getCookies()	Zwraca listę plików cookie wysłanych przez klienta w tym żądaniu
getSession()	Zwraca obiekt sesji użytkownika

Interfejs "ServletResponse" to narzędzie do wysyłania danych do klienta. Metody odpowiedzi "HttpResponse" przedstawione w poniższej tabeli 2.

**Tabela 2: Główne metody obiektu HttpResponse**

getCharacterEncoding()	Возвращает MIME тип кодировки (к примеру - UTF8), в которой будет выдаваться информация.
setLocale()/getLocale()	Ustawienie/zwracanie języka używanego w dokumencie.
getOutputStream()	Zwraca strumień danych wyjściowych dla servletu.
getWriter()	Automatycznie konwertuje łańcuchy znaków na zestaw znaków określony w getCharacterEncoding () i getLocale ().
setContentLength()	Ustawienie wartości pola HTTP nagłówka „Content-Length”.
setContentType()	Służy do wysyłania typu zawartości MIME dokumentu.
isCommitted()	Sprawdza czy wysyłanie danych do klienta już się rozpoczęło.
reset()	Jeśli nagłówek HTTP nie został jeszcze wysłany, metoda resetowania „resetuje” nagłówek HTTP do wartości „domyślnych”.
addCookie()	Dołączanie obiektu Cookie do odpowiedzi.

## 4.2 Java Servlet Page.

Technologia Sun Servlets została opracowana przez Sun Microsystems - stosuje platformę Java do realizacji CGI i rozszerzeń API serwera. Technologia rozwiązuje problem wydajności, wykonując wszystkie żądania jako wątki w jednym procesie. Serwlety są niezależne od platformy, ponieważ działają w wirtualnej maszynie Java (JVM) i mogą łatwo udostępniać zasoby [2]. Technologia ma wiele funkcji, a duża liczba bibliotek zapewnia najbardziej różnorodne narzędzia potrzebne w rozwoju.

Strony JSP mają składnię łączoną wyrażoną przez łączenie standardowej składni zgodnej z specyfikacją HTML i składni JSP zdefiniowanej przez specyfikację Java Server Pages. Składnia JSP określa zasady pisania stron JSP składających się ze standardowych znaczników HTML i znaczników JSP [2].

Strony JSP, oprócz znaczników HTML, zawierają znaczniki JSP następujących kategorii:

- dyrektywy (dyrektywy): "page", "taglib", "include";
- deklaracje;
- skrypty;
- wyrażenia;
- komentarze (komentarze).

Dyrektywy JSP dostarczają globalnych informacji dotyczących konkretnych żądań wysyłanych do serwera i dostarczają informacji potrzebnych na etapie emisji. Dyrektywy są zawsze umieszczane na początku strony JSP przed wszystkimi innymi znacznikami, tak że analizator (parser) JSP wybiera globalne instrukcje podczas analizowania tekstu na samym początku. Składnia dyrektyw JSP jest następująca:

**<% @ dyrektywa Nazwa atrybutu = "wartość"%>**

### 4.2.1 Dyrektywa "JSP page"

Dyrektywa JSP 'page' definiuje właściwości strony JSP, które mają wpływ na translator. Kolejność atrybutów w dyrektywie 'page' jest nieistotna.

Przykład dyrektywy dotyczącej strony JSP (zawiera informacje o stronie):

**<% @ page info = "JSP Page"%>**

Lista możliwych atrybutów dyrektywy "page" jest przedstawiona poniżej:

- language - definiuje język używany w skryptletach plików JSP, wyrażeniach lub dowolnych dołączonych plikach. Wartością domyślną jest „java”.
- extends - określa superklasę generowanego serwletu. Ten atrybut powinien być używany z wielką ostrożnością, ponieważ możliwe jest, że serwer już używa jakiejś superklasy.

- import - definiowanie importowanych pakietów, na przykład: `%@ page import="java.util.* %`.
- session - wartości atrybuta może być wartość logiczna (true lub false). Wartość "true" (domyślnie) wskazuje, że predefiniowana zmienna sesyjna (typ HttpSession) musi być powiązana z istniejącą sesją, jeśli istnieje, w przeciwnym razie zostanie utworzona nowa sesja, do której zostanie nawiązane powiązanie. Wartość "false" wskazuje, że sesje nie będą używane, a próby uzyskania dostępu do sesji spowodują błąd podczas tłumaczenia strony JSP na serwlet.
- buffer - wartości atrybuta może przyjmować znaczenie "none" lub rozmiar bufora w kilobajtach. Ustawia rozmiar bufora dla JspWriter. Wartość domyślna zależy od ustawień serwera i nie powinna przekraczać 8 kilobajtów.
- autoFlush - wartości atrybuta może być wartość logiczna (true lub false). Określa czy bufor powinien zostać zwolniony automatycznie, gdy jest pełny, czy wystąpił błąd. Domyślnie ustawiony jako "true".
- isThreadSafe - wartości atrybuta może być wartość logiczna (true lub false). Wartość "true" (domyślnie) ustawia normalny tryb wykonywania serwletu, gdy wiele żądań jest przetwarzanych jednocześnie przy użyciu pojedynczej instancji serwletu, w oparciu o założenie, że autor zsynchronizował dostęp do zmiennych tej instancji. Wartość false wskazuje, że serwlet musi odziedziczyć SingleThreadModel (model jednowątkowy), w którym sekwencyjne lub jednoczesne żądania są przetwarzane przez oddzielne instancje serwletu.
- info - ciąg informacyjny o stronie JSP.
- errorPage - wartość atrybutu to URL strony, która powinna być wyświetlana w przypadku możliwych błędów.
- isErrorPage - wartości atrybuta może być wartość logiczna (true lub false). Sygnalizuje, czy tę stronę można wykorzystać do obsługi błędów innych stron JSP. Domyślnie wartość - "false".
- contentType - definiuje kodowanie strony JSP i odpowiedzi, a także typ "MIME" odpowiedzi JSP. Domyślny typ zawartości to text / html, kodowanie to ISO-8859-1. Na przykład: `contentType="text/html; charset=ISO-8859-1"`.
- pageEncoding - Określa kodowanie znaków na stronie JSP. Domyślnie jest to zestaw znaków z atrybutu contentType, jeśli jest tam zdefiniowany. Jeśli wartość zestawu znaków w atrybucie contentType nie jest zdefiniowana, wartość pageEncoding jest ustawiona na ISO-8859-1.

### 4.2.2 Dyrektywa JSP taglib

Dyrektywę "JSP taglib" używany do podłączenia biblioteki znaczników do strony JSP. Zawiera dwa atrybuty: "URI" - identyfikator podłączonej biblioteki, "prefix" - definicja nazwy biblioteki używana dla pracy na stronie JSP. Dyrektywa "taglib" ma następującą składnię:

```
<% @ taglib uri = "URI dołączonej biblioteki załączników" prefix = "nazwa prefiksu"%>
```

### 4.2.3 Dyrektywa JSP "include"

Dyrektywa dołączania JSP umożliwia wstawianie tekstu lub kodu podczas translacji strony JSP na serwlet. Składnia dyrektywy "include" jest następująca:

```
<% @ include file = "Względny identyfikator URI dołączonej strony"%>
```

Dyrektywa "include" ma jeden atrybut - "file" służący do wczytywania tekstu(kodu) do określonego zasobu w pliku JSP. Określony identyfikator URI jest zwykle interpretowany w odniesieniu do strony JSP, na której znajduje się łącze, ale podobnie jak w przypadku innych względnych identyfikatorów URI, można ustawić system tak, aby pozycjonował interesujący zasób w stosunku do katalogu domowego serwera webowego, dodając symbol „/” na początku identyfikatora URI.

### 4.2.4 Składnia deklaracji JSP

Dyrektywa deklaracji JSP określa deklarację zmiennych i metod w języku skryptowym, który jest dalej używany na stronie JSP. Na przykład:

```
<%! int a = 7;%>
```

Deklaracje są czasami używane do dodawania dodatkowej funkcjonalności podczas pracy z dynamicznymi danymi pochodzącymi z właściwości komponentów "JavaBeans". Przykłady deklaracji są przedstawione w tabeli 4.

**Tabela 3: Przykład deklaracji oraz inicjalizacji atrybutów i metod.**

<code>&lt;%! int i ; %&gt;</code>	Deklaracja globalnej zmiennej całkowitej 'i'.
<code>&lt;%! String hi = "Hello, World!"; %&gt;</code>	Deklaracja i inicjalizacja globalnej zmiennej łańcuchowej 'hi'.
<code>&lt;%! int k = 30; %&gt;</code>	Deklaracja i inicjalizacja globalnej zmiennej całkowitej 'k'.
<code>&lt;%! public int sub (int a, int b){return a - b}; %&gt;</code>	Deklarowanie podrzędnej metody "sub" odejmowania globalnych dwóch wartości całkowitych dla całej strony JSP.

### 4.2.5 JSP Scriptlets

Scriptlety zawierają różne fragmenty kodu napisane w języku zdefiniowanym w atrybucie "language" dyrektywy "page". Fragmenty kodu muszą być zgodne ze składnią języka skryptletowego (rysunek 9). Skryplety mają następującą składnię: `<%tekst scriptletu%>`

```

<%
    for (Book book : setBook) {

%>

<div class="singleBook">
    <div class="bookInformation">
        <p class="bookName"> <%=book.getBookName() %></p>
        
        <br><strong>ISBN:</strong> <%=book.getIsbn() %>
        <br><strong>Publisher:</strong> <%=book.getPublisher().getPublisherName() %>
        <br><strong>Number of Pages:</strong> <%=book.getPageNumber() %>
        <br><strong>Publish Date:</strong> <%=book.getPublishYear() %>
        <br><strong>Author:</strong> <%=book.getAuthor().getName() %>
        <div> <a href="${pageContext.request.contextPath}/PDFProcessing?idB=<%=book.getId() %>" target="_blank"></a>
            <a href="${pageContext.request.contextPath}/DownloadPDFProcessing?idB=<%=book.getId() %>" target="_blank">
                </a>
        </div>
    </div>
</div>
<%>%>

```

Rysunek 9: Przykład użycia JSP Scriptlet.

#### 4.2.6 Wyrażenia JSP

Wyrażenia na stronie JSP są wykonywalnym wyrażeniem napisanym w języku skryptu określonym w deklaracji językowej (zwykle Java). Wynik wyrażenia JSP o wymaganym typie "String" jest wysyłany do standardowego strumienia wyjściowego przy użyciu bieżącego obiektu JspWriter. Jeśli wynik wyrażenia nie może być rzutowany na typ "String", występuje błąd translacji, jeśli problem został wykryty podczas etapu tłumaczenia, lub zgłoszony zostanie wyjątek "ClassCastException", jeśli podczas wykonywania kwerendy (query) wykryto niezgodność. Wyrażenie ma następującą składnię:

**<% = tekst %>**

Wyrażenia są wykonywane podczas działania protokołu HTTP. Wartość wyrażenia konwertowana w ciąg znaków i dołączana do odpowiedniej pozycji pliku JSP.

Aby uprościć wyrażenia, można użyć kilku predefiniowanych zmiennych. Najczęściej używane zmienne to:

- request - dla odwołania się do "HttpServletRequest".
- response dla odwołania się do "HttpServletResponse";
- session - dla odwołania się do "HttpSession".
- out - dla odwołania się do PrintWriter (buforowana wersja typu "JspWriter" do wysyłania danych do klienta).

## 4.2.7 Wykorzystanie JSTL w JSP

JSP Standard Tag Library (JSTL) to standardowa biblioteka znaczników, która zapewnia dostęp do znaczników kontrolowania zachowania strony, znaczników internacjonalizacji i znaczników SQL.

Żeby podłączyć "JSTL Core" na początku JSP strony podaje się polecenie "<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>".

Po podłączeniu "JSTL Core" my możemy zamienić wykorzystanie scripletów na JSTL znaczniki na stronie, co zwiększa estetykę oraz czytelność koda dla części zespołu który pracuje nad wyglądem strony [9].

Są następujące szeroko stosowane konstrukcje JSTL:

1. Dostęp do parametrów jest przeprowadzany za pomocą następnej konstrukcji: "\${user}" - odczyt parametru "user" z żądania, "\${sessionScope.user}" - odczyt parametru "user" z sesji.

2. Dla wyświetlenia na ekran znaczenia parametru wykorzystuje się znacznik "c:out" (na przykład <c:out value="{user}" />).

3. Dla deklaracji zmiennych wykorzystuje się załącznik c:set (przykład deklaracji zmiennej myName z wartością Vitalii: "<c:set var="myName" value="Vitalii"/>")

4. Konstrukcję "<c:forEach var="user" items="\${userList}">\${user}</c:forEach>" wykorzystuje się dla wyprowadzenia w pętli elementów listy userList( za pomocą atrybutu "var" deklaruje się pojedynczy element listy).

5. Instrukcja warunkowa przeprowadzana za pomocą konstrukcji "<c:if test="\${isVisible == true}">Visible</c:if>", gdzie w atrybucie "test" jest opisywany warunek.

6. W JSTL zrealizowany analogie konstrukcji wyboru "switch-case" dostępnej w Java: (rysunek 10).

```
<c:choose>
<c:when test="${val == 1}">
    <p>Equals to 1</p>
</c:when>
<c:when test="${val == 2}">
    <p>Equals to 2</p>
</c:when>
<c:otherwise>
    <p>Undefined</p>
</c:otherwise>
</c:choose>
```

**Rysunek 10: Przykład użycia konstrukcji "choose-when" w JSTL.**



7. Znacznik "<c:url>" umożliwia utworzenie adresu względem katalogu głównego aplikacji. Tego znacznika można użyć na przykład podczas tworzenia łączu "<a href='<c:url value=\"/edit\" />'><c:url value=\"/edit\" /></a>".

8. Załącznik "redirect" umożliwia przełączenia się na inny adres "<c:redirect url=\"/notfound.jsp\" />".

## 4.3 Java Servlet Facets

JavaServer Faces (JSF) to specyfikacja Java do tworzenia zorientowanych na komponenty interfejsów użytkownika aplikacji internetowych, napisanych w Javie. Służy on ułatwieniu tworzenia interfejsów użytkownika dla aplikacji Java EE. W przeciwieństwie do innych technologii MVC (Model-View-Controller), które są sterowane żadaniami, podejście JSF opiera się na wykorzystaniu komponentów. Stan komponentów interfejsu użytkownika jest zapisywany, gdy użytkownik zażąda nowej strony, a następnie zostanie przywrócony, jeśli żądanie zostanie powtórzone. JSP i Facelets są powszechnie używane do wyświetlania danych, ale JSF można dostosować do innych technologii, takich jak XUL (XML User Interface Language) [4].

### 4.3.1 Sterowanie nawigacją

Reguły przejścia JSF opisują się w pliku konfiguracyjnym XML przy użyciu załącznika "<navigation-rule>". Na przykład można przejść ze strony "feedBackPage.xhtml" na stronę logowania "login.xhtml", używając reguły przejścia "exit" określonej w aukcji (rysunek 11):

```
<navigation-rule>
  <from-view-id>/pages/feedBackPage.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>exit</from-outcome>
    <to-view-id>/LoginPage/loginPage.xhtml</to-view-id>
    <redirect />
  </navigation-case>
</navigation-rule>
```

**Rysunek 11: Deklaracja przełącznika przejścia 'exit'.**

Po zwracaniu 'exit' w wywołanym konstruktorze odbędzie się przejście zgodnie z regułą nawigacyjną.

### 4.3.2 CDI bean

Korzystając z JSF Framework, wykorzystamy koncepcje "Dependency Injection" (DI) - jeden z typów inwersji sterowania "Inversion of Control" (IoC).

Dependency Injection (DI) jest to sytuacja, w której osadzamy element jednej klasy w drugiej. W praktyce DI realizowany poprzez przekazanie parametrów konstruktora lub użycie seterów.

W JSF koncepcja "Dependency Injection" (DI) spełniona przez użycie CDI bean.

