Podstawowe informacje o technologii Java EE 6

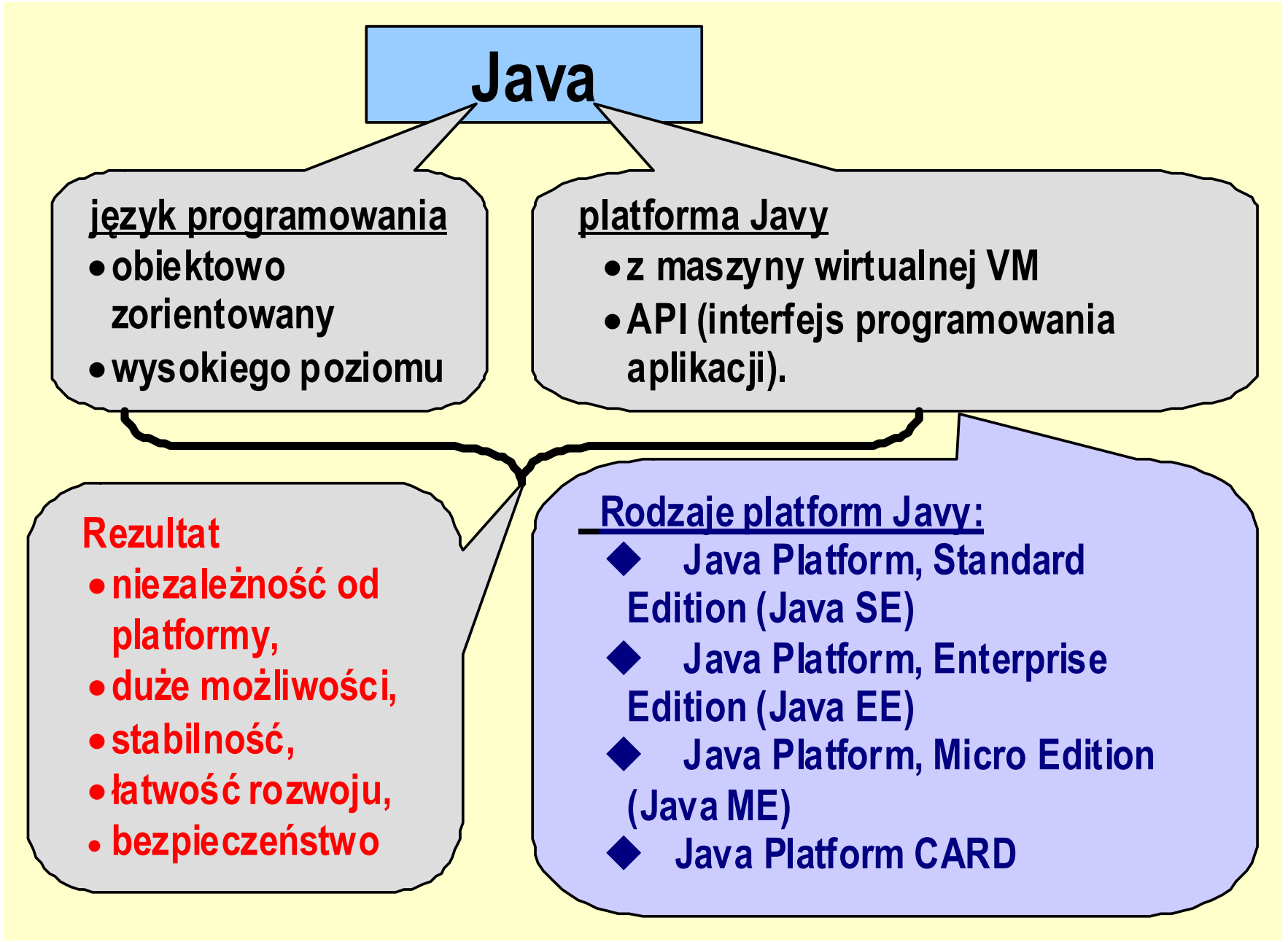
na podstawie

http://docs.oracle.com/javaee/6/tutorial/doc/

Programowanie komponentowe 1

Wprowadzenie do techonologii Java EE 6

Platformy Javy



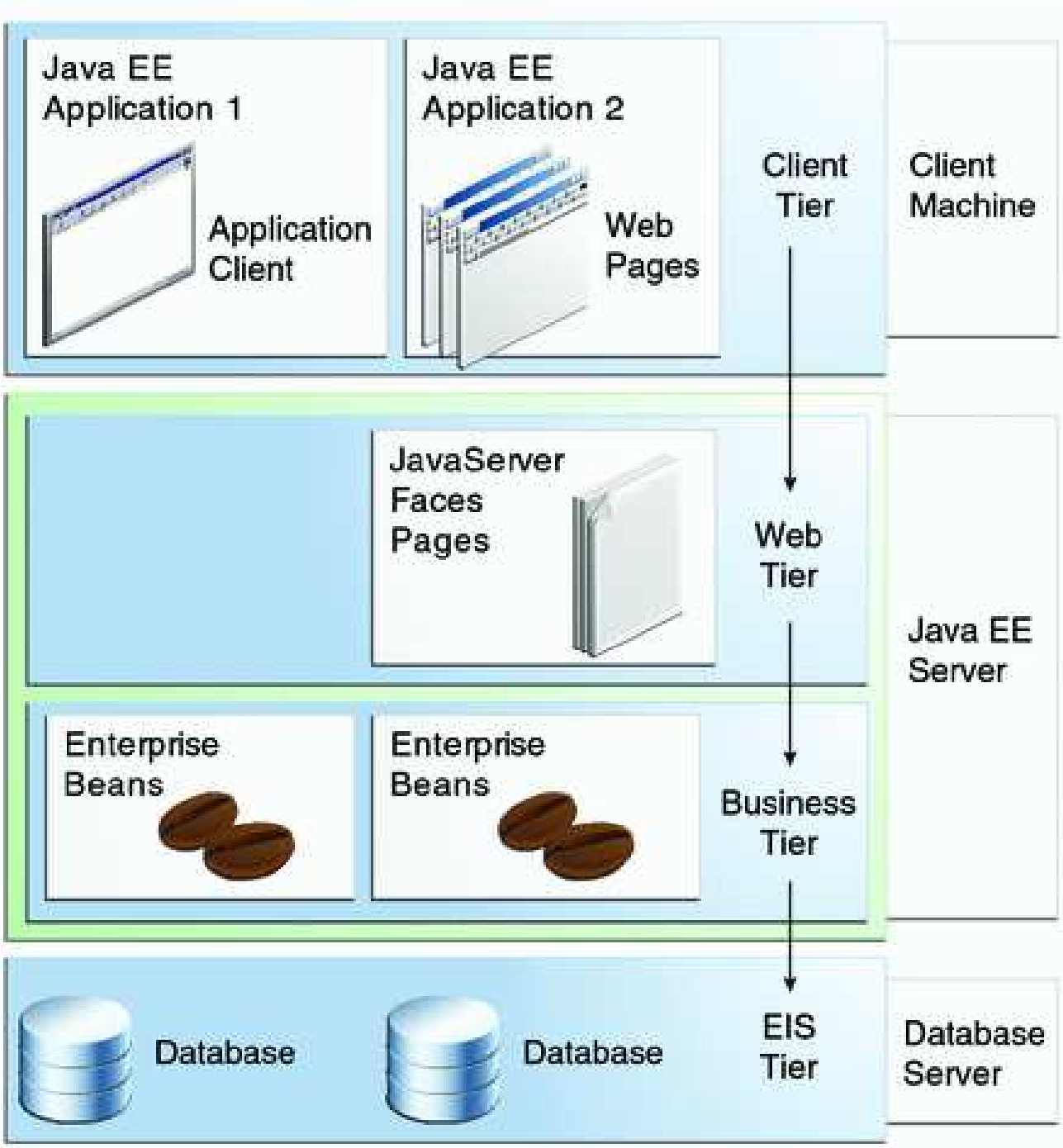
Model aplikacji Java EE 6

Aplikacja oparta ma modelu Java EE 6 jest skalowalna, dostępna i łatwa w zarządzaniu.