Zgodnie ze specyfikacją kontener CDI traktuje każdą klasę spełniającą następujące warunki jako zarządzany komponent bean:

- Nie jest to nie statyczna klasa wewnętrzna.
- Nie jest opatrzony adnotacją definiującą komponent EJB lub zadeklarowaną jako klasa komponentu EJB w "ejb-jar.xml".
- Nie implementuje "javax.enterprise.inject.spi.Extension".
- Ma odpowiedniego konstruktora - albo:
  - klasa zawiera konstruktor bezparametrowy
  - klasa deklaruje konstruktor z adnotacją "@Inject".

Anotacja "@Named" - oznacza klasę jako CDI Bean.

Annotation "@Inject" - używając tej adnotacji określamy punkt wtrysku. Punkt wtrysku jest używany przez kontener CDI do wstrzykiwania zależności w momencie tworzenia instancji 'bean'.

W JSF 2.3 CDI beany mają cykl życia określony w tabeli 5.

**Tabela 4: Gatunki typów życia CDI bean**

Cykl Życia	Opis
@RequestScoped	Cykl życia Używany domyślnie. Bean tworzony przy każdym żądaniu HTTP. Jeśli formularz zawiera dane przesyłane do serwera w celu przetworzenia, komponent zostanie utworzony 2 razy. Początkowo komponent jest tworzony podczas otwierania strony, z którą jest powiązany komponent. Komponent jest tworzony ponownie poprzez wysłanie formularza.
@SessionScoped	Komponent jest tworzony raz podczas tworzenia sesji i jest używany przez cały okres sesji. Zarządzany 'bean' musi być Serializable.
@ApplicationScoped	Komponent jest tworzony raz, gdy jest dostępny i używany przez cały okres użytkowania aplikacji.
@ViewScoped	Komponent jest tworzony raz podczas uzyskiwania dostępu do strony i jest używany dokładnie tak długo, jak użytkownik jest na stronie (w tym żądania ajax).
@CustomScoped	Bean żyje, gdy wchodzi w określoną mapę.
@NoneScoped	Komponent zostanie utworzony, ale nie związany z żadnym obszarem życia.

#### 4.3.3 Podłączenia AJAX w JSF

W JSF Framework jest możliwość podłączenia technologii AJAX za pomocą załącznika "f:ajax" (rysunek 12). Ten załącznik jest zwykle osadzony w innym załączniku JSF typu "h:commandButton". Wartością atrybutu "execute" jest rozdzielona spacjami lista identyfikatorów komponentów, które będą używane jako dane wejściowe podczas uruchamiania żądania AJAX.

Wartością atrybutu "render" jest rozdzielona spacjami lista komponentów, które zostaną ponownie wyświetlone po zakończeniu żądania AJAX.

```
<h:commandButton actionListener="#{bean.calculate}">
    <f:ajax execute="first second" render="sum" />
</h:commandButton>
```

**Rysunek 12: Przykład użycia "AJAX" w "JSF".**

## 4.4 Spring Framework

Spring Framework to uniwersalna technologia z otwartym kodem źródłowym na platformie Java. Spring opiera się na dwóch podstawowych założeniach: "Dependency Injection" (pojęcie "Dependency Injection" (DI) zostało wyjaśnione w podrozdziale "Java Servlet Facets") oraz "Aspect oriented programming" (AOP).

AOP polega na tym, że logika biznesowa aplikacji nie jest podzielona na obiekty, lecz na relacje (concerns). Funkcje obejmujące wiele punktów aplikacji nazywane są "cross-cutting concerns" lub problemami przekrojowymi i są one oddzielone od logiki biznesowej samej aplikacji.

Kluczową jednostką w OOP jest "obiekt", a kluczową jednostką w AOP jest „aspekt”. Przykładem „aspektu” mogą być: bezpieczeństwo, buforowanie, logowanie.

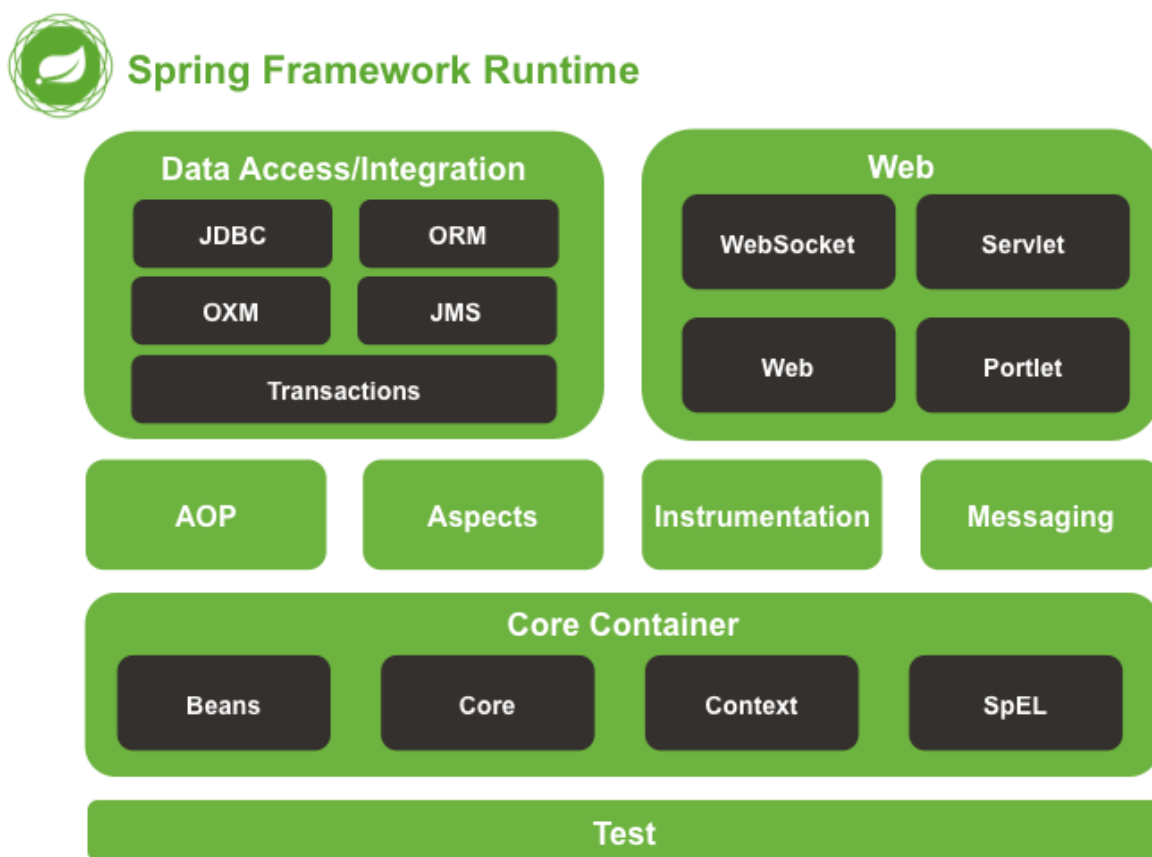
Istnieją następujące kluczowe pojęcia AOP [11]:

- Aspekt zmienia zachowanie reszty kodu, stosując poradę (ang. advice) w punktach połączenia określonych przez określony wycinek. Ten sam aspekt można wykorzystać do wprowadzenia dodatkowych funkcjonalności.
- Porady (ang. advice) - kod, który należy wywołać z punktu połączenia. Porada może być wykonana przed, po lub zamiast punktu połączenia.
- Punkt łączenia (ang. join point) - punkt w wykonywalnym programie (wywołanie metody, tworzenie obiektu, dostęp zmienny), w którym należy zastosować poradę.
- Przecięcie (ang. pointcut) - zestaw punktów łączenia w których ma być wykonana porada.
- Wprowadzenie (ang. introduction) - dodaje nowe atrybuty i / lub metody do istniejących klas.
- Cel (ang .target) - obiekt, do którego zostaną zastosowane porady.
- Przeplatanie (ang. weaving) - łączenie obiektów z odpowiednimi aspektami (być może na etapie kompilacji, ładowania lub wykonywania programu).
- Istnieje kilka rodzajów porad (ang .advice):
  - Before - uruchamia poradę przed wykonaniem metody.

- After - uruchamia poradę po wykonaniu metody, bez względu na wynik jej działania.
- After-returning - uruchamia poradę po wykonaniu metody, tylko jeśli zakończy się powodzeniem.
- After-throwing - uruchamia poradę po wykonaniu metody, tylko gdy ta metoda generuje wyjątek.
- Around - uruchamia poradę przed i po wykonaniu metody.

#### 4.4.1 Architektura Spring Framework

Dzisiaj Spring jest podzielony na kilka indywidualnych modułów(rysunek 13).



**Rysunek 13: Struktura Spring Framework.**

Kontener podstawowy (Core Container) zawiera w sobie "Beans", "Core", "Context" [11]:

- Beans jest odpowiedzialna za "BeanFactory", który jest złożoną implementacją wzorca "Factory" (GoF).
- Moduł "Core" zapewnia kluczowe elementy struktury, w tym właściwości "IoC" i "DI".
- Kontekst jest zbudowany na podstawie "Beans" and "Core" i umożliwia dostęp do dowolnego obiektu zdefiniowanego w ustawieniach.

- Moduł "SpEL" zapewnia potężny język wyrażeń służący do manipulowania obiektami podczas wykonywania.

Kontener "Data Access / Integration" składa się z "JDBC", "ORM", "OXM", "JMS" i modułu "Transactions":

- JDBC narzędzia wykorzystywane do połączenia z bazą danych. Do pracy z relacyjnymi bazami danych Spring Framework zapewnia narzędzie "JdbcTemplate", który jest obwolutą standardowego narzędzia JDBC. "JdbcTemplate" jest prosty i elastyczny w użyciu, jego możliwości są wystarczające do realizacji projektów małych i średnich rozmiarów.
- Moduł ORM zapewnia integrację z takimi popularnymi ORM jak "Hibernate", "JPA".
- Moduł OXM jest odpowiedzialny za komunikację z "Object / XML" - XMLBeans.
- Moduł "Java Messaging Service" (JMS) jest odpowiedzialny za tworzenie, wysyłanie i odbieranie wiadomości.
- Transakcje obsługują zarządzanie transakcjami dla klas, które implementują określone metody.
- Web warstwa składa się z "Web", "Web-MVC", "Web-Socket", "Web-Portlet":
  - Web moduł udostępnia takie funkcjonalność, jak na przykład przesyłanie plików.
  - Web-MVC realizują implementację Spring MVC dla aplikacji internetowych.
  - Web-Socket zapewnia obsługę komunikacji między klientem a serwerem za pomocą gniazd sieciowych w aplikacjach internetowych.
  - Web-Portlet zapewnia implementację MVC do środowiska portleta.
- Spring zawiera również szereg innych ważnych modułów, takich jak AOP, Aspects oraz Test:
- AOP implementuje programowanie zorientowane na aspekt i pozwala na wykorzystanie całego arsenału możliwości AOP.
- Moduł "Aspects" zapewnia integrację z "AspectJ", który jest również potężnym środowiskiem AOP.
- Wreszcie moduł "Test" zapewnia testowanie przy użyciu "JUnit Framework".

#### 4.4.2 Obiekty bean w Spring Framework

Bean to obiekt, który jest podstawą aplikacji i są zarządzany przez kontener IoC. Obiekty te są tworzone przy użyciu metadanych konfiguracji określonych w kontenerze [11].

Istnieją trzy główne sposoby konfigurowania aplikacji (czyli określenie, których obiektów potrzebujemy dla pracy z Springiem):

- przy użyciu plików XML konfiguracji;

- używając Java konfiguracji;
- automatyczna konfiguracja.

Najbardziej priorytetową jest automatyczna konfiguracja oraz najmniej priorytetowym jest wykorzystanie konfiguracji XML. Dla konfiguracji beanów za pomocą XML wykorzystuje się załącznik `<bean>...</bean>`) oraz następujące atrybuty/wbudowane załączniki:

- `class`(obowiązkowy atrybut) - wskazuje na konkretną klasę aplikacji Java, która zostanie użyta do utworzenia komponentu bean.
- `name` - unikalny identyfikator komponentu bean.
- `scope` (atrybut opcjonalny) - właściwość, która określa zakres działania tworzonych obiektów. W Spring Framework można wymienić następujące wartości dla danej właściwości:
  - `singleton` - definiuje jeden pojedynczy komponent bean dla każdego kontenera Spring IoC (używany domyślnie).
  - `prototype` - pozwala mieć dowolną liczbę wystąpień komponentu bean.
  - `request` - jedna instancja komponentu bean jest tworzona dla każdego żądania HTTP. Dotyczy wyłącznie aplikacji `ApplicationContext`.
  - `session` - jedna instancja komponentu bean jest tworzona dla każdej sesji HTTP.
  - `global-session` - jedna instancja komponentu bean jest tworzona dla każdej globalnej sesji HTTP.
- `constructor-arg` - definiuje konstruktor używany do wstrzykiwania zależności.
- `properties` - wbudowany załącznik, który definiuje właściwości wprowadzenia zależności.
- `property` - wbudowany załącznik, za pomocą którego jest możliwość przepisania wartości domyślnej do wybranych pól beana.
- `initialization method` (atrybut opcjonalny) - w tym atrybucie jest definiowana metoda inicjowania komponentu bean.
- `destruction method` - metoda niszczenia beana używanego podczas niszczenia pojemnika tworzącego bean.
- `autowiring mode` (atrybut opcjonalny) - określa tryb automatycznego wiązania dla wstrzykiwania zależności.
- `lazy-initialization mode` (atrybut opcjonalny) - tryb leniwej inicjalizacji (kontener IoC utworzy bean przy pierwszym żądaniu, a nie przy uruchomieniu aplikacji).
- `parent` (atrybut opcjonalny) - atrybut który wskazuje na beana rodzica w przypadku dziedziczenia beanów.

Dla konfiguracji beanów za pomocą Java konfiguracji wykorzystywane są adnotacje "@Configuration" i "@Bean".

"@ Configuration" - adnotacja, używana przed klasą, co będzie wykorzystana przez kontener Spring IoC jako klasa konfiguracji dla beanów.

@Bean - adnotacja "@Bean" określona przed metodą informuje Spring, że obiekt zwrócony przez tę metodę musi być zarejestrowany jako bean.

Dla automatycznej konfiguracji beanów wykorzystuje się adnotacja "@Component". Adnotacja "@Component" wpisywana przed klasą i sygnalizuje IoC kontenerowi, że klasa jest beanem. Jeśli musimy wyraźnie wskazać Spring, że bean dla tej klasy musi mieć określoną nazwę - nazwa ta może być podana w nawiasach po adnotacji.

Również w Spring Framework istnieją następne adnotacje, które pomagają powiązać beany pomiędzy sobą:

- @Required - adnotacja, która sygnalizuje, że wartość pola, co ustawia wymieniona metoda, musi być ustawiona w pliku XML. Jeśli tego nie zrobimy, otrzymamy wyjątek "BeanInitializationException".
- @Autowired - adnotacja, która zapewnia kontrolę nad tym, gdzie i jak powinno być wykonane automatyczne łączenie. Możemy wykorzystywać "@Autowired" zarówno dla metod, jak i konstruktorów.
- @Qualifier - adnotacja jest wykorzystywana żeby powiedzieć kontenerowi którego beana potrzebujemy w przypadku jeśli wystąpiła sytuacja taka, że mamy kilka beanów tego samego typu.

Domyślnie łączenie adnotacji nie jest zawarte w kontenerze Spring. Dlatego przed użyciem adnotacji musimy zezwolić na ich użycie w pliku konfiguracyjnym Spring za pomocą konstrukcji "<context:annotation-config />".

#### 4.4.3 Spring Boo

Spring Boot to technologia do szybkiego tworzenia aplikacji opartych na Spring Framework i jego komponentach zawartych w Spring Data, Spring Security i innych podprojektach. Spring Boot zapewnia ogromną liczbę skonfigurowanych komponentów, co pozwala skrócić czas poświęcony na konfigurację aplikacji i skupić się bezpośrednio na programowaniu, a także uprościć pracę z zależnościami(rysunek 14).

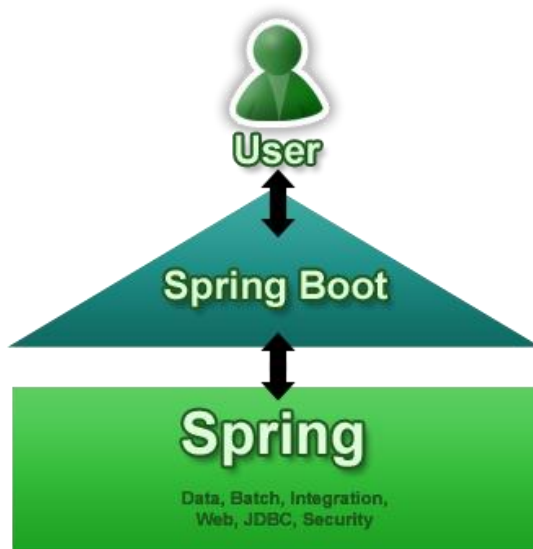
Spring Boot nie jest środkiem do automatycznego generowania kodu, ale jest wtyczką do systemu automatyzacji projektu (obsługuje Maven i Gradle).

Wtyczka zapewnia funkcje testowania i wdrażania aplikacji Spring. Polecenie "mvn spring-boot: run" umożliwia uruchamianie aplikacji na porcie "8080". Ponadto Spring Boot pozwala spakować aplikację do osobnego pliku "jar" z wbudowanym kontenerem webowym.

Główną zaletą Spring Boot jest konfiguracja zasobów na podstawie zawartości ścieżki klas. Na przykład, jeśli plik "pom.xml" projektu Maven zawiera zależności JPA i sterownik dla "PostgreSQL", Spring Boot skonfiguruje moduł trwałości dla "PostgreSQL". W przypadku jeśli dodana zależność sieciowa, domyślnie zostanie skonfigurowano Spring MVC.

Główne cele Spring Boot:

- Zapewnienie szybszego początku pracy z Spring Framework.
- Zapewnia szereg нефunkcyjnych funkcjonalności, które są wspólne dla większości projektów (na przykład wbudowane serwery, zabezpieczenia, metryki, kontrole kondycji, konfiguracja zewnętrzna).



**Rysunek 14: Spring Boot jako punkt skupienia na większym ekosystemie Spring.**

#### 4.4.4 Spring Security

Spring Security to platforma Java / JavaEE, która zapewnia mechanizmy budowania systemów uwierzytelniania i autoryzacji, a także inne funkcje bezpieczeństwa dla aplikacji korporacyjnych utworzonych za pomocą Spring Framework.

Możliwości:

- Kompleksowa i rozszerzalna obsługa zarówno uwierzytelniania, jak i autoryzacji.
- Ochrona przed atakami, takimi jak utrwalanie sesji, clickjacking, fałszowanie żądań między witrynami.
- Integracja z Servlet API.
- Integracja z Spring Web MVC.

Kluczowe obiekty kontekstowe "Spring Security":



- "SecurityContextHolder" zawiera informacje o bieżącym kontekście bezpieczeństwa aplikacji, w tym szczegółowe informacje o użytkowniku (Principal) aktualnie pracującym z aplikacją.
- "SecurityContext" zawiera obiekt "Authentication" i, w razie potrzeby, informacje dotyczące bezpieczeństwa związane z żądaniem od użytkownika.
- "Authentication" reprezentuje użytkownika (Principal) z punktu widzenia Spring Security.
- "GrantedAuthority" odzwierciedla uprawnienia przyznane użytkownikowi w całej aplikacji, takie uprawnienia (zwykle nazywane "rolami"), na przykład "ROLE\_ANONYMOUS", "ROLE\_USER", "ROLE\_ADMIN".
- "UserDetails" zapewnia informacje niezbędne do zbudowania obiektu uwierzytelnienia z DAO-obiektów aplikacji lub innych źródeł danych bezpieczeństwa.
- "UserDetailsService" służy do utworzenia obiektu "UserDetails" poprzez zaimplementowanie metody tego interfejsu "loadUserByUsername(nazwa użytkownika)".

#### 4.4.5 Dialekt Thymeleaf

Thymeleaf to nowoczesny silnik szablonów po stronie klienta dla środowisk internetowych i autonomicznych, który obsługuje HTML, XML, JavaScript, CSS, a nawet zwykły tekst.

Głównym celem Thymeleaf jest stworzenie eleganckiego i wygodnego sposobu na produkowanie szablonów. Thymeleaf opiera się na koncepcji szablonów naturalnych, żeby wprowadzić własną logikę w plikach szablonów, aby szablon ten nie wpływał na wyświetlanie dyzajnu prototypu [8]. Przykład podłączenia "Thymeleaf":

```
"<html xmlns:th="http://www.thymeleaf.org">".
```

Są następujące szeroko stosowane konstrukcje "Thymeleaf" [8]:

1. Atrybut załącznika th: text = "# {key}" można użyć do wyświetlenia wartości z plików właściwości.
2. Atrybut załącznika th: text = "\${attributename}" można użyć do wyświetlenia wartości z sesji lub model.
3. Jeśli atrybut modelu jest zestawem obiektów, można użyć atrybutu th:each="<object:\${objectList} załącznika, aby przejść przez obiekty (na przykład: <div th:each="user: \${userList}">).
4. Dla podłączenia zewnętrznego fragmentu można użyć atrybutu "th:include", "th:insert", "th:replase" załącznika.

- "th:replase" - zamienia blok załącznika na blok fragmentu.
- "th:insert" - wstawia treść bloku fragmentu w blok załącznika.
- "th:include" - - wstawia blok fragmentu w blok załącznika.

5. Atrybut `th: if = "$ {condition}"` służy do wyświetlenia bloku (załącznika), jeśli warunek jest spełniony. Atrybut `th:unless = "$ {condition}"` służy do wyświetlania bloku (załącznika), jeśli warunek nie jest spełniony.

6. Atrybuty `th:switch` i `th:case` są używane do warunkowego wyświetlania zawartości przy użyciu struktury instrukcji "switch".

7. Dane wejściowe formularza można przetwarzać za pomocą atrybutów `th:action = "@{url}"` i `th:object = "$ {object}"`. `Th:action` służy do podania adresu URL akcji formularza, a `th:obiekt` służy do określenia obiektu, z którym zostaną powiązane dane przesłanego formularza. Poszczególne pola są wyświetlane za pomocą atrybutu `th:field = "*** {name}"`, gdzie "name" jest pasującą właściwością (polem) obiektu.

8. `th:style="@{/path}"` - ustawienia stylu elementu dla CSS, gdzie "path" - względna ścieżka do elementu.

9. `th:href="@{/url(param1=value1, param2=value2)}"` - atrybut, który generuje wartość atrybutu `href` w linku załącznika "<a>", gdzie `param1`, `param2` - nazwy parametrów, a `value1`, `value2` -wartości.

## 5 Wykorzystywane narzędzia oraz technologie

Istnieje sporo narzędzi dla budowania web-aplikacji z wykorzystaniem języka Java. W niniejszym rozdziale zostaną wymienione wszystkie narzędzia, które zostały użyte dla stworzenia aplikacji webowych z wykorzystaniem różnych technologii oraz narzędzia służące do przeprowadzenia badań nad nimi.

### 5.1 GlassFish Application Server

GlassFish - wzorcowa implementacja serwera aplikacji dla Javy EE 7. Pierwszy serwer aplikacyjny w którym zaimplementowano najnowszy interfejs "API Java EE". GlassFish jest otwartym i bezpłatnym produktem co został wyprodukowany Sun Microsystems przy wsparciu Oracle Corporation. GlassFish ma modułową architekturę opartą na OSGi i rozwijaną poprzez warstwę abstrakcji HK2. Pełny interfejs administracji internetowej (przepisany dla wersji 2 z JSF i AJAX), interfejs wiersza poleceń, skrypty i wbudowane narzędzia programistyczne. GlassFish zapewnia wiele dostępnych właściwości za pośrednictwem JMX, na przykład za pośrednictwem JConsole.

Serwer GlassFish zawiera interfejs sieciowy (Rysunek 15), który można odtworzyć pod adresem URL: "http: // localhost: 4848" w dowolnej przeglądarce po uruchomieniu serwera GlassFish. Żeby uruchomić serwer trzeba przejść do folderu z lokalizacją GlassFish, uruchomić 'GlassFish.bin' i wisać polecenie "/asadmin start-domain domain1". Po wdrożeniu aplikacji na serwerze można ją odtworzyć używając adresu - "http://localhost:8080".



Rysunek 15: Konsola administracyjna serwera aplikacji GlassFish.

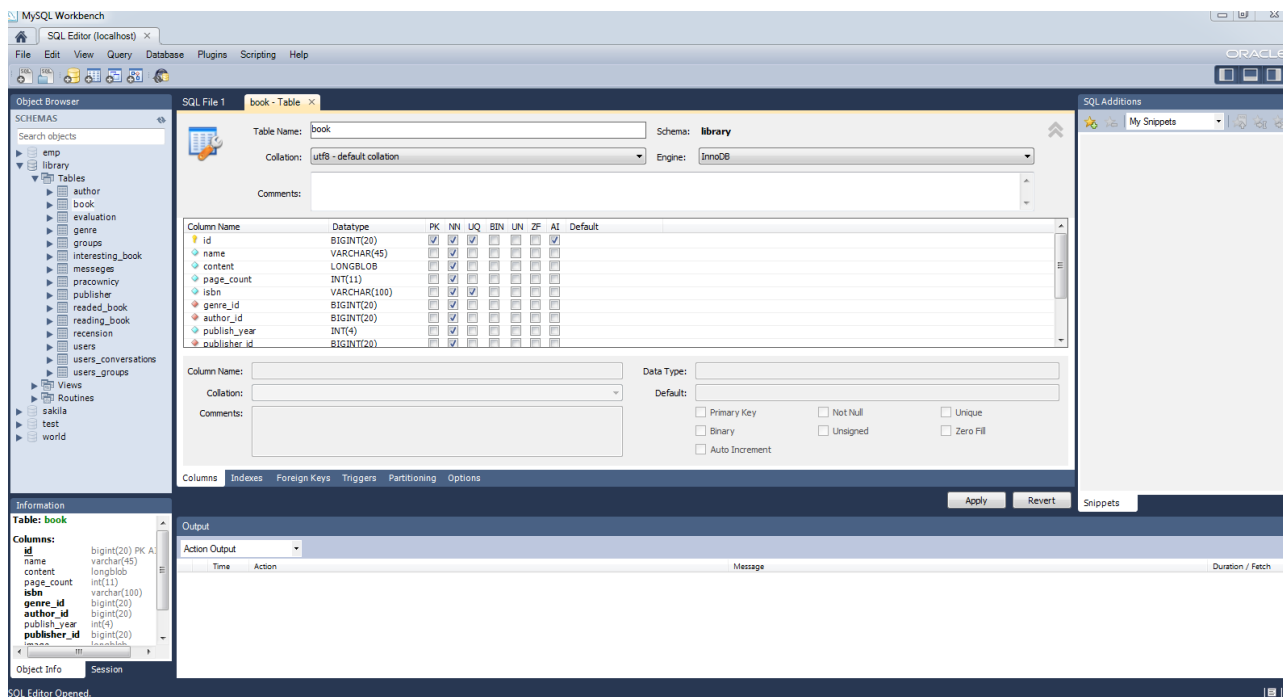
Zalety serwera GlassFish:

- 1) Bezpłatna licencja.
- 2) Implementacja wzorcowa Java EE: co oznacza, że deweloperzy innych serwerów aplikacji mogą korzystać z GlassFish aby upewnić się, że ich produkty są zgodne ze specyfikacją. Teoretycznie GlassFish może być używany do debugowania innych serwerów aplikacji. Jeśli aplikacja została błędnie wdrożona na innym serwerze aplikacji, ale na GlassFish wykonują się dobrze - może to wskazywać, że inny serwer aplikacji działa nieprawidłowo.
- 3) Obsługuje najnowsze wersje specyfikacji Java EE.

## 5.2 Serwer MySQL

MySQL to darmowy system zarządzania bazami danych z otwartym kodem źródłowym wyprodukowany przez Sun Microsystems oraz wspierany przez Oracle Corporation. MySQL jest dobrym rozwiązaniem dla małych i średnich aplikacji. Zwykle MySQL jest używany jako serwer, do którego uzyskują dostęp lokalni lub zdalni klienci, ale pakiet dystrybucyjny zawiera wewnętrzną bibliotekę serwerów, która umożliwia dołączenie MySQLdo samodzielnych programów.

Do łatwego tworzenia oraz zarządzania, wizualnego projektowania baz danych istnieje aplikacja o nazwie MySQL Workbench (rysunek 16).



**Rysunek 16: Interfejs MySQL Workbench.**

Charakterystyka MySQL [6]:

1) Charakterystyka wewnętrzne:

- Napisany w C i C ++. Przetestowany na wielu różnych kompilatorach.
- Działa na większości nowoczesnych platform takich jak Windows, Linux, MacOS.

- W pełni wielowątkowy przy użyciu wątków jądra. Oznacza to, że łatwo jest zorganizować pracę z wieloma procesorami, jeśli sprzętowe możliwości pozwalają na to.
- Bardzo szybkie tabele dyskowe oparte na drzewach B z kompresją indeksu.
- Bardzo szybki system alokacji pamięci oparty na wątkach.
- Bardzo szybkie połączenia z wykorzystaniem zoptymalizowanej metody jednozakresowego łączenia wielokrotnego.
- Funkcje SQL są implementowane za pomocą dobrze zoptymalizowanej biblioteki klas, dzięki czemu są wykonywane tak szybko, jak to możliwe. Zwykle po zainicjowaniu zapytania nie następuje żadna alokacja pamięci.

## 2) Charakterystyka kolumn w MySQL

- Duża liczba: liczby całkowite podpisane i bez znaku, taki jak "FLOAT", "DOUBLE", "CHAR", "VARCHAR", "TEXT", "BLOB", "DATA", "TIME", "TIMESPACE", "TIMESTAMP", "YEAR", "SET" i "ENUM".
- Kolumny typu wiersz o stałej i zmiennej długości.
- Kolumny typu "BLOB" dla przechowywania plików w dowolnym formacie jako ciągu bitów.
- Wszystkie kolumny mają wartości domyślne.

## 3) Charakterystyka poleceń i funkcji w MySQL:

- Pełna obsługa operatorów i funkcji w zapytaniach "SELECT" i "WHERE".
- Pełna obsługa instrukcji dla grupowania (GROUP BY) i uporządkowania (ORDER BY) z wyrażeniami SQL.
- Obsługa funkcji: "COUNT ()", "COUNT (DISTINCT ...)", "AVG ()", "STD ()", "SUM ()", "MAX ()" i "MIN ()" w wyrażeniach SQL.
- Obsługa "LEFT OUTER JOIN" oraz "RIGHT OUTER JOIN", "INNER JOIN".
- "DELETE", "INSERT", "REPLACE" i "UPDATE" zwracają liczbę zmienionych wierszy.
- Polecenie "SHOW", które jest specyficzne dla MySQL, może być użyte do uzyskania informacji o bazach danych, tabelach i indeksach.
- Nazwy funkcji nie kolidują z nazwami tabel i kolumn.
- W tym samym zapytaniu można określić tabele z różnych baz danych.

## 4) Charakterystyka bezpieczeństwa w MySQL:

- System oparty na uprawnieniach i hasłach, zapewniając tym samym elastyczność i bezpieczeństwo.

- Hasła są chronione oraz szyfrowane, gdy są przesyłane przez sieć po podłączeniu do serwera.

#### 5) Skalowalność i ograniczenia w MySQL:

- Jest możliwość zarządzania bardzo dużymi bazami danych.
- Dozwolone jest do 32 indeksów dla każdej tabeli.
- Każdy indeks może zawierać od 1 do 16 kolumn.
- Maksymalna wielkość indeksu to 500 bitów.

#### 6) Lokalizacja w MySQL:

- Serwer może dostarczać komunikaty o błędach dla klientów w różnych językach.
- Pełna obsługa wielu różnych kodowań.
- Wybrany zestaw znaków jest używany do przechowywania wszystkich danych.
- Sortowanie odbywa się zgodnie z wybranym alfabetem.

#### 7) Nawiązywanie połączeń w MySQL:

- Klienci mogą łączyć się z MySQL przy użyciu gniazd TCP / IP lub gniazd Unix.
- Obsługa ODBC (Open-DataBase-Connectivity) dla Win32.