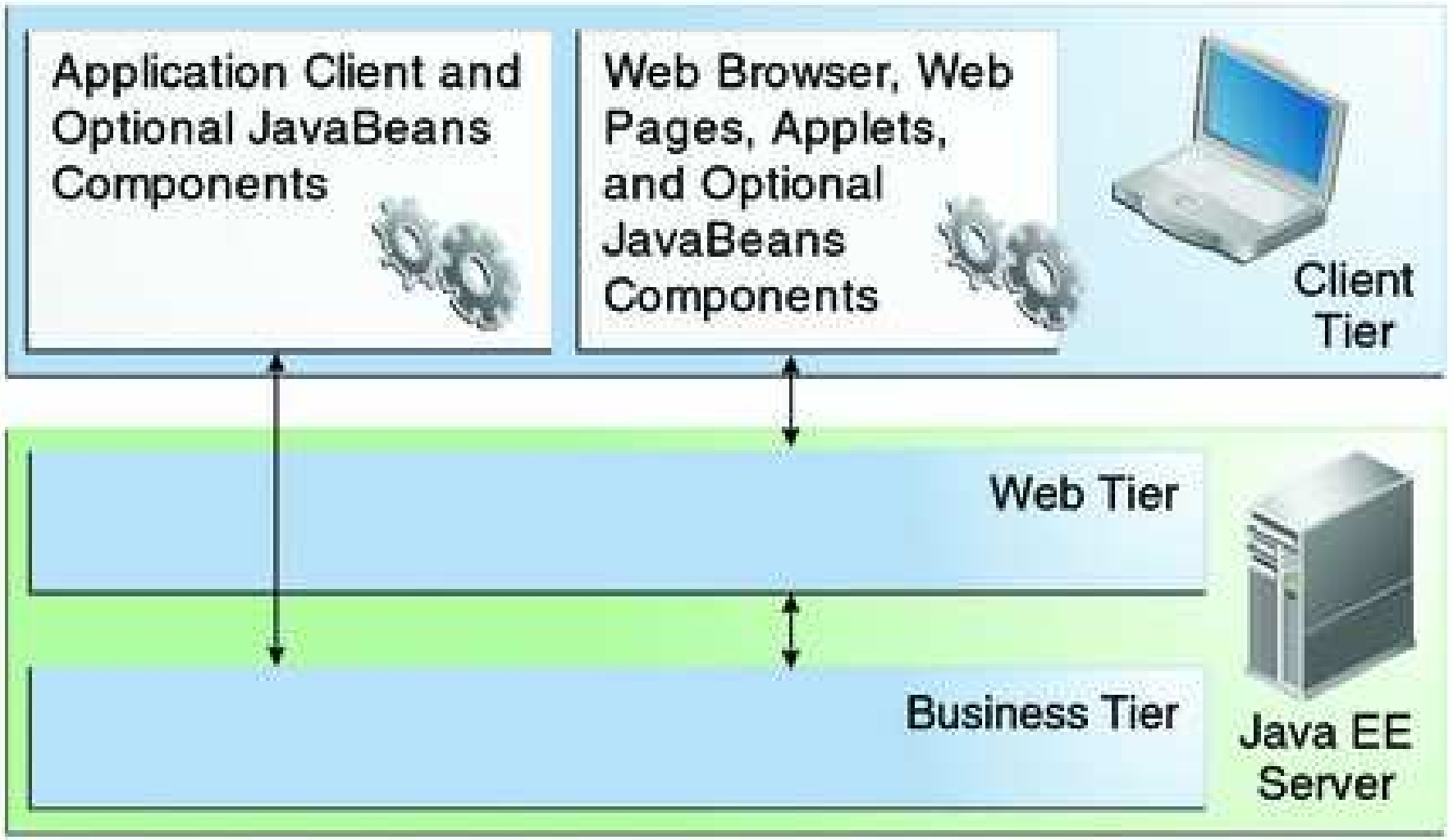
Aplikacja na platformie Java EE 6 składa się z **dwóch części**:

- logiki biznesowej i prezentacji, które są realizowane przez programistów **w postaci** komponentów wielokrotnego użytku.
- standardowe usługi systemowe świadczone przez platformę Java EE do obsługi transakcji i zarządzania stanem aplikacji, wielowątkowości, puli zasobów, i pozostałych złożonych niskopoziomowych funkcji.