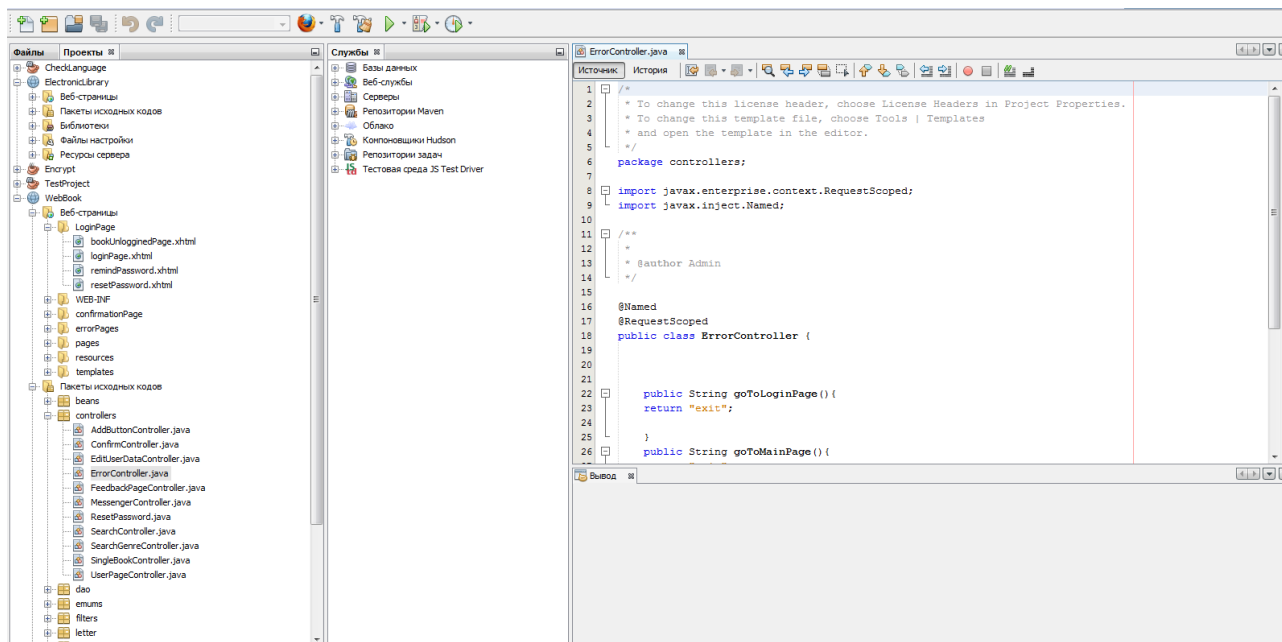
#### Zalety MySQL:

- Wielowątkowość.
- Optymalizacja połączeń.
- Zapisy stałej i zmiennej długości.
- Sterownik ODBC dołączony do źródeł MySQL.
- Elastyczny system uprawnień i haseł.
- Do 16 kluczy w tabeli. Każdy klawisz może mieć do 15 pól.
- Obsługa liczb od 1 do 4 bajtów długości (int, float, double, fixed), łańcuchów o zmiennej długości i znaczników czasu.
- Szybki system pamięci oparty na wątkach.
- Wszystkie dane są przechowywane w formacie ISO8859\_1.
- Wszystkie operacje pracy z ciągami znaków nie zwracają uwagi na przypadek znaków w przetwarzanych ciągach.
- Alias ma zastosowanie zarówno do tabel, jak i poszczególnych kolumn w tabeli.
- Wszystkie pola mają wartości domyślne. INSERT może być używany na dowolnym podzbiórze pól.
- Łatwe zarządzanie tabelami, w tym dodawanie i usuwanie kluczy i pól.

## 5.3 NetBeans IDE

NetBeans - to darmowe zintegrowane środowisko programistyczne (IDE) z otwartym kodem źródłowym wspierana przez Oracle (rysunek 17) . Środowisko zapewnia wszystkie narzędzia potrzebne do tworzenia profesjonalnych aplikacji korporacyjnych, mobilnych i internetowych na platformie Java, a także C / C ++, PHP, JavaScript, Groovy i Ruby.

Dostępny na systemach operacyjnych Windows, Mac, Linux i Solaris.



**Rysunek 17: Zintegrowane środowisko programistyczne NetBeans.**

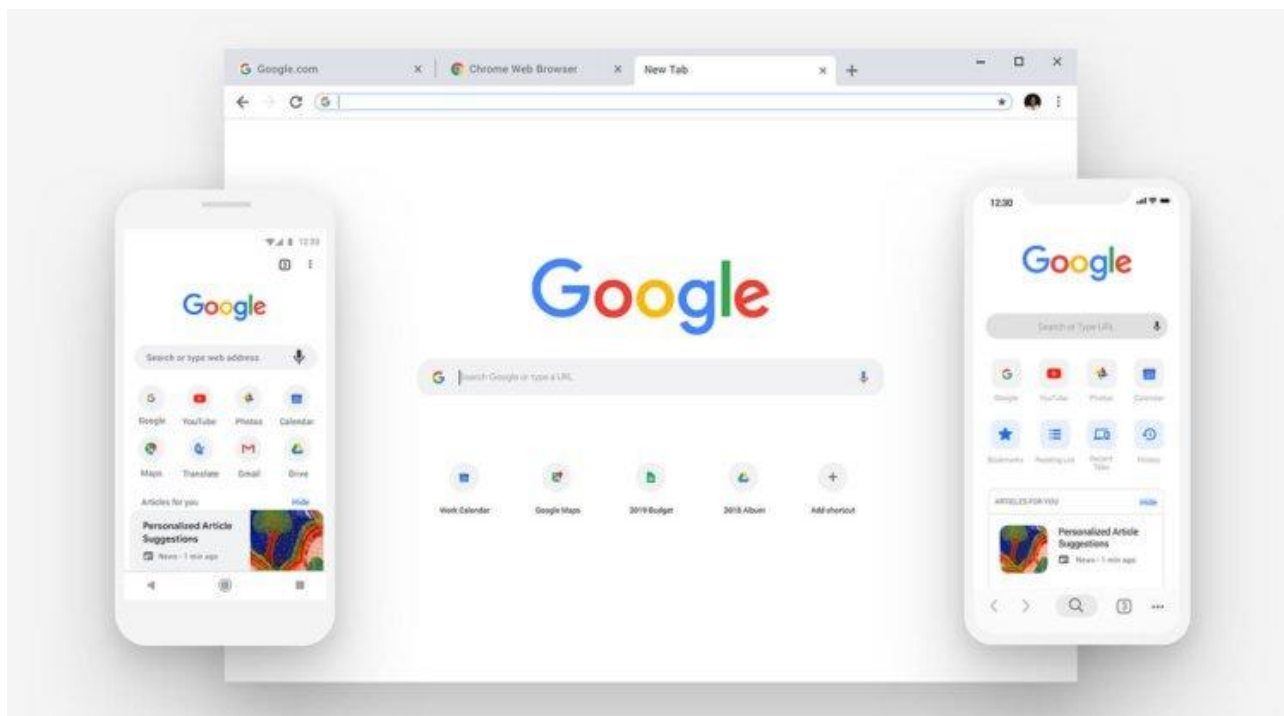
Charakterystyka NetBeans IDE [5]:

1. Obszar roboczy NetBeans IDE jest w pełni konfigurowalny.
2. NetBeans IDE zawiera zaawansowany wielojęzyczny edytor dla różnych języków programowania - Java, C / C ++, Ruby, Groovy, PHP, JavaScript, CSS, XML, HTML, RHTML, JSP, dokumentacja Javadoc.
3. Edytor NetBeans robi wierszy wcięć, sprawdza dopasowanie nawiasów i słów.
4. Sprawdza błędy podczas wprowadzania, wyświetla opcje uzupełnienia kodu i fragmenty dokumentacji dla wybranego języka programowania.
5. Jest możliwość przeglądania hierarchii i struktury dowolnej klasy Java - wyświetlane są interfejsy, klasy podstawowe, klasy pochodne i członkowie klasy.
6. Możliwe jest tworzenie projektów w wolnym formacie lub rozpoczęcie pracy z wzorcowym projektem. IDE zawiera wzorce i przykładowe projekty dla aplikacji Java SE, aplikacji mobilnych, internetowych i korporacyjnych, wtyczek NetBeans, aplikacji Groovy, PHP, C/ C ++, Ruby i Ruby on Rails.
7. NetBeans ma wbudowaną obsługę "CVS", "Mercurial" i "Subversion". Używanie edytora oznaczonego kolorami, aby wyświetlić zmiany.

8. Jest możliwość wykorzystania technologii Maven do automatyzacji konfiguracji oraz montażu projektów za pomocą opisu ich struktury w plikach 'pom.xml'.
9. Możliwość tworzenia, testowania, debugowania i wdrażania aplikacji działających na telefonach komórkowych, urządzeniach PDA, dekodernach i systemach wbudowanych.
10. Wbudowana możliwość pracy z bazami danych - wbudowany klient do baz danych - "MySQL", "Postgres", "Oracle", edytor zapytań SQL, możliwość edycji tabel bazy danych bezpośrednio za pomocą edytora tabel.
11. Integracja z serwerami aplikacji i kontenerami Serwletów - automatyczne wdrażanie aplikacji, zarządzanie serwerami - uruchamianie, zatrzymywanie, restartowanie.
12. Rozszerzenie funkcjonalności poprzez wtyczki, elastyczny system do zarządzania komponentami, modułami, aktualizowania i pobierania modułów przez Internet.

## 5.4 Przeglądarka Google Chrome oraz wtyczka "Performance page analyzer"

Dla testowania web-aplikacji zostało wykorzystana przeglądarka Google Chrome wersji 74 (rysunek 18).



**Rysunek 18: Interfejs Google Chrome.**

Google Chrome jest najpopularniejszą przeglądarką na świecie, która została wyprodukowana przez Google w 2008 roku.

Zalety Google Chrome:

- minimalistyczny wygląd
- estetyczność

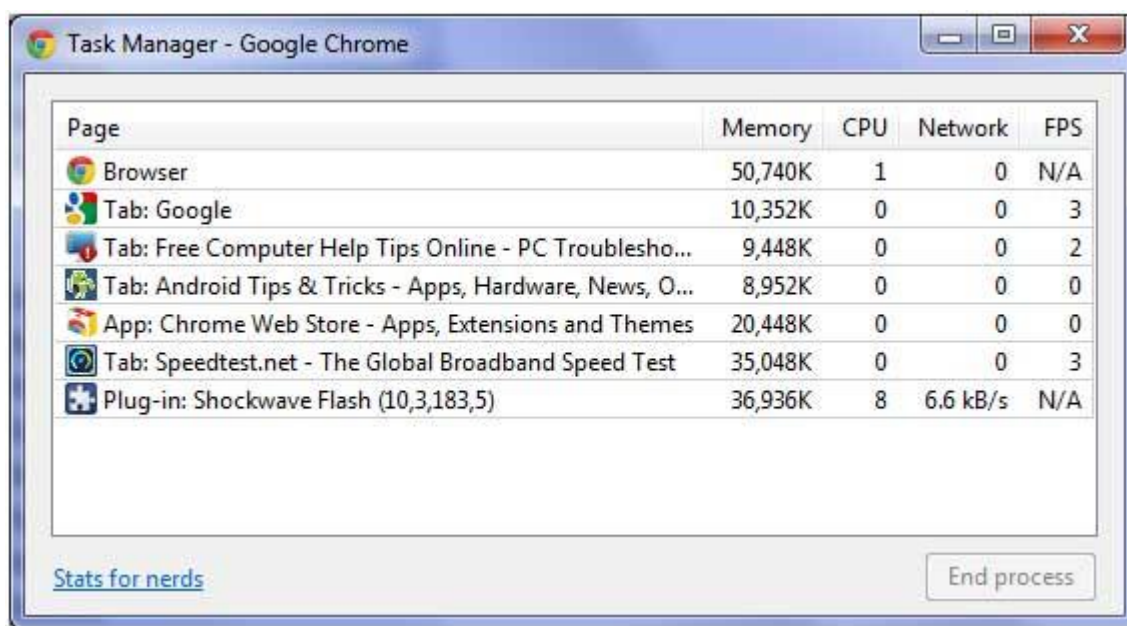


- dostępność wielu wtyczek co rozszerzają jego możliwości
- łatwa synchronizacja z kontami Google.
- omnibox - tak zwany pasek wyszukiwania i adresu w Chrome, co ułatwia wyszukiwanie.

- sklep Chrome ma największą różnorodność treści w porównaniu do innych przeglądarek. Aplikacja "Chrome Store" jest wstępnie zainstalowana w sekcji Usługi.

- duża wydajność poprzez wykorzystanie pamięci operacyjnej komputera.

Za pomocą kombinacji klawiszów "shift+esc" wywołuje się menadżer zadań przeglądarki Google Chrome (rysunek 19), który reprezentuje zużycie pamięci operacyjnej, obciążenia procesora, zużycie pamięci procesora, obciążenia sieci przez każdy proces (zakładkę), co została uruchomiona w przeglądarce.



Page	Memory	CPU	Network	FPS
Browser	50,740K	1	0	N/A
Tab: Google	10,352K	0	0	3
Tab: Free Computer Help Tips Online - PC Troublesho...	9,448K	0	0	2
Tab: Android Tips & Tricks - Apps, Hardware, News, O...	8,952K	0	0	0
App: Chrome Web Store - Apps, Extensions and Themes	20,448K	0	0	0
Tab: Speedtest.net - The Global Broadband Speed Test	35,048K	0	0	3
Plug-in: Shockwave Flash (10,3,183,5)	36,936K	8	6.6 kB/s	N/A

Stats for nerds

End process

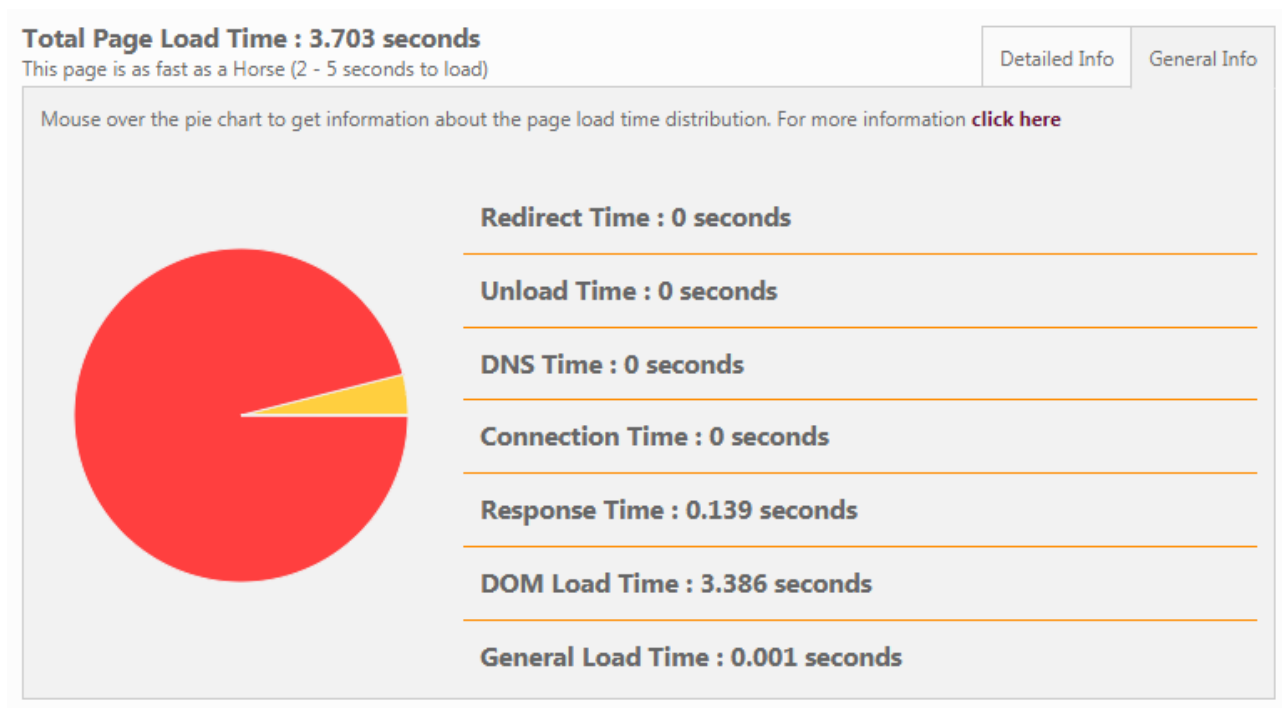
**Rysunek 19: Przykład wykorzystania Menadżer zadań w Google Chrome.**

Do testowania szybkości ładowania stron została użyta wtyczka "Analyze page performance" w Google Chrome (rysunek 20).

"Analyze page performance" wyznacza ogólny czas ładowania strony opierając się na sumę następujących poszczególnych czasów:

- Redirect Time - to czas ładowania zasobów na stronie.
- Unload Time - to czas potrzebny do rozładowania strony.
- DNS (Domain Name System) Time - to czas potrzebny na przeprowadzenie wyszukiwania domena.
- Connection Time - jest to czas potrzebny stronie do nawiązania połączenia.
- Response Time - to czas potrzebny serwerowi na odpowiedź na żądanie.
- DOM Load Time - ilość czasu potrzebnego do utworzenia elementów DOM.

- General Load Time - jest to czas wywoływania zdarzenie load (loadEventEnd - load Event Start).



Rysunek 20: Interfejs "Analyze page performance".

## 5.4 Bootstrap Framework

Bootstrap to Framework CSS, który został wyprodukowany przez firmę Twitter do użytku wewnętrznego, o początkowej nazwie roboczej Twitter Blueprint, ale ostatecznie został opublikowany w domenie publicznej i stał się dobrym zestawem narzędzi programistycznych. Aby skorzystać z tej technologii na początku pobrać pakiet z oficjalnej strony "getbootstrap.com". Następnie dodać do projektu oraz podłączyć do nagłówka projektu za pomocą polecenia "<link href='\"css/bootstrap.min.css\"' rel='\"stylesheet\"'>". Większość komponentów Bootstrap wymagają użycia JavaScript (dokładniej - potrzebują jQuery oraz Popper.js). Dla aktywacji wtyczek przed zamykającym znacznikiem "<body>" należy podłączyć bibliotekę jQuery oraz skrypt "bootstrap.min.js".

Makieta strony dzieli się stronę na następujące funkcjonalne części: nagłówek strony (header), główna część (content), kolumna boczna (sidebar) i stopka witryny (footer). Aby poprawnie wyświetlić stronę, należy obliczyć szerokość w procentach każdego elementu i przypisać właściwości wyświetlania. Przy podziale strony na części zwykle definiuje się procentową szerokość każdej z nich, ale przy użyciu danej strony na innym narzędziu czasami należy zmienić rozmiary komponentów strony. Aby rozwiązać ten problem stosuje się siatkę Bootstrap. Klasy są ustawione dla bloków, które wskazują, jak szeroki powinien być element i jak będzie wyświetlany na różnych urządzeniach. Siatka działa jak tabela, w której znajdują się wiersze i kolumny,

maksymalna liczba kolumn wynosi 12 (rysunek 21). Także siatkę można osadzić w dowolnej innej siatce [7].

.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1
.col-md-8								.col-md-4			
.col-md-4				.col-md-4				.col-md-4			
.col-md-6						.col-md-6					

**Rysunek 21: Przykład podziału kolumn w siatce Bootstrap**

Oprócz siatki istnieje ogromna liczba różnych komponentów: menu nawigacyjne, formularz, tabele, okna modalne, zakładki, alerty, podpowiedzi.

Zalety Bootstrap:

1. Obecność zmieniającego się języka stylów (Less).
2. Łatwa integracja z różnymi środowiskami programistycznymi.
3. Duży zestaw szablonów.
4. Wysoka szybkość pracy.
5. Łatwość w użyciu.

Wady Bootstrap:

1. Drobne błędy podczas pracy z JS.
2. Rzadkie konflikty niektórych narzędzi.

Powyższe niedociągnięcia są regularnie korygowane w kolejnych wersjach Bootstrap Framework.

## 5.6 Technologia Ajax

Ajax (Asynchronous Javascript And XML) - jest to skrót od podejścia do tworzenia aplikacji internetowych przy użyciu następujących technologii:

- standaryzowana prezentacja XHTML i CSS;
- dynamiczne mapowanie i interakcja użytkownika z DOM;
- wymiana danych i przetwarzanie w postaci XML i XSLT;
- Javascript;
- żądania asynchroniczne przy użyciu obiektu "XMLHttpRequest".

Podczas korzystania z AJAX nie ma potrzeby odświeżania całej strony przy każdej aktualizacji jej treści, ponieważ aktualizowana jest tylko jej konkretna część. Jest to znacznie wygodniejsze, ponieważ nie musi się długo czekać i jest bardziej ekonomiczne. To prawda, że

twórca musi upewnić się, że użytkownik jest świadomy tego, co dzieje się na stronie. Także nie wszystkie przeglądarki obsługują AJAX (starsze wersje przeglądarek i przeglądarek tekstowych).

Zalety AJAX:

- Możliwość stworzenia wygodnego interfejsu internetowego
- Aktywna interakcja użytkownika
- Częściowe przeładowanie strony zamiast pełnego
- Użyteczność

AJAX wykorzystuje dwie metody pracy ze stroną internetową: modyfikacja zawartości strony internetowej bez jej ponownego ładowania i dynamiczny dostęp do serwera. Druga może być wykonana na kilka sposobów, w szczególności "XMLHttpRequest", o którym będziemy mówić, oraz zastosowanie techniki ukrytej ramki.

Gdzie lepiej używać AJAX:

- Formularze.
- Nawigacja w postaci „drzewa”.
- System komentarzy.
- Sortowanie. Często strony sortują według daty, nazwy. Z wykorzystaniem AJAX sortowanie będzie znacznie wygodniejsze.

Gdzie lepiej nie używać AJAX:

- Wyszukiwanie.
- Normalna nawigacja.
- Aktualizacja dużej ilości tekstu.
- Dekoracje.

## 5.7 Technologia Maven

Maven jest narzędziem do budowania projektu Java: kompilowania, tworzenia pliku "jar", pakietu dystrybucyjnego dla programu i generowania dokumentacji. Do tworzenia nowego projektu używa się wtyczka "archetype" w celu (goal) - "generate." Polecenie musi określać parametry GAV (groupId, artifactId, version) oraz typ artefaktu "archetypeArtifactId".

Projekt opisuje się w pliku XML o nazwie "POM.XML" (przykładowa konfiguracja "POW.XML" przedstawiona na rysunku 22), gdzie używane są następujące załączniki i sekcje:

- project - element najwyższego poziomu projektu, opis projektu;
- GAV - parametry projektu (groupId, artifactId, version), gdzie:
  - groupId - identyfikator producenta obiektu.
  - artifactId - identyfikator obiektu.

- version - wersja obiektu.
- repositories - to miejsce, w którym przechowywane są pliki "jar", "pom", "javadoc" oraz kody źródłowe. W projekcie można zastosować:
  - centralny repozytorium dostępne do odczytu dla wszystkich użytkowników w Internecie,
  - wewnętrzny repozytorium „korporacyjne” - dodatkowe repozytorium zespołu programistów,
  - lokalny repozytorium;
- dependencies - sekcja służąca do określania zależności projektu (linki, które mówią, że na niektórych etapach cyklu życia projektu z wykorzystaniem technologii Maven wymagane są pewne artefakty). Zależności określone w sekcji "<dependencies>" pliku "POM.XML". Dla każdego artefaktu użytego w projekcie należy podać jego GAV-parametry (groupId, artifactId, version).
- build - sekcja gdzie określa się lokalizacji plików źródłowych, pliki zasobów oraz które wtyczki są używane.
- url - Internet adres projektu.
- properties - sekcja właściwości projektu;

Wykorzystanie pakietu Maven często sprowadza się do wykonania jednego z kolejnych zestawów poleceń, które nazywane celami:

- validate — walidacja meta-informacji o projekcie;
- compile — kompilacja kodów źródłowych;
- test — wdrażanie testy klasów.
- package — pakowanie skompilowanych klas w określonym formacie ( jar, war).
- integration-test — przesyłanie spakowanych klas do środowiska integracyjnego i przeprowadzanie testów.
- verify — sprawdź poprawność pakietu wobec wymagań jakościowych.
- install — wysyłanie pakietu do lokalnego repozytorium.
- deploy — wysyłanie pakietu na zdalny serwer produkcyjny.

Zalety Maven:

- Niezależność od systemu operacyjnego.
- Zarządzanie zależnościami.
- Kompilacja z wiersza poleceń.
- Dobra integracja ze środowiskiem programistycznym.
- Deklaratywny opis projektu.

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>demo</name>
  <description>Demo project for Spring Boot</description>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.3.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <java.version>1.8</java.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.webjars</groupId>
      <artifactId>bootstrap</artifactId>
      <version>3.3.6</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>