Wielowarswowa architektura aplikacji EE



Komunikacja między warstwą klienta i serwerem aplikacji

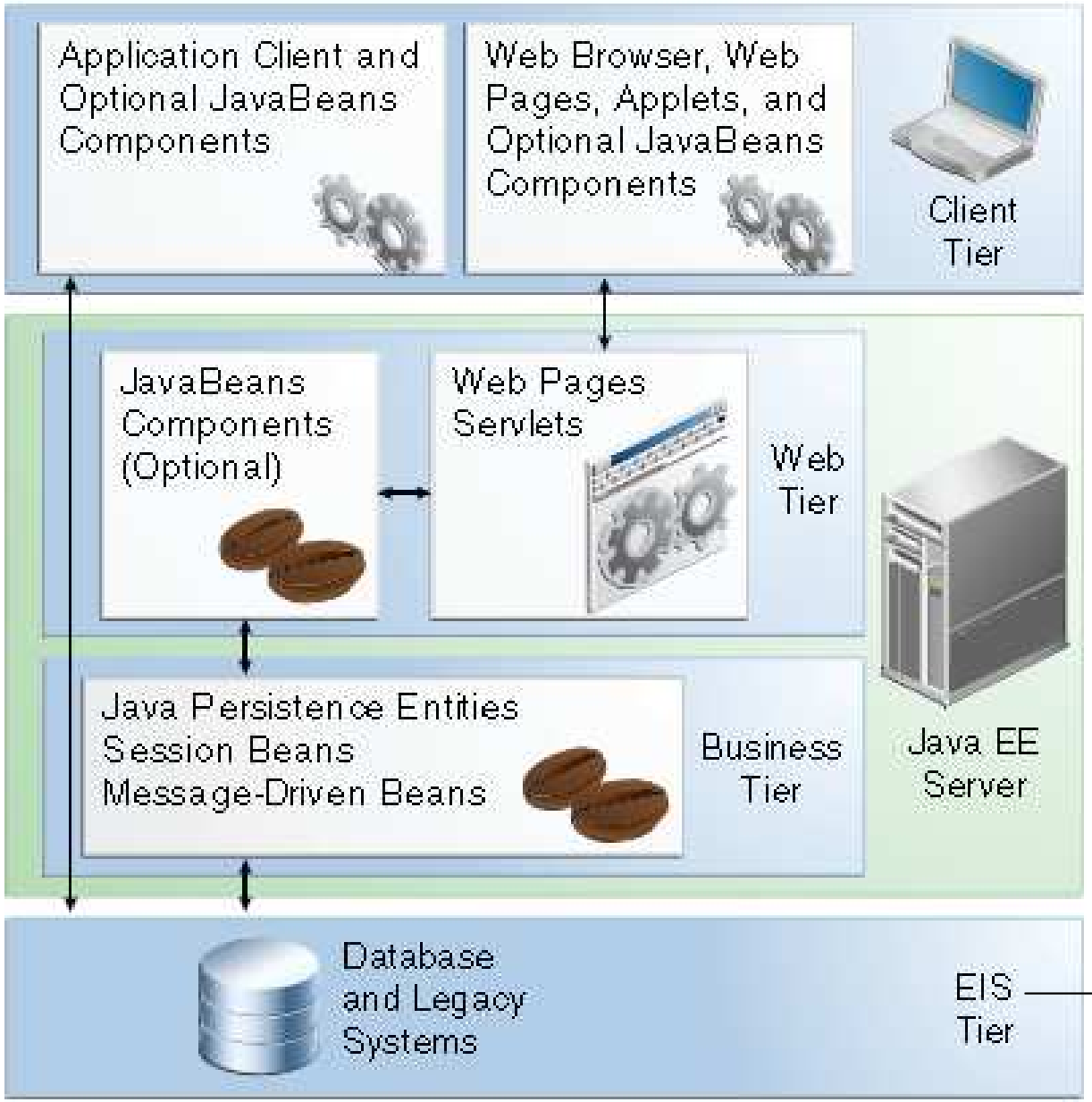


Komponentowa budowa aplikacji EE

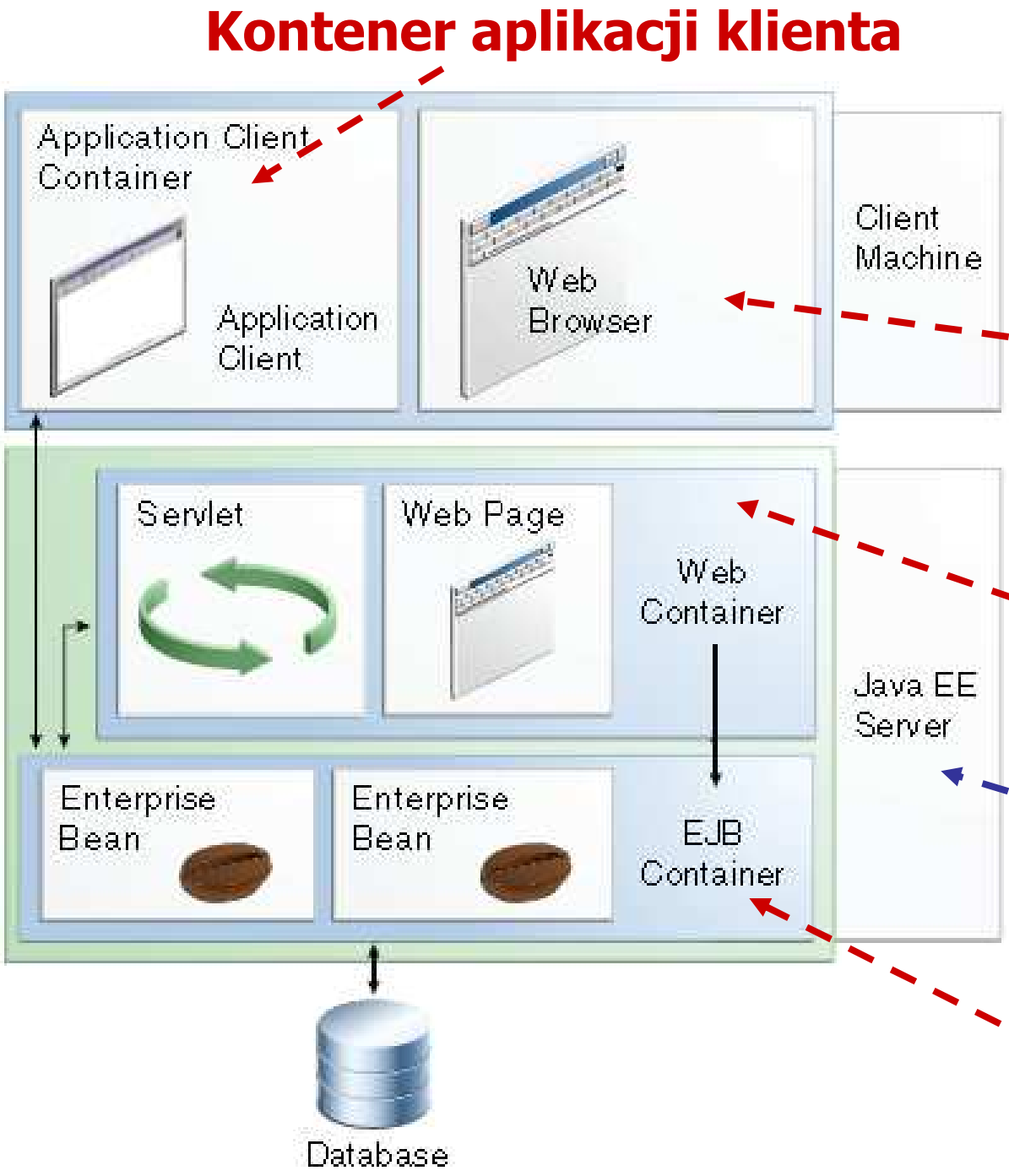
Komponent Java EE – samowystarczalna jednostka funkcjonalna oprogramowania, która składa się z powiązanych klas i plików i komunikuje się z innymi komponentami:

- **komponenty warstwy klienta** (działające na **maszynie klienta**): aplikacje klienckie (GUI oparte na pakietach AWT/Swing), aplety
- **komponenty warstwy internetowej (prezentacji)** działające na **serwerze aplikacji Java EE 6:**
  a) Java Servlet,
  b) JavaServer Pages (JSP) technology
  c) **JavaServer Faces**,
- **komponenty warstwy biznesowej**: Enterprise JavaBeans (ziarna EJB) działające na **serwerze aplikacji Java EE 6:**
  a) klasy typu Entity do obsługi trwałości (Java Persistence Entity),
  b) sesyjne ziarna EJB (Session Beans)
  c) ziarna EJB sterowane wiadomościami (Message-Driven Beans),

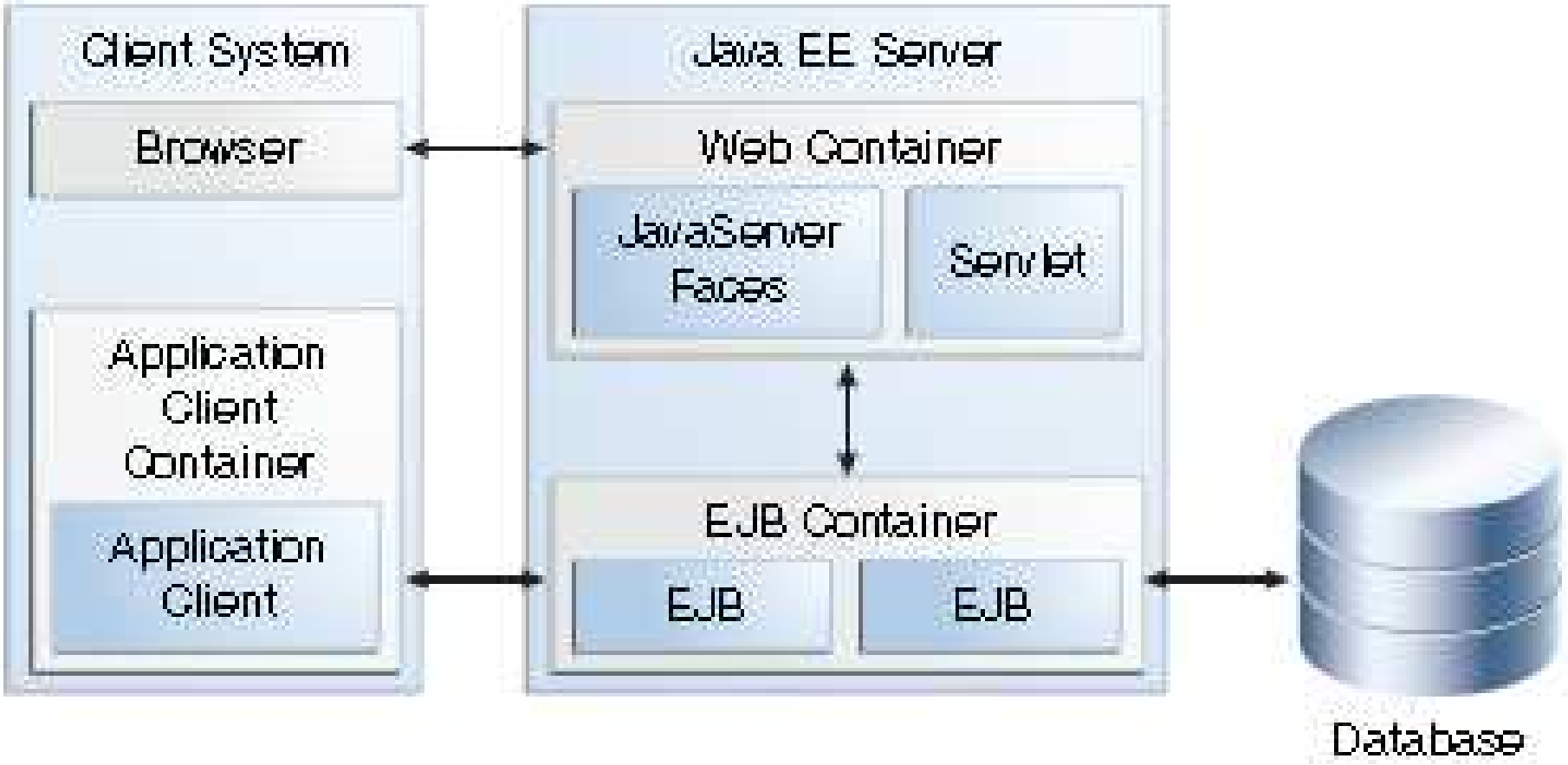
Komponenty poszczególnych warstw aplikacji EE