</project>

```

**Rysunek 22: Przykładowa konfiguracja "POM.XML" w projekcie z ciem Bootstrap oraz Spring Framework.**

## 6 Charakterystyka aplikacji testowych

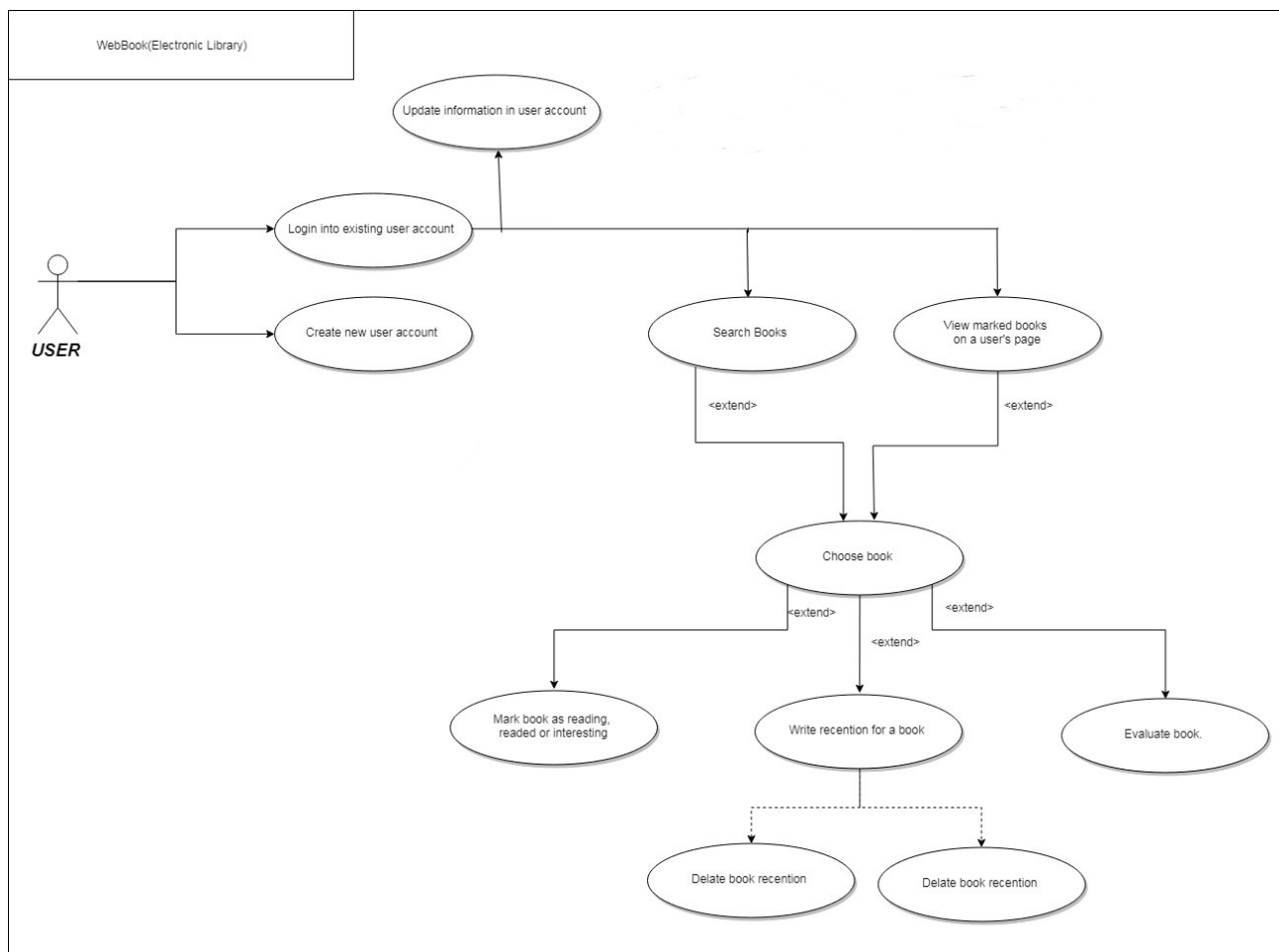
Wydajność różnych technologii do tworzenia aplikacji webowych w języku Java (JSP, JSF, Spring Framework), zostaną zbadane w oparciu na projekt Electronic Library (Biblioteka Elektroniczna), który zostanie stworzony w oparciu na każdej z powyżej wymienionych technologii oraz każda funkcjonalność tych aplikacji zostanie przetestowana według zużycia pamięci procesora (CPU), zużycia pamięci operacyjnej (RAM), szybkość wykonania kodu.

Funkcje, które zawiera biblioteka elektroniczna (rysunek 23):

- Rejestracja użytkownika.
- Logowanie do systemu.
- Wyszukiwanie książki według gatunku, pierwszej litery, słowa kluczowego.
- Pobieranie książki.
- Czytanie książki.
- Ocenianie książki.
- Zostawianie recenzji o książce.
- Usuwanie recenzji.
- Modyfikacja recenzji.
- Zaznaczona książka jako interesująca, czytająca lub przeczytana i dodawanie ją do odpowiedniej sekcji na stronie użytkownika.
- Edytowanie danych użytkownika.

W aplikacji jest zaimplementowana autoryzacja użytkowników przy logowaniu do aplikacji. Są trzy role które mogą mieć użytkownicy w tej aplikacji "user", "admin", "main\_admin". Bez uwierzytelniania obcy użytkownik nie ma dostępu do informacji oferowanej przez bibliotekę.

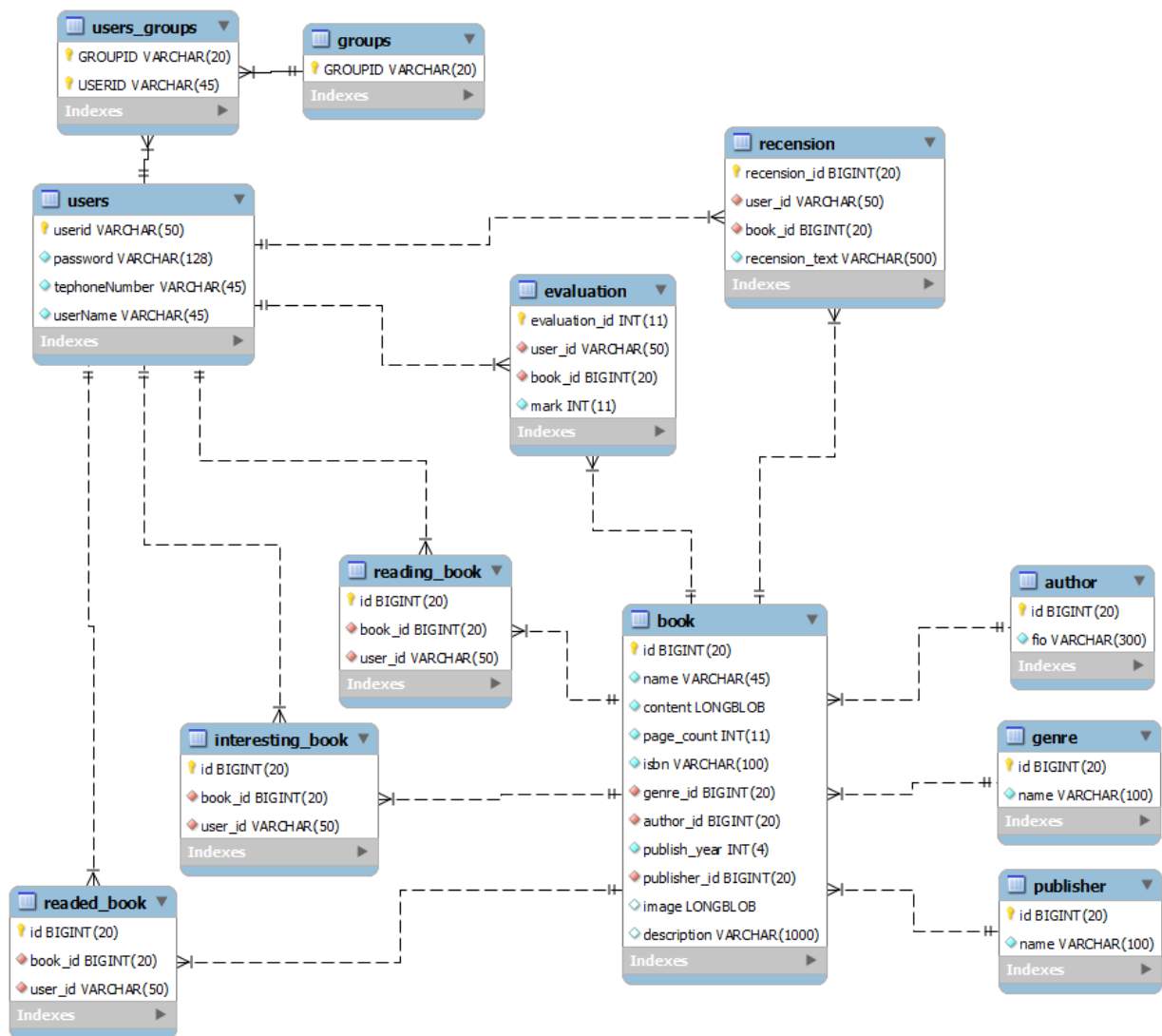
Wszystkie aplikacje działają w podobny sposób ale przy napisaniu każdej z nich została użyte różne mechanizmy oraz dodatkowe możliwości (na przykład użycie "JDBC Template" dla połączenia z bazą danych w Spring Framework), które oferowane każdą technologią. Architektura każdej poszczególnej aplikacji została dostosowana do podstawowych metodyk tworzenia w każdej konkretnej technologii.



**Rysunek 23: Diagram przypadków użycia web-aplikacji Elektronik Library.**

Dla aplikacji napisanych za pomocą technologii JSP oraz JSF wszystkie ustawienia po podłączeniu z bazą danych, uwierzytelniania i autoryzacji użytkownika zostały ustawione w konsoli administracyjnej serwera aplikacji GlassFish oraz pobierane do aplikacji przez deskryptor wdrożenia (ang. Deployment Descriptor) jako zasoby. W przypadku Spring Framework ustawienia umieszczono w pliku "application.properties" oraz została napisana własna realizacja uwierzytelniania i autoryzacji użytkownika za pomocą Spring Security. Aplikacji korzystają z bazy danych opartej na MySQL Server o nazwie "libraries"( rysunek 24) , która mieści w sobie następujące tablice: "user\_groups", "groups", "recension", "users", "evaluations", "reading\_book", "interested\_book", "readed\_book", "book", "author", "genre", "publisher", które powiązane między sobą relacjami według logiki aplikacji.





Rysunek 24: Fizyczny model bazy danych "libraries".

## 7 Badania i eksperymenty

W tym rozdziale zostaną opisane przeprowadzone badania dotyczące skuteczności tworzenia aplikacji webowych za pomocą kilka różnych metodyk w języku Java.

### 7.1 Cel oraz metodyka badań

Celem badań jest wyznaczenie najbardziej korzystnej technologii dla tworzenia aplikacji serwerowych w języku Java z punktu widzenia szybkości działania aplikacji oraz zapotrzebowania na pamięć operacyjną i moc obliczeniową procesora. Badania zostaną przeprowadzone dla trzech aplikacji napisanych w oparciu o następujące technologie: Java Servlet Pages, Java Servlet Faces oraz Spring Framework. Wszystkie aplikacje zbudowane zostały w sposób właściwy dla aplikacji napisanych w każdej z powyżej wymienionych metodyk, czyli struktura aplikacji dostosowana do technologii w której aplikacja została stworzona. Na przykład w przypadku JSP po wysłaniu żądania z klienta zostanie ono przesłane i opracowane na odpowiednim serwlecie, w przypadku JSF oraz Spring zostaną użyte odpowiedni kontroler do opracowania żądania. Także zostały użyte własne wersje realizacji JDBC Spring dla połączenia z bazą danych oraz uwierzytelniania, w przypadku aplikacji napisane w oparciu o technologie JSF i JSP wykorzystano standardową implementację JDBC oraz realizację serwera aplikacji GlassFish dla uwierzytelniania. Pomimo faktów, że struktura projektów oraz realizacja technologii wykorzystywanych dla realizacji niektórych funkcjonalności określone przez metodykę tworzenia oprogramowania różni się, logika działania aplikacji została taka sama oraz wszystkie aplikacje korzystają z tych samych zewnętrznych zasobów (na przykład wszystkie aplikacje połączone do jednej bazy danych), skutkiem czego jest minimalizacja wpływu zewnętrznych czynników na przeprowadzone badania.

W czasie badań zostały określone zwiększenia zużycia pamięci operacyjnej, wykorzystanie pamięci procesora oraz czasy ładowań stron dla każdej funkcjonalności wszystkich aplikacji. Zużycie pamięci operacyjnej oraz wykorzystania procesora wyliczone jako różnica pomiędzy maksymalną zajętością pamięci przy wdrażaniu mierzonej funkcji oraz zajętości przed jej użyciem. Każde badanie przeprowadzone pięciokrotnie, następnie została wyliczona średnia z zaokrągleniem do milisekundy w przypadku czasu oraz do kilobajta w przypadku zużycia pamięci operacyjnej oraz wykorzystania pamięci procesora.

Także w drugiej części badań będą zbadane ogólne charakterystyki aplikacji taki jak: czas potrzebny dla zapakowania projektu w "\*.war" (\*.war" - plik o dowolnej nazwie z rozszerzeniem "war") plik (w milisekundach), czas wdrożenia aplikacji na serwerze (w milisekundach), liczba bibliotek dołączanych do pliku z rozszerzeniem ".war" oraz ich waga (w kilobajtach), liczba plików

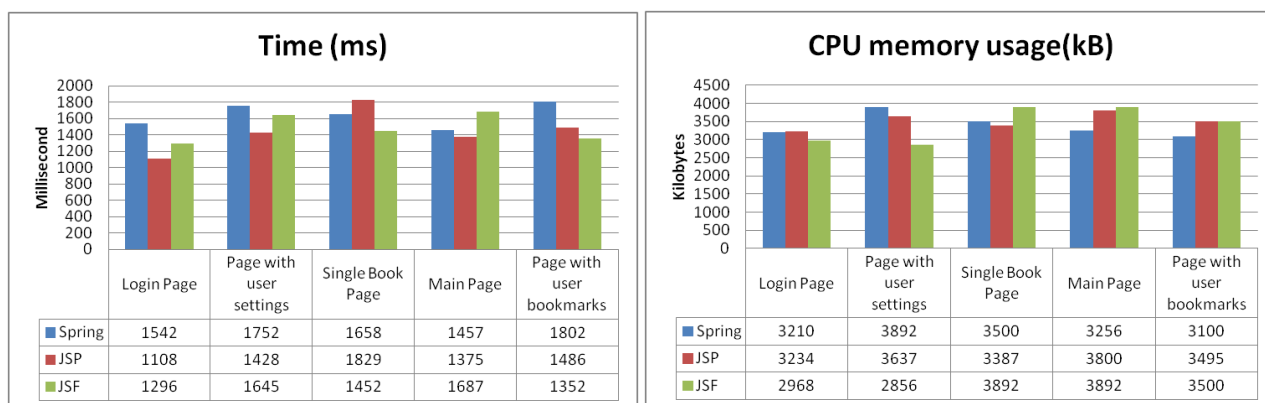
w spakowanym pliku "\*.war" i waga pliku "\*.war" (w kilobajtach). Badania zostały przeprowadzone dla każdej aplikacji stworzonej w ciągu realizacji pracy magisterskiej.

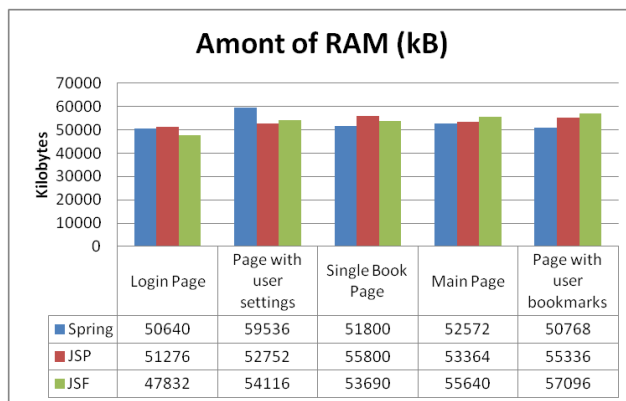
Badania zostały przeprowadzone za pomocą następujących narzędzi:

1. Wtyczka "Performance Page Analyzer" została wykorzystana dla pobrania pomiarów czasu ładowania poszczególnej strony.
2. Menadżer zadań przeglądarki Google Chrome został wykorzystany dla pobrania pomiarów zużycia pamięci operacyjnej i wykorzystania pamięci procesora przy ładowaniu poszczególnej strony aplikacji.
3. Czas pakowania został wyświetlony w wiersze poleceń pakowaniu projektu z wykorzystaniem technologii Maven.
4. Czas wdrażania został pobrany po wdrożeniu projektu na serwerze aplikacji w NetBeans IDEA.
5. Liczba bibliotek dołączanych do pliku z rozszerzeniem ".war", ich objętość, liczba plików zawieranych w pliku "\*.war" oraz objętość pliku zostały uzyskane za pomocą eksploratora systemu operacyjnego Windows, we właściwościach pliku "\*.war".

## 7.2 Przebieg badań i omówienie wyników

Na początku zostało zbadane zużycie pamięci operacyjnej, wykorzystania pamięci procesora oraz czasy ładowań dla wszystkich stron w każdej aplikacji (Rysunek 29). Obserwując diagramy na rysunku 29 można zauważyć że charakterystyki dojsć zbliżone i w poniższym przypadku nie da się wyodrębnić przewagi jakiejś konkretnej technologii.

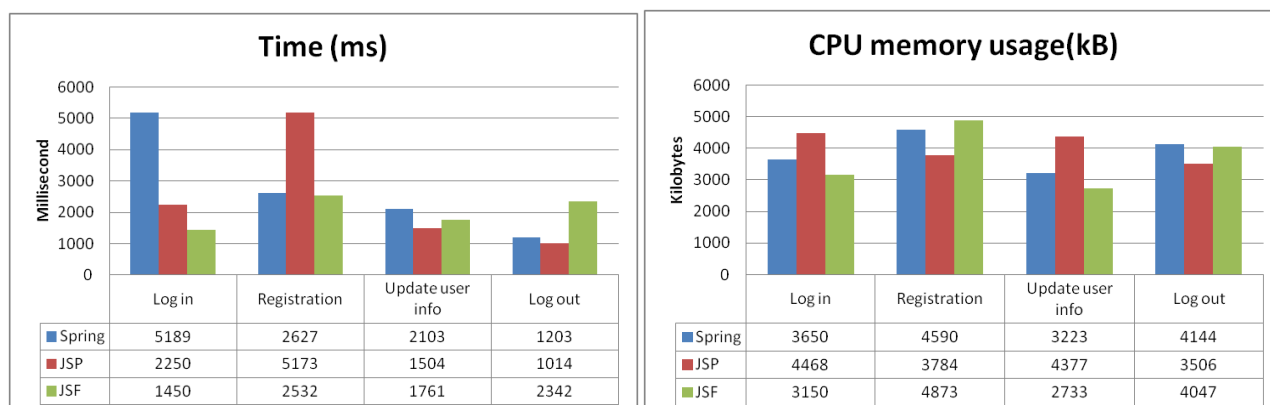


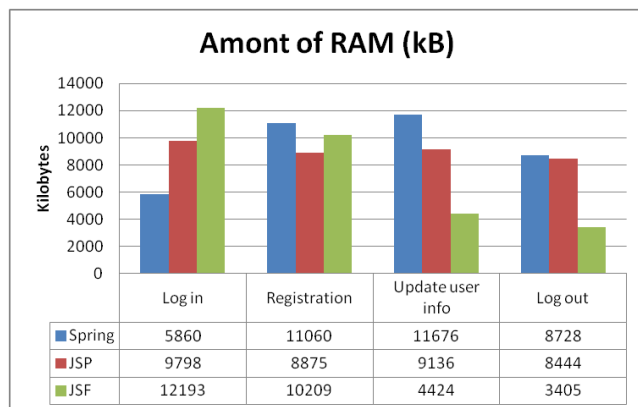


**Rysunek 29: Diagramy zużycia pamięci operacyjnej, wykorzystania pamięci procesora oraz czasy ładowań dla wszystkich stron w aplikacjach stworzonych w oparciu o technologie Spring, JSP, JSF**

Funkcjonalności aplikacji zostały podzielone na kilka grup badawczych, według celów ich realizacji oraz obiektów na których operują (na przykład operacji logowania, rejestracji, wylogowania, zmianie informacji użytkownika wszystkie powiązane z kontem użytkownika przedstawione na rysunku 30). Wszystkie pomiary dla każdej funkcjonalności realizowane z wykorzystaniem tych samych kont użytkowników aplikacji i przy operowaniu nad tymi samymi obiektami (na przykład przy pobieraniu pomiarów dla funkcjonalności logowanie zostało przeprowadzone logowanie do tego samego konta w każdej aplikacji).

Poniżej na rysunku 30 zostały przedstawione diagramy zwiększenia zużycia pamięci operacyjnej, zwiększenia wykorzystania pamięci procesora oraz czasy wykonania następujących funkcji: logowanie, rejestracji, wylogowania, zmianie informacji użytkownika (pierwsza grupa badawcza). Obserwując poniższe diagramy można zauważyć że w tej grupie badawczej najlepsze wyniki zostały zademonstrowane przez projekt napisany za pomocą technologii JSF, najgorsze wyniki zaprezentował projekt stworzony z wykorzystaniem technologii Spring.

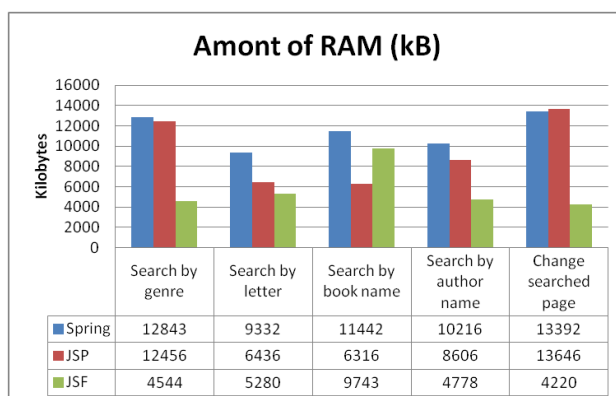
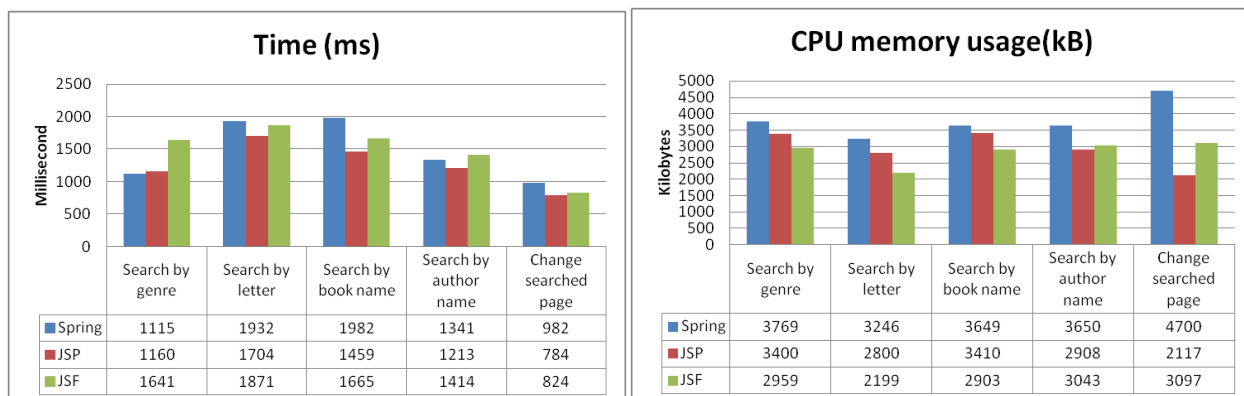




**Rysunek 30: Diagramy zwiększenia zużycia pamięci operacyjnej, zwiększenia wykorzystania pamięci procesora oraz czasu wykonania dla funkcjonalności, które powiązane z kontem użytkownika w aplikacjach stworzonych w oparciu o technologie Spring, JSP, JSF**

Następnie została rozpatrzona druga grupa badawcza. Jej funkcjonalności powiązane z różnymi gatunkami wyszukiwania książek w stworzonych aplikacjach, takich jak: wyszukiwanie książki za nazwą autora, nazwą książki jej gatunkiem oraz według pierwszej litery nazwy książki. Wyniki wyszukiwania zwracane w postaci jednej lub kilku stron na każdej z których umieszczono po trzy książki. Przełączenie pomiędzy stronami też zostało przetestowane w tej grupie (rysunek 31).

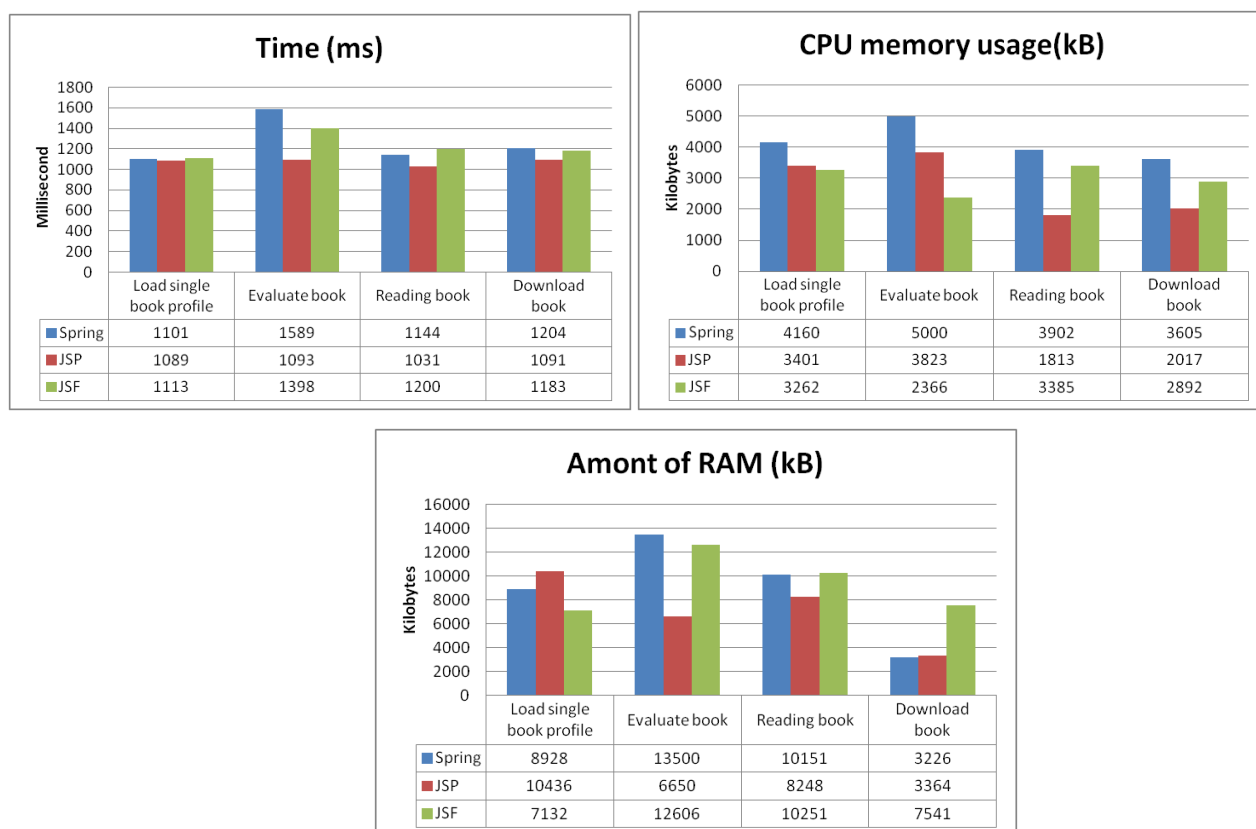
W tej grupie również jak i w poprzedniej najlepszy wyniki zostały zademonstrowane przez projekt napisany za pomocą technologii JSF, najgorsze wyniki zaprezentował projekt stworzony z wykorzystaniem technologii Spring.



**Rysunek 31: Diagramy zwiększenia zużycia pamięci operacyjnej, zwiększenia wykorzystania pamięci procesora oraz czasu wykonania dla funkcjonalności, które powiązane z różnymi rodzajami wyszukiwania książek w aplikacjach stworzonych w oparciu o technologie Spring, JSP, JSF**

W trzeciej grupie badawczej zostały zbadane funkcje, które powiązane z operowaniem na pojedynczej książce, takie jak: ładowanie profilu książki, otwieranie PDF książki w przeglądarce, ściągania książki, ocenianie książki (rysunek 32).

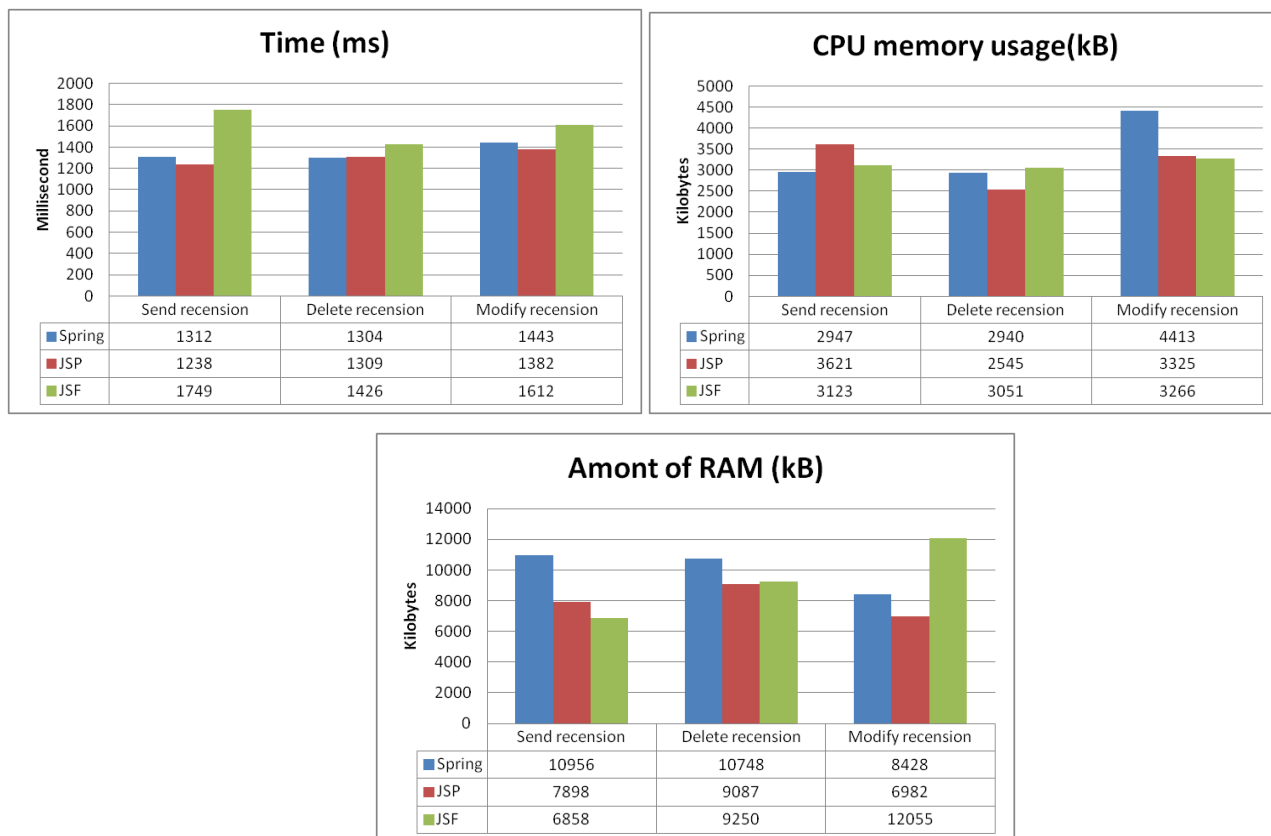
W tej grupie najlepsze wyniki zostały zademonstrowane przez projekt napisany za pomocą technologii JSP, najgorsze wyniki zaprezentował projekt stworzony z wykorzystaniem technologii Spring.



**Rysunek 32: Diagramy zwiększenia zużycia pamięci operacyjnej, zwiększenia wykorzystania pamięci procesora oraz czasu wykonania dla funkcjonalności, operowanych nad pojedynczą książką w aplikacjach stworzonych w oparciu o technologie Spring, JSP, JSF**

W czwartej grupie badawczej zostały zbadane funkcje, które powiązane były z recenzowaniem książki, takie jak: wysyłanie recenzji, usuwanie recenzji, modyfikacja recenzji (rysunek 33).

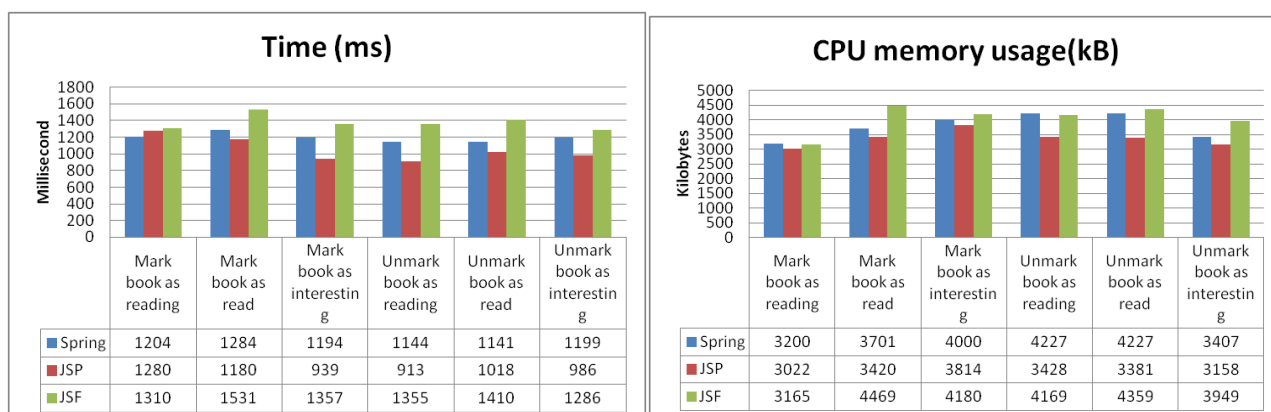
W tej grupie najlepsze wyniki zostały osiągnięte przez projekt napisany za pomocą technologii JSP, najgorsze wyniki zaprezentował projekt stworzony z wykorzystaniem technologii JSF.

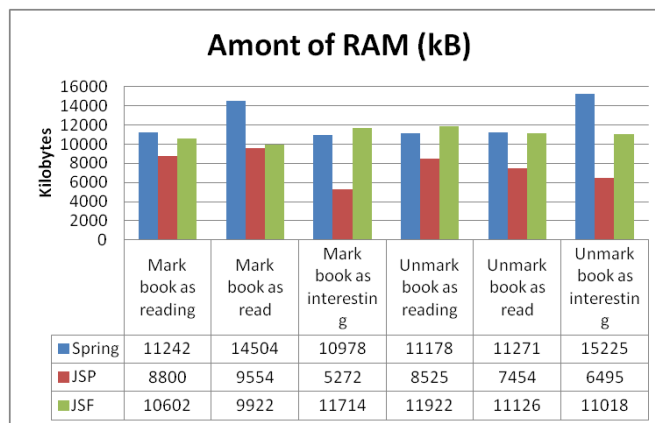


**Rysunek 33: Diagramy zwiększenia zużycia pamięci operacyjnej, zwiększenia wykorzystania pamięci procesora oraz czasu wykonania dla funkcjonalności, operowanych nad recenzjami książki w aplikacjach stworzonych w oparciu o technologie Spring, JSP, JSF**

Piąta grupa badawcza składa się funkcji, które realizują usuwanie i dodawanie książek do zakładek w kategorii przeczytane książki, czytające książki, interesujące książki (rysunek 34).

W tej grupie również jak i w poprzedniej najlepsze wyniki zostały osiągnięte przez projekt napisany za pomocą technologii JSP, najgorsze wyniki zaprezentował projekt stworzony z wykorzystaniem technologii JSF.

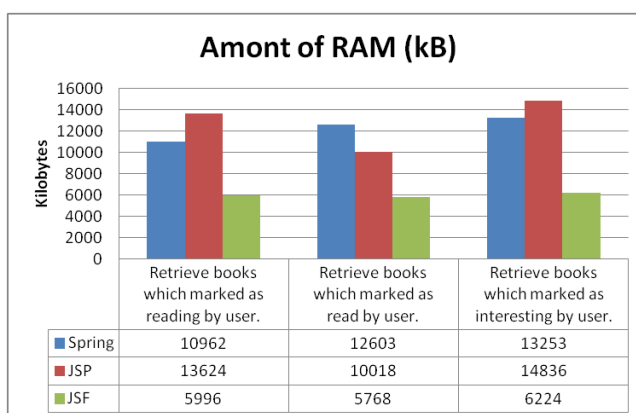
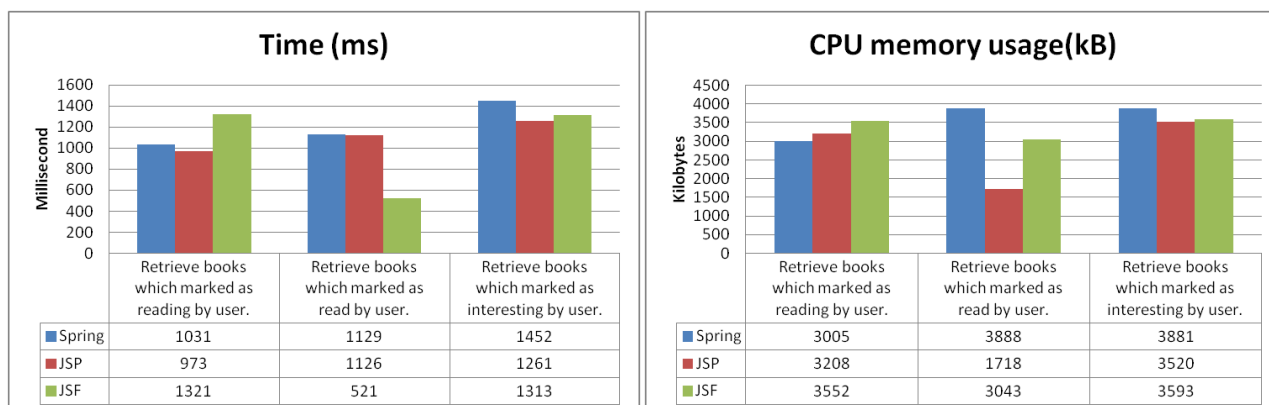




**Rysunek 34: Diagramy zwiększenia zużycia pamięci operacyjnej, zwiększenia wykorzystania pamięci procesora oraz czasu wykonania dla funkcjonalności, operowanych nad aktualizacją zakładek użytkownika w aplikacjach stworzonych w oparciu o technologie Spring, JSP, JSF**

Ostatnia (szósta) grupa badawcza zawiera funkcje, które powiązanie były z uzyskaniem książek zaznaczonych jako przeczytane, czytające oraz interesujące (rysunek 35) przez zalogowanego użytkownika.

W tej grupie najlepsze wyniki osiągnęły projekty napisane za pomocą technologii JSP oraz JSF, najgorsze wyniki zaprezentował projekt stworzony z wykorzystaniem technologii Spring.



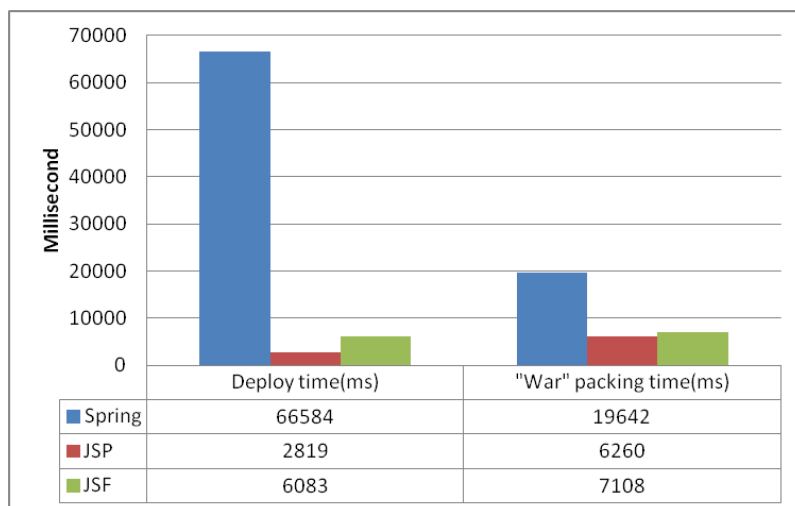
**Rysunek 35: Diagramy zwiększenia zużycia pamięci operacyjnej, zwiększenia**



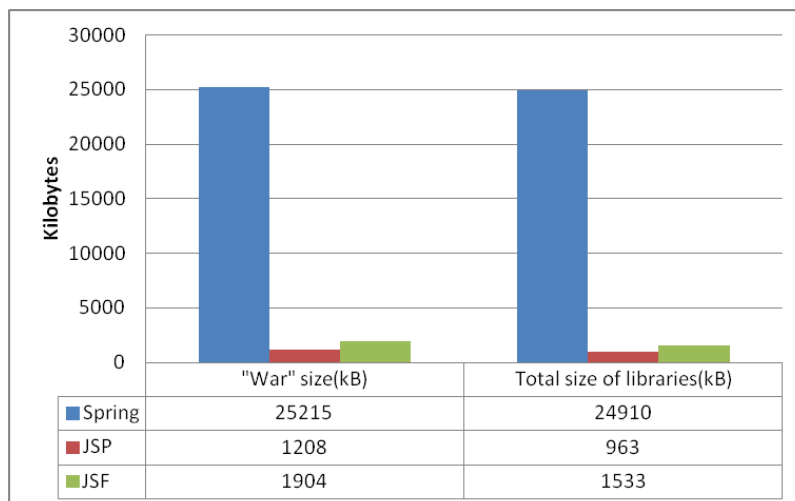
**wykorzystania pamięci procesora oraz czasu wykonania dla funkcjonalności,  
z uzyskaniem i wyświetleniem książek poprzednio dodanych do zakładek w aplikacjach  
stworzonych w oparciu o technologie Spring, JSP, JSF**

W następnej części badań zostały zbadane cechy plików "\*.war" wygenerowanych dla wszystkich aplikacji.

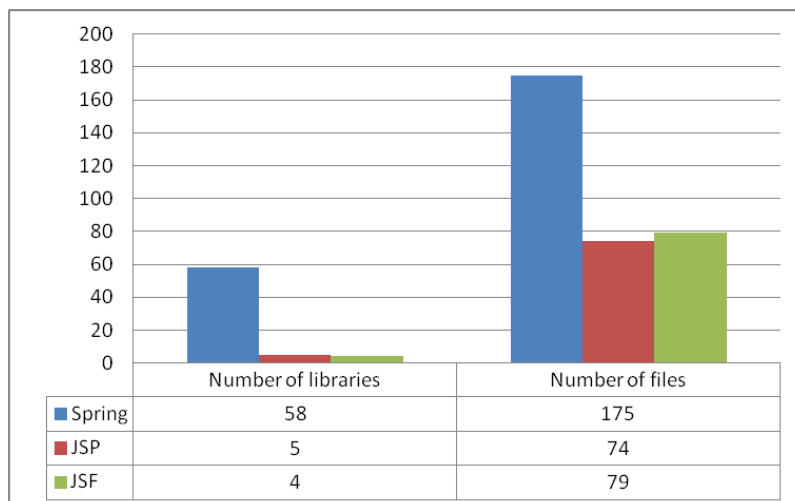
Pliki "\*.war" zostały zbadane według następujących czynników: czas potrzebny do zapakowania projektu w ".war" plik i czas wdrożenia aplikacji na serwerze (rysunek 36), waga plików oraz waga bibliotek dołączanych do każdego pliku (rysunek 37), ilość plików mieszczonych w każdym pliku "\*.war" i ilość bibliotek dołączanych do pliku (rysunek 38).



**Rysunek 36: Diagram porównania czasu potrzebnego do zapakowania projektów w ".war" pliki i czasów wdrożenia aplikacji na serwerze**



**Rysunek 37: Diagram porównania wag plików oraz wag bibliotek dołączanych do każdego pliku**



**Rysunek 38: Diagram porównania ilości plików mieszczonych w każdym pliku ".war" i ilość bibliotek dołączanych do pliku**

Obserwując rysunki 36, 37, 38 można wywnioskować, że w tej części badań aplikacja, która została napisana z wykorzystaniem Spring Framework ma dużo gorsze wyniki w porównaniu do innych aplikacji. Przy porównaniu dwóch innych aplikacji można stwierdzić, że one wykazały dość podobne wyniki, chociaż aplikacja napisana z wykorzystaniem technologii JSP ma troszeczkę lepsze wyniki dla prawie wszystkich testów.

### 7.3 Dyskusja wyników.

W pierwszej części badań było zbadano wykorzystanie pamięci operacyjnej, wykorzystanie pamięci procesora oraz czasy wykonania dla kilku grup funkcjonalności. Wyniki po grupach otrzymaliśmy następujące:

1. Pierwsza grupa: najlepsze wyniki - JSF, najgorsze - Spring.
2. Druga grupa: najlepsze wyniki - JSF, najgorsze - Spring.
3. Trzecia grupa: najlepsze wyniki - JSP, najgorsze - Spring.
4. Czwarta grupa: najlepsze wyniki - JSP, najgorsze - JSF.
5. Piąta grupa: najlepsze wyniki - JSP najgorsze - JSF.
6. Szósta grupa: najlepsze wyniki - JSP oraz JSF, najgorsze - Spring.

Obserwując wyniki według badanych jednostek, dla wszystkich funkcji otrzymaliśmy następujące wyniki:

1. Czas wykonania funkcji: najlepsze wyniki - JSP, najgorsze - JSF.
2. Wykorzystanie pamięci procesora: najlepsze wyniki - JSP, najgorsze - Spring.
3. Wykorzystanie pamięci operacyjnej: najlepsze wyniki - JSP, najgorsze - Spring.