Kontenery aplikacji Java EE 6



Zależności między kontenerami aplikacji



(1) Usługi kontenerów

Kontenery to interfejsy między komponentami i funkcjami niskiego poziomu platformy, które wspierają komponenty.

Zanim komponent zostanie użyty, musi być:

- **przetłumaczony na kod modułu typu EE („bajtkod”) przystosowany do korzystania z usług kontenera**
- **i następnie umieszczony w swoim kontenerze w wyniku procesu „deploy”.**

(2) Usługi kontenerów

Oto niektóre z najważniejszych konfigurowalnych usług kontenerów

■ **autotryzacja - model zabezpieczeń Java EE** pozwala skonfigurować komponent internetowy lub biznesowy tak, że zasoby systemowe są dostępne tylko dla autoryzowanych użytkowników.

■ **niepodzielność tranzakcji - model tranzakcji Java EE** pozwala określić relacje między metodami, które składają się na pojedynczą transakcję, tak aby wszystkie metody w jednej transakcji były traktowane są jako całość.

■ **usługi wyszukiwań JNDI** (**Java Naming and Directory Interface API**) zapewniają jednolity interfejs do wielu nazw i katalogowania usług, tak aby komponenty aplikacji mogły uzyskać dostęp do tych usług.

■ **zdalne wywołania metod - model zdalnych połączeń Java EE** zarządza niskiego poziomu komunikacją między komponentami aplikacji klienckiej (warstwą klienta) i komponentami biznesowymi. Klient wywołuje metody komponentu biznesowego tak, jakby istniał na tej samej maszynie wirtualnej.

Wniosek:

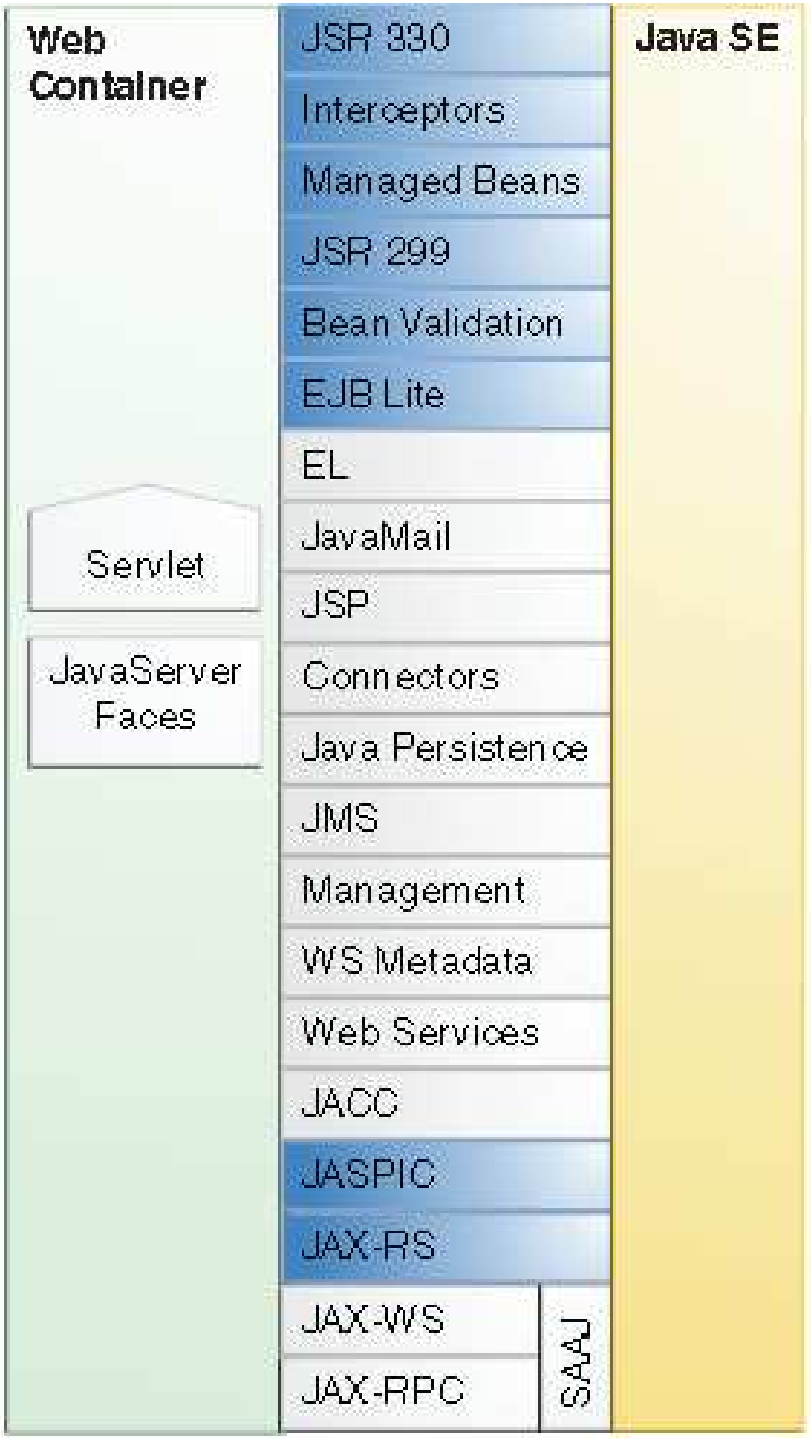
Dzięki konfigurowaniu usług kontenera te same komponenty mogą być różnie dostosowane do środowiska np. w dostępie do bazy danych.

Kontenery wykonują usługi, których nie można konfigurować:

- cykl życia komponentów typu Servlet i biznesowych
- zarządzanie pulą połączeń do baz danych,
- trwałość danych
- dostęp do API platformy EE

Zadania poszczególnych kontenerów

- **Serwer Java EE:** uruchomieniowa część produktu Java EE. Java EE serwer udostępnia komponenty EJB i internetowe.
- **Kontener EJB:** Zarządza wykonaniem komponentów EJB aplikacji Java EE. Komponenty EJB oraz ich kontener uruchamiane są na serwerze Java EE.
- **Kontener internetowy:** Zarządza wykonaniem stron internetowych, serwletów, i niektórych komponentów EJB dla aplikacji Java EE. Komponenty typu Web i ich kontener uruchamiane są na serwerze Java EE.
- **Kontener aplikacji klienckiej:** Zarządza wykonaniem składników klienckich aplikacji. Komponenty klienta i jego kontener działają na maszynie klienta.
- **Kontener apletu:** Zarządza wykonywaniem apletów. Składa się z przeglądarki internetowej i dodatku Java uruchomionych na maszynie klienta.



API (Application Programming Interface)

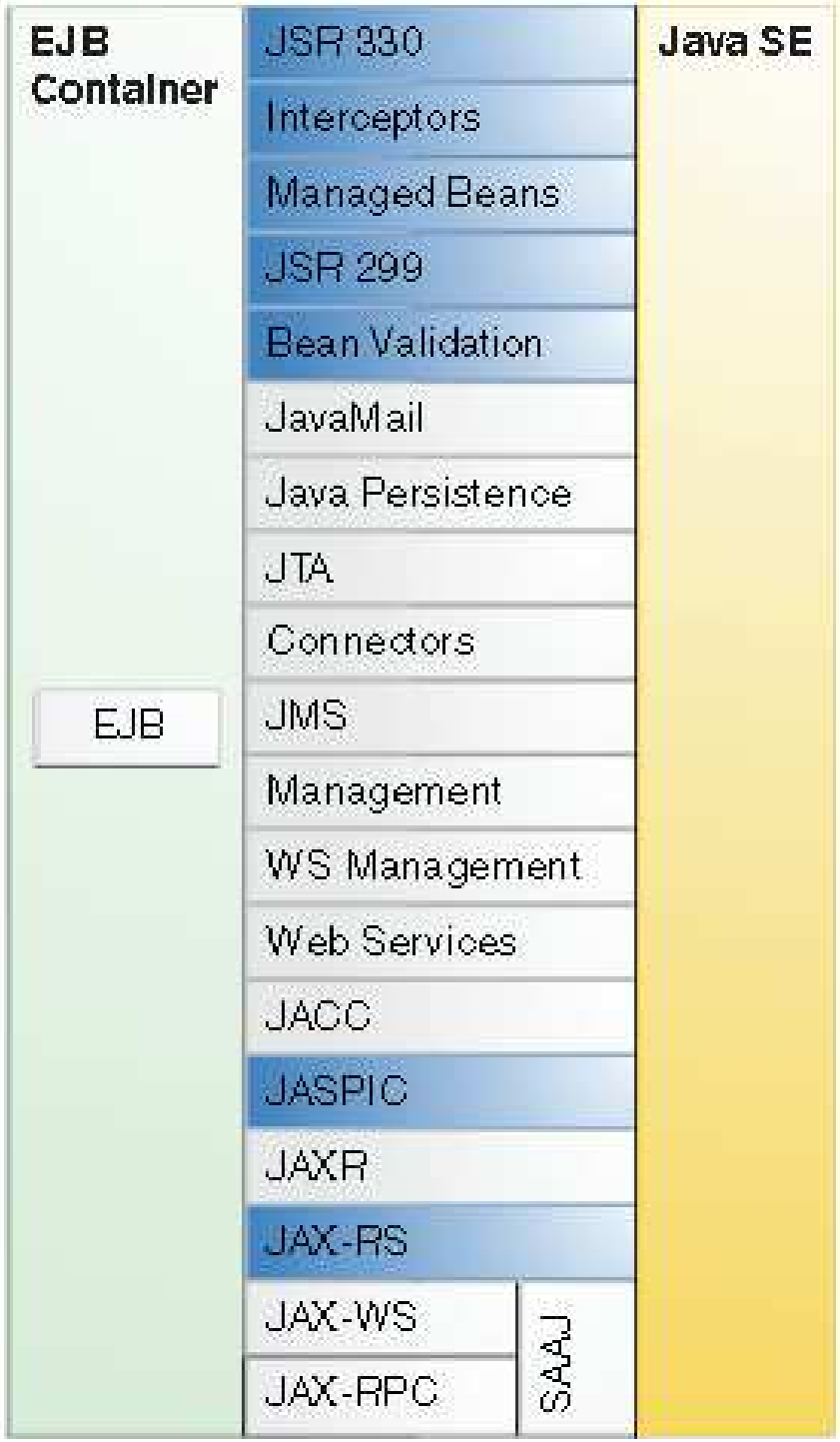
-

Interfejs programowania aplikacji kontenera internetowego

Nowe w w Java EE 6

JSRs: Java Specification Requests

http://jcp.org/en/jsr/all



API (Application Programming Interface)

-

Interfejs programowania aplikacji kontenera EJB

Nowe w w Java EE 6

API (Application Programming Interface)

-

Interfejs programowania aplikacji kontenera aplikacji klienckiej

Nowe w w Java EE 6

Tworzenie aplikacji Java EE 6

- **Build – tworzenie modułów Javy**

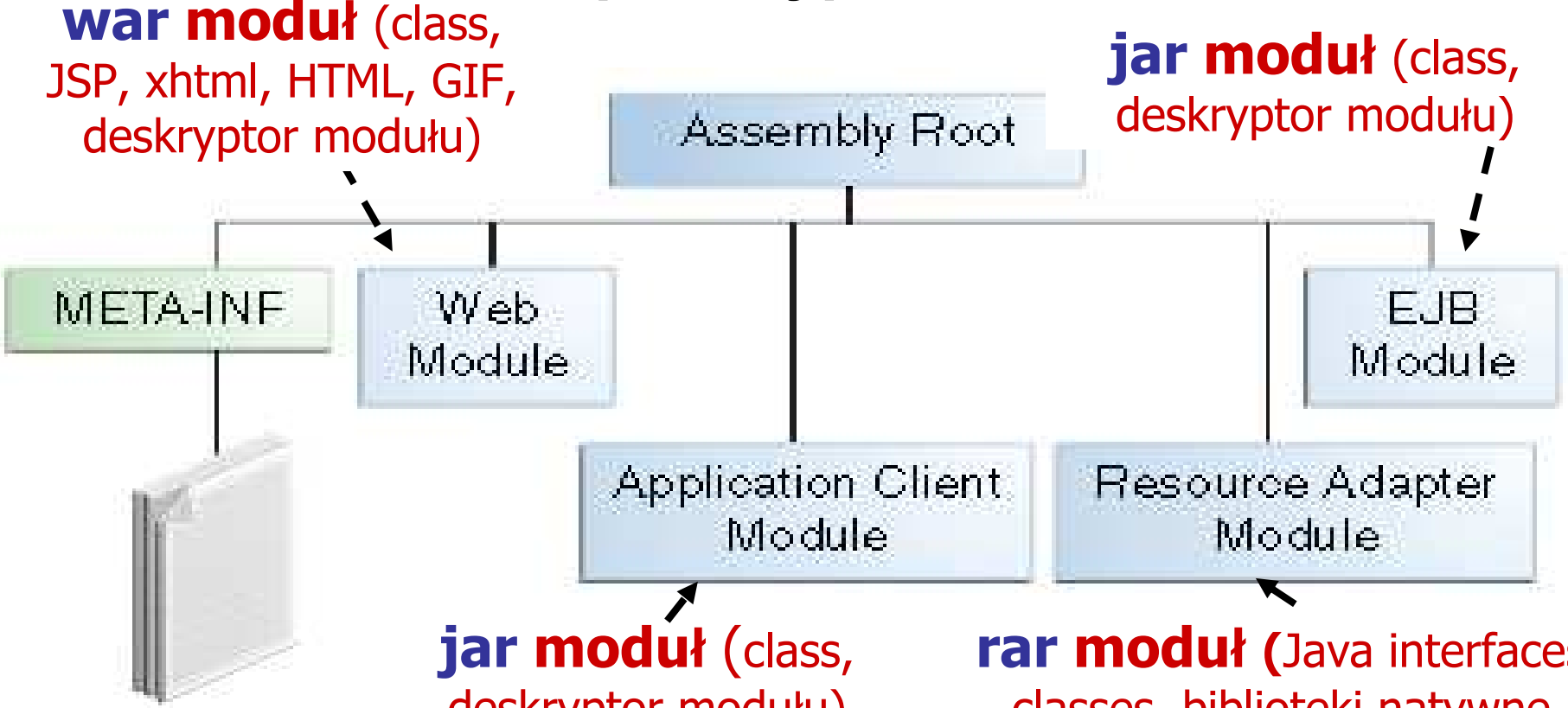
  a) tworzenie funkcjonalnych komponentów Javy (EJB, JSP page, servlet, applet, etc.)

  b) tworzenie opcjonalnego deskryptora opisującego zawartość modułu

- **Deploy: łączenie modułów z kontenerami**

  specyfikacja użytkowników oraz nazw lokalnych baz danych

Spakowana struktura aplikacji Java EE 6 – pliku typu EAR



Role uczestników w procesie tworzenia aplikacji Java EE 6

Role osób w procesie tworzenia technologii EE

Dostawca produktu Java EE

Dostawca produktu Java EE projektuje i implementuje dostępną do nabycia platformę Java EE i inne elementy określone w specyfikacji Java EE. Dostawcy wyrobów są zazwyczaj producentami serwerów aplikacji, które implementują zgodnie ze specyfikacją platformy Java EE 6.

Dostawca narzędzi

Dostawcą narzędzi jest firma lub osoba, która tworzy narzędzia do rozwoju, kompilacji i pakowania komponentów używane przez dostawców komponentów, produktów kompilacji i rozmieszczania komponentów (‘deploying’).

Dostawca komponentów aplikacji

Forma lub osoba tworzy komponenty internetowe, biznesowe (EJB), aplety, aplikacjie klienckie używane w aplikacji Java EE

Role osób biorących udział w tworzeniu aplikacji Java EE 6

Programista komponenetów EJB

Tworzy komponenty EJB

- pisze i kompiluje kod źródłowy
- specyfikuje deskryptor procesu wdrożenia „deployment” (opcjonalnie)
- pakuje pliki .class i deskryptor procesu „deployment” do pliku typu EJB jar.

Programista komponentów internetowych

Wykonuje komponenty interenetowe i pakuje je do postaci war

- pisze i kompiluje kod źródłowy komponentów typu servlet
- pisze pliki JavaServer Faces, JSP, i HTML
- specyfikuje deskryptor procesu wdrożenia „deployment” (opcjonalnie)
- pakuje pliki .class, .jsp, and.html i deskryptor procesu wdrożenia „deployment” do pliku typu WAR

Programista aplikacji klikenckich

Wykonuje szereg zadań w celu wykonania komponentów warstwy klienckiej

- pisze i kompiluje kod źródłowy
- specyfikuje deskryptor procesu wdrożenia „deployment” komponentów klienckich (opcjonalnie)
- pakuje pliki typu .class i deskryptor procesu wdrożenia „deployment” do pliku JAR

Role osób biorących udział w tworzeniu aplikacji Java EE 6

Kompilacja aplikacji EE

Jest to firma lub osoba, która

- pakuje pliki jar (EJB) i WAR utworzone w poprzednich fazach przez programistów komponentów do aplikacji Java EE (EAR)
- określa deskryptor wdrożenia dla aplikacji Java EE (opcjonalnie) - edytuje deskryptor wdrożenia bezpośrednio lub używa narzędzia, które prawidłowo w sposób interaktywny dodają tagi XML do deskryptora
- weryfikuje, czy zawartość pliku EAR jest dobrze zbudowana i zgodna ze specyfikacją Java EE

Role osób biorących udział w tworzeniu aplikacji Java EE 6

Wdrażanie i administrowanie aplikacją

Jest to firma lub osoba, która ma następujące obowiązki:

- **konfiguracja** aplikacji Java EE lub komponentów odpowiednio do środowiska operacyjnego (ustawienia zabezpieczeń, przypisanie atrybutów transakcji, wdrażanie klas i interfesów)
- **weryfikacja** zawartości plików EAR, JAR i / lub WAR, czy są dobrze zbudowane i spełniają specyfikację Java EE
- **wdrażanie** (instalacja) aplikacji Java EE lub komponentów na serwerze aplikacji Java EE
- **zarządzanie** infrastrukturą sieciową, z której korzystają komponenty Java EE, nadzorowanie środowiska wykonawczego
- **specyfikacja**: kontroli transakcji, atrybutów bezpieczeństwa i połączeń do baz danych

Dodatek do wykładu

API (Application Programming Interface)

Java EE 6

-

Wprowadzenie do interfejsu programowania aplikacji Java EE 6

(1) Charakterystyka komponentów EJB

1) Enterprise JavaBeans (EJB) – komponent, składnik, lub ziarno typu Enterprise (EJB), jest klasą zawierającą pola i metody do wdrożenia modułów logiki biznesowej.
2) Ziarno EJB może realizować samodzielnie lub w powiązaniu z innymi ziarnami EJB logikę biznesową na serwerze aplikacji Java EE 6.
3) Ziarna EJB:
   - **ziarna sesyjne** (Session Bean) reprezentują przejściowe połączenie z aplikacją klienta. Kiedy klient kończy wykonywanie, ziarno sesji i jego dane zostają usunięte.
   - **ziarno sterowane wiadomościami** (Message-Driven Beans). łączy cechy ziarna sesji i słuchacza wiadomości, pozwalając komponentowi biznesowemu otrzymywać wiadomości asynchronicznie. Często są to wiadomości typu Java Message Service (JMS).

(2) Charakterystyka komponentów EJB

Nowe elementy technologii Java EE 6:

- możliwość spakowania lokalnych ziaren EJB w plikach typu WAR
- Ziarna sesyjne typu Singleton, które zapewniają łatwy dostęp do wspólnego stanu danych
- lekki podzbiór funkcjonalności Enterprise JavaBeans (EJB Lite), które mogą być świadczone: jak Java EE Web Profile.

Platforma Java EE 6 wymaga Enterprise JavaBeans 3.1 i Interceptors 1.1.

Technologia Java Servlet

Technologia Java Servlet pozwala zdefiniować klasy serwletu typu HTTP. Klasa reprezentująca Servlet rozszerza możliwości serwerów aplikacji w obsłudze modelu programowania typu żądanie-odpowiedź. Chociaż serwlety mogą reagować na wszelkiego rodzaju żądania, są powszechnie stosowane w celu rozszerzenia aplikacji obsługiwanych przez serwery WWW.
W Java EE 6, nowe funkcje technologii Java Servlet obejmują:

- Wsparcie adnotacji
- Wsparcie mechanizów asynchronicznych
- Łatwość konfiguracji
- Ulepszenia istniejących API
- możliwość używania w postaci komponentu typu plugg-in (wsparcie wileoużywalności)

Platforma Java EE 6 wymaga technologii Servlet 3.0.

Technologia JavaServer Faces (JSF) (1)

Technologia JavaServer Faces jest środowiskiem do tworzenia interfejsu użytkownika w aplikacjach internetowych.

Głównymi składnikami technologii JavaServer Faces są:
- framework do budowy GUI zbudowanego z komponentów.
- elastyczny model do generowania elementów strony www w postaci różnych rodzajów znaczników HTML lub innych języków i technologii opartych na znacznikach. Obiekt typu Renderer generuje znaczniki do renderowania komponentu i konwertuje dane przechowywane w obiektach modelu do typów, które mogą być reprezentowane w widoku.
- standard RenderKit do generowania znaczników HTML/4.01.

Następujące funkcje wspierają komponenty GUI:
- walidacja wejściowa
- obsługa zdarzeń
- konwersja danych między obiektami modelu i komponentów
- tworzenie modelu obiektów typu Managed
- konfiguracja nawigacji strony
- język wyrażeń (EL – Expression Language)

Technologia JavaServer Faces (JSF) (2)

Wszystkie te funkcje są dostępne za pomocą standardowego API Javy i plików konfiguracyjnych typu XML.

W platformie Java EE 6, nowe funkcje JavaServer Faces obejmują:

- korzystanie z adnotacji zamiast pliku konfiguracyjnego w celu zarządzania tzw ManagedBean oraz innymi komponentami
- zastąpienie technologii wyświetlania opartą na stronach JSP (JavaServer Pages) technologią Facelets opartą na plikach XHTM
- wsparcie Ajax
- używanie komponentów kompozytowych
- niejawna nawigacja

Platforma Java EE 6 wymaga JavaServer Faces 2.0 i języka wyrażeń EL 2.2.

Technologia JavaServer Pages (JSP)

Technologia JavaServer Pages (JSP) pozwala umieścić fragmenty kodu serwletu bezpośrednio w dokumencie tekstowym. Strona JSP jest dokumentem tekstowym, który zawiera dwa rodzaje tekstu:

- dane statyczne, które mogą być wyrażone w dowolnej formie tekstowej np. HTML lub XML
- elementy JSP, które określają, w jaki sposób konstruuje dynamiczną zawartość strony
  Więcej informacji na temat technologii JSP, w tutorialu Java EE 5: http://docs.oracle.com/javaee/5/tutorial/doc/.

Platforma Java EE 6 wymaga JavaServer Pages 2.2 zapewniających kompatybilność z poprzednimi wersjami, ale zaleca się w nowych aplikacjach stosowanie technologii Facelets jako technologii wyświetlania.

JavaServer Pages Standard Tag Library (JSTL)

JavaServer Pages Standard Tag Library (JSTL) hermetyzuje podstawowe funkcje wspólne dla wielu aplikacji JSP. Zamiast mieszania tagów od licznych dostawców w swoich aplikacjach JSP, należy użyć jednego, standardowego zestawu znaczników. Taka standaryzacja pozwala wdrażać aplikacje w dowolnym kontenerze JSP obsługującym JSTL i sprawia, że realizacja tagów jest zoptymalizowana.

JSTL ma następujące znaczniki:

Iteratorowe, warunkowe do obsługi przepływu sterowania, do manipulowania dokumentami XML, internacjonalizacji, dostępu do bazy danych za pomocą SQL i najczęściej używanych funkcji.

Platforma Java EE 6 stosuje JSTL 1.2.

Java Persistence API (JPA)

Java Persistence API (JPA) jest oparta na standardach Javy dotyczące trwałości.

Trwałość wykorzystuje podejście ORM (Object/relational Mapping) jako pomost pomiędzy modelem obiektowym i relacyjną bazą danych.

Java Persistence API może być również używana w aplikacjach platformy Java SE, poza środowiskiem Java EE.

Java Persistence składa się z następujących obszarów:
- Java Persistence API
- języka zapytań (Java Persistence Language)
- metadane ORM

Platforma Java EE 6 wymaga Persistence API Java 2.0.

JavaTransaction API (JTA)

Java Transaction API (JTA) udostępnia standardowy interfejs dla zarządzania transakcjami.

Architektura Java EE zapewnia zatwierdzanie tranzakcji za pomocą mechanizmu (**auto commit)** oraz mechanizmu **rollbac**k do wycofania skutków transakcji.

Auto commit oznacza, że wszystkie inne aplikacje, które przeglądają dane będą mogły zobaczyć aktualizacje danych po każdej operacji odczytu lub zapisu danych.

Jeśli aplikacja wykonuje dwie odrębne operacje dostępu do bazy danych, które zależą od siebie, można używać JTA API, aby wyznaczyć, gdzie cała transakcja, zawierająca obie operacje, rozpoczyna, wycofuje, i zatwierdza wynik.

Platforma Java EE 6 wymaga Java Transaction API 1,1.

Java API for RESTfulWeb Services (JAX-RS)

Java API REST Web Services (JAX-RS) definiuje API do rozwoju usług internetowych zbudowany zgodnie ze stylem architektury opisanym przez Representational State Transfer (REST).

Aplikacja JAX-RS jest aplikacją internetową, która składa się z klas spakowanych jako serwlet razem z wymaganymi bibliotekami w pliku typu WAR.

JAX-RS API jest nowy na platformie Java EE 6.

Platforma Java EE 6 wymaga JAX-RS 1.1

Managed Beans

Managed Beans są lekkimi obiektami (POJOs) z minimalnymi wymaganiami zarządzanym przez kontener obiektów, wspierające mały zestaw podstawowych usług, takich jak wstrzyknięcia zasobów, wywołań zwrotnych cyklu życia aplikacji internetowej i przechwytywań.

Obiekty typu Managed Beans stanowią uogólnienie zarządzanych komponentów JavaServer Faces i mogą być używane w dowolnym miejscu w aplikacji Java EE, a nie tylko w modułach internetowych.

Specyfikacja Managed Beans jest częścią specyfikacji platformy Java EE 6 (JSR 316).
Managed Beans są nowymi elementami na platformie Java EE 6.

Platforma Java EE 6 wymaga Managed Beans 1,0.

Contexts and Dependency Injection (CDI) na platformie Java EE (JSR 299)

Contexts and Dependency Injection CDI dla platformy Java EE definiuje zestaw kontekstowych usług, świadczonych przez kontenery Java EE, które ułatwiają programistom wykorzystywać ziarna EJB wraz z technologią JavaServer Faces w aplikacjach internetowych.

Zostały one przeznaczone do użytku z obiektami sesyjnymi (stateful), jednak CDI ma również wiele szerszych zastosowań, pozwalając programistom na dużą elastyczność integracji różnych rodzajów komponentów w luźno powiązaną całość, ale w niezawodny sposób.

CDI jest nowym mechanizmem dla platformy Java EE 6. Platform Java EE 6 wymaga CDI 1,0.

Dependency Injection (DI) for Java (JSR 330)

Dependency Injection for Java definiuje standardowy zestaw adnotacji (i jeden interfejs) stosowanych w klasach, w których można ten mechanizm zastosować.

Na platformie Java EE, CDI zapewnia wstrzykiwanie zależności. W szczególności można użyć punkty wstrzykiwania DI tylko w aplikacji obsługującej CDI.

Dependency Injection for Java jest nowym mechanizmem dla platformy Java EE 6 .

Platforma Java EE 6 wymaga Dependency Injection for Java 1.0.

Bean Validation – walidacja ziaren

Specyfikacja Bean Validation definiuje model metadanych i API walidacji danych komponentów typu JavaBeans. Zamiast podziału sprawdzania poprawności danych w kilku warstwach, takich jak przeglądarki (warstwa klienta) i po stronie serwera (warstwa internetowa i biznesowa), można zdefiniować warunki walidacji w jednym miejscu i używać je w różnych warstwach.

Mechanizm Validation Bean jest nowy na platformie Java EE 6.

Platforma Java EE 6 wymaga Bean Validation 1,0.

Java Message Service (JMS) API

JavaMessage Service (JMS) API jest standardem komunikacji, który umożliwia standardowym komponentom aplikacji Java EE tworzyć, wysyłać, odbierać i czytać wiadomości.

Umożliwia komunikację rozproszoną, która jest **luźno powiązana, niezawodna i asynchroniczna.**

Platforma Java EE 6 wymaga JMS 1,1.

Java EE Connector Architecture (1)

Java EE Connector Architecture jest używana przez narzędzia dostawców i integratorów systemów w celu stworzenia adapterów zasobów obsługujących dostęp do systemów informatycznych, które mogą być podłączone do jakiegokolwiek produktu Java EE.

Adapter zasobów jest komponentem oprogramowania, który pozwala **komponentom aplikacji Java EE na dostęp i interakcję z podstawowym menedżerem zasobów (bazy danych, usług internetowych, CRM, ERP itp) w warstwe EIS (Enterprise Information System) aplikacji Java EE.** Ponieważ adapter zasobów jest specyficzny dla jego menedżera zasobów, każdy typ bazy danych ma inny adapter zasobów.

Java EE Connector Architecture (2)

Java EE Connector Architecture zapewnia również wydajnościowo zorientowaną, bezpieczną, skalowalną i opartą na wymianie komunikatów transakcyjną **integrację platformy Java EE z usługami internetowymi z istniejących EIS**, które mogą być dostępne w trybie synchronicznym albo asynchronicznym.

Istniejące aplikacje i EIS zintegrowane poprzez Java EE Connector Architecture na platformie Java EE mogą być dostępne w sieci jako usługi internetowe XML przy użyciu JAX-WS i modeli komponentów Java EE.

Zatem JAX-WS i Java EE Connector Architecture są komplementarnymi technologiami do technologii EAI (Enterprise Application Integration) oraz end-to-end integracji biznesowej.

Platforma Java EE 6 wymaga Java EE Connector Architecture 1.6.

JavaMail API

Aplikacje Java EE używaja JavaMail API do wysyłania powiadomień e-mail.

JavaMail API składa się z dwóch części:
- interfejs na poziomie aplikacji używany przez komponenty aplikacji do wysyłania poczty
- interfejs usługodawcy.

Platforma Java EE zawiera JavaMail API z usługodawcą, który umożliwia komponentom aplikacji do wysyłania poczty internetowej.

Platforma Java EE 6 wymaga JavaMail 1,4.

Java Authorization Contract for Containers (JACC)

Specyfikacja Java Authorization Contract for Containers (JACC) definiuje „umowę” między serwerem aplikacji Java EE oraz dostawcą polityki autoryzacji. Wszystkie pojemniki Java EE obsługują taką umowę.
Specyfikacja JACC definiuje klasy java.security.Permission spełniające model autoryzacji Java EE.

Specyfikacja definiuje warunki dostępu kontenera do operacji instancji tych klas definiujących uprawnienia.

Technologia definiuje semantykę dostawców polityki autoryzacji, którzy korzystają z nowych klas uprawnień do spełnienia wymogów dostępu na platformie Java EE, w tym definicji i korzystania z ról.

Platforma Java EE 6 wymaga JACC 1,4.

Java Authentication Service Provider Interface for Containers (JASPIC) (1)

Specyfikacja Java Authentication Service Provider Interface for Containers (JASPIC) definiuje interfejs dostawcy usług (SPI – Service Provider Interface), dzięki któremu dostawcy uwierzytelniania, którzy implementują mechanizmy uwierzytelniania wiadomości mogą być zintegrowani z kontenerem klienta lub serwera lub ze środowiskiem przetwarzania wiadomości.

Dostawcy uwierzytelniania zintegrowani poprzez ten interfejs działają na wiadomościach świadczonych im przez ich wywołujące kontenery.

Java Authentication Service Provider Interface for Containers (JASPIC) (2)

Dostawcy uwierzytelniania przekształcają wiadomości wychodzące tak, aby źródło każdej wiadomości mogło być potwierdzone przez kontener odbiorcy, a odbiorca wiadomości mógłby być uwierztelniony przez nadawcę wiadomości.

Dostawcy uwierzytelniania uwierzytelniają każdą przychodzącą wiadomość i dostarczaja do swoich wywołujących kontenerów wynik uwierzytelnienia odbiorcy wiadomości.
JASPIC jest nowy dla platformy Java EE 6.

Java EE 6 platforma wymaga JASPIC 1,0

Pozostałe technologie Java EE 6

- **Web Services Support**
- **XML**
- **SOAP (Simple Object Access Protocol) Transport Protocol**
- **WSDL (Web Services Description Language) Standard Format**

Java EE 6 APIs in the Java Platform, Standard Edition 6 and 7

Java Database Connectivity (JDBC) API

JavaDatabase Connectivity (JDBC) API pozwala wywołać polecenia SQL z metod języka programowania Java. Interfejs API JDBC może być zastosowany w sesyjnym ziarnie EJB przy dostępie do bazy danych.

Można również używać JDBC API z serwletu lub strony JSP przy dostępie do bazy danych bezpośrednio, bez przechodzenia przez ziarno sesyjne EJB.
JDBC API ma dwie części:
■ interfejs używany na poziomie aplikacji przez komponenty aplikacji korzystające z dostępu do bazy danych
■ Interfejs usługodawcy do instalacji sterownika JDBC do platformy Java EE

Platforma Java SE 6 wymaga JDBC 4.0.

Java EE 6 APIs in the Java Platform, Standard Edition 6 and 7

Java Naming and Directory Interface (JNDI) API

Interfejs JavaNaming i Directory (JNDI) API zapewnia aplikacjom dostęp do interfejsu Javy usług katalogowych, który umożliwia odkrywanie i wyszukiwanie danych oraz obiektów za pomocą nazw.

Opiera się na usługach takich jak LDAP **(Lightweight Directory Access Protocol),** DNS (**Domain Name System**) i NIS (**Network Information Service**).

Technologia JNDI API zapewnia aplikacji z poziomu metod wykonywanie standardowych operacji katalogowych, takich jak kojarzenie atrybutów z obiektami i wyszukiwanie obiektów za pomocą ich cech. Korzystając z JNDI, aplikacja Java EE może przechowywać i pobrać dowolny nazwany typ obiektu Java, pozwalając aplikacji Java EE współistnieć z wieloma starszymi aplikacjami i systemami.

Java EE zapewnia aplikacjom klienckim, ziarnom EJB i komponentom internetowym korzystanie z mechanizmu JNDI. Nazewnictwo środowiska pozwala komponentowi dostosować się, bez konieczności dostępu do jego kodu źródłowego lub jego zmiany. Kontener implementuje środowisko komponentu i dostarcza je do komponentu jako kontekst nazewnictwa JNDI.

Komponent Java EE może zlokalizować swój kontekst w kontekście nazewnictwie środowiska przy użyciu interfejsów JNDI.

Komponent może utworzyć obiekt **javax.naming.InitialContext** i poszukać usług lub obiektów w kontekście nazewnictwa środowiska za pomocą obiektu InitialContext pod nazwą java: **comp / env**.

Nazewnictwo środowiska komponentu jest przechowywane bezpośrednio w kontekście nazewnictwa środowiska lub w jednym z jego bezpośrednich lub pośrednich podkontekstów.

Komponent Java EE może uzyskać dostęp do obiektów za pomocą systemu nazewnictwa lub bezpośredni dostęp do obiektów zdefiniowanych przez użytkownika. Nazwy obiektów dostarczanych przez systemu - takie jak obiekty JTA UserTransaction, są przechowywane w kontekście nazewnictwa środowiska java: comp / env.

Platforma Java EE pozwala komponentom nadać nazwy obiektóm definiowanym przez użytkownika, takich jak ziarna EJB, wejściom środowiskowym, obiektom JDBC DataSource i połączeniom wiadomości.

Obiekt powinien mieć w nazwie podkontekst