W drugiej części badań zostały zbadane cechy spakowanych (przy pomocy technologii Maven) plików "\*.war", które zostały wygenerowane dla wszystkich aplikacji. Najgorsze wyniki zostały uzyskany dla Spring Framework oraz najlepsze JSP.

Podsumowując, można stwierdzić, że najlepsze wyniki po przeprowadzeniu badań zaprezentowała aplikacja napisana za pomocą JSP oraz najgorsze wyniki były w aplikacji napisanej za pomocą Spring Framework.

## 8. Podsumowanie i wnioski końcowe

W trakcie realizacji tej pracy zostały zbadane trzy technologie stosowane do tworzenia aplikacji webowych w języku Java (Java Servlet Faces, Java Servlet Pages, Spring Framework). Przeprowadzone badania wykazały, że najlepsze wyniki osiągnęła aplikacja napisana za pomocą technologii JSP, najgorsze- za pomocą Spring Framework.

Ale patrząc na wyżej wymienione technologie z punktu widzenia twórców oprogramowania, można stwierdzić że Spring nadaje dużo więcej narzędzi dla łatwego oraz efektywnego tworzenia aplikacji WWW, głównie dzięki możliwości wykorzystania aspektów i wstrzykiwaniu zależności. Projekty napisane za pomocą Spring Framework wykazują się elastycznością oraz podatnością na rozbudowę głównie dzięki słabemu powiązaniu między kluczowymi obiektami w aplikacji.

Aplikacja zrealizowana z wykorzystaniem technologii JSF wykazała bardzo podobne wyniki do JSP. W JSF projektach stosuje się model MVC, w przeciwieństwie do JSP, gdzie każde żądanie opracowuje konkretny serwet oraz w JSF jest możliwość wykorzystania wstrzykiwania zależności, co skutkuje słabym powiązaniem pomiędzy obiektami w JSF aplikacjach. Także w technologii JSF jest możliwość łatwego wykorzystania technologii AJAX, co zmniejsza ilość kodu w JavaScript w przypadku konieczności aktualizacji danych w warstwie klienckiej aplikacji.

Podsumowując wyniki badań oraz doświadczenia nabyte w trakcie napisania aplikacji w wyżej wymienionych technologiach można stwierdzić, że technologię JSP można efektywnie używać do tworzenia niedużych aplikacji WWW, nie posiadających złożonej logiki biznesowej.

Porównując JSF i Spring można zauważyć, że zaletami Spring w porównaniu do JSF są: większa elastyczność w realizacji i rozbudowie projektu, łatwość w wymianie interfejsów, duża liczba narzędzi ułatwiających tworzenie projektu. W przypadku JSF dużą zaletą jest łatwa możliwość wykorzystania technologii AJAX, projekty napisane z wykorzystaniem JSF Framework są szybsze oraz mniejsze w sensie objętości kodu od projektów napisanych z wykorzystaniem Spring Framework.

## 9. Bibliografia

- [1] <https://gerardnico.com/lang/java/j2ee>. [dostęp 05.06.2019]
- [2 ] JSR 245: JavaServer<sup>TM</sup> Pages 2.1 - <https://www.jcp.org/en/jsr/detail?id=245> [dostęp 05.06.2019]
- [3] JSR 315: Java<sup>TM</sup> Servlet 3.0 Specification - <https://jcp.org/en/jsr/detail?id=315> [dostęp 05.06.2019]
- [4] JSR 372: JavaServer Faces (JSF 2.3) Specification. - <https://jcp.org/en/jsr/detail?id=372> [dostęp 05.06.2019]
- [5] NetBeans documentation <https://netbeans.org/features/index.html> [dostęp 05.06.2019]
- [6] MySQL documentation [https://docs.oracle.com/cd/E17952\\_01/mysql-8.0-en/index.html](https://docs.oracle.com/cd/E17952_01/mysql-8.0-en/index.html) [dostęp 05.06.2019]
- [7] Bootstrap documentation\_ <https://getbootstrap.com/> [dostęp 03.09.2019]
- [8] Thymeleafe documentation <https://www.thymeleaf.org/documentation.html> [dostęp 03.09.2019]
- [9] JSTL documentation <https://docs.oracle.com/javaee/5/jstl/1.1/docs/tlddocs/> [dostęp 03.09.2019]
- [10] HTTP documentation <https://devdocs.io/http/> [dostęp 03.09.2019]
- [11] Spring Framework documentation <https://spring.io/docs/> [dostęp 03.09.2019]

## 10. Spis rysunków i tabel.

Rysunek 1: Aplikacje wielowarstwowe .....	10
Rysunek 2: Komunikacja z serwerem na różnych poziomach aplikacji webowej JEE .....	12
Rysunek 3: Serwer JEE i kontenery .....	13
Rysunek 4: Przykład wymiany danych w Internecie za pomoc HTTP protokołu.....	14
Rysunek 5: Przykład HTTP żądania.....	15
Rysunek 6: Przykład HTTP odpowiedzi. ....	15
Rysunek 7: Przykład deskryptora wdrożenia .....	16
Rysunek 8: Serwlety wewnątrz kontenera Java Servlet .....	17
Rysunek 9: Przykład użycia JSP Scriptlets .....	23
Rysunek 10: Przykład użycia konstrukcji "choose-when" w JSTL.....	24
Rysunek 11: Deklaracja przełącznika przejścia 'exit'. ....	25
Rysunek 12: Przykład użycia "AJAX" w "JSF". ....	26
Rysunek 13: Struktura Spring Framework. ....	28
Rysunek 14: Spring Boot jako punkt skupienia na większym ekosystemie Spring .....	32
Rysunek 15: Konsola administracyjna serwera aplikacji GlassFish. ....	35
Rysunek 16: Interfejs MySQL Workbench. ....	36
Rysunek 17: Zintegrowane środowisko programistyczne NetBeans .....	39
Rysunek 18: Interfejs Google Chrome. ....	40
Rysunek 19: Przykład wykorzystania Menadżer zadań w Google Chrome.....	41
Rysunek 20: Interfejs "Analyze page performance". ....	42
Rysunek 21: Przykład podziału kolumn w siatce Bootstrap. ....	43
Rysunek 22: Przykładowa konfiguracja POM.XML w projekcie z ciem Bootstrap oraz Spring Framework.....	46
Rysunek 23: Diagram przypadków użycia web-aplikacji Elektronik Library. ....	48
Rysunek 24: Fizyczny model bazy danych "libraries". ....	49
Rysunek 29: Diagramy zużycia pamięci operacyjnej, wykorzystania pamięci procesora oraz czasy ładowań dla wszystkich stron w aplikacjach stworzonych w oparciu o technologie Spring, JSP, JSF. ....	51

Rysunek 30: Diagramy zwiększenia zużycia pamięci operacyjnej, zwiększenia wykorzystania pamięci procesora oraz czasy wykonania dla funkcjonalności, które powiązane z kontem użytkownika w aplikacjach stworzonych w oparciu o technologie Spring, JSP, JSF.....	52
Rysunek 31: Diagramy zwiększenia zużycia pamięci operacyjnej, zwiększenia wykorzystania pamięci procesora oraz czasy wykonania dla funkcjonalności, które powiązane z różnymi rodzajami wyszukiwania książek w aplikacjach stworzonych w oparciu o technologie Spring, JSP, JSF. ....	53
Rysunek 32: Diagramy zwiększenia zużycia pamięci operacyjnej, zwiększenia wykorzystania pamięci procesora oraz czasy wykonania dla funkcjonalności, operowanych nad pojedynczą książką w aplikacjach stworzonych w oparciu o technologie Spring, JSP, JSF. ....	54
Rysunek 33: Diagramy zwiększenia zużycia pamięci operacyjnej, zwiększenia wykorzystania pamięci procesora oraz czasy wykonania dla funkcjonalności, operowanych nad recenzjami książki w aplikacjach stworzonych w oparciu o technologie Spring, JSP, JSF. ....	55
Rysunek 34: Diagramy zwiększenia zużycia pamięci operacyjnej, zwiększenia wykorzystania pamięci procesora oraz czasy wykonania dla funkcjonalności, operowanych nad aktualizacją zakładek użytkownika w aplikacjach stworzonych w oparciu o technologie Spring, JSP, JSF.....	55
Rysunek 35: Diagramy zwiększenia zużycia pamięci operacyjnej, zwiększenia wykorzystania pamięci procesora oraz czasy wykonania dla funkcjonalności, z uzyskaniem i wyświetleniem książek poprzednio dodanych do zakładek w aplikacjach stworzonych w oparciu o technologie Spring, JSP, JSF.....	56
Rysunek 36: Diagram porównania czasu potrzebnego do zapakowania projektów w ".war" pliki i czasów wdrożenia aplikacji na serwerze. ....	57
Rysunek 37: Diagram porównania wag plików oraz wag bibliotek dołączanych do każdego pliku. ....	57
Rysunek 38: Diagram porównania ilości plików mieszczonych w każdym pliku ".war" i ilość bibliotek dołączanych do pliku. ....	58
 Tabela 1: Główne metody obiektu HttpRequest.....	19
Tabela 2: Główne metody obiektu HttpServletResponse. ....	19
Tabela 3: Przykład deklaracji oraz inicjalizacji atrybutów i metod. ....	22
Tabela 4: Gatunki typów życia CDI bean. ....	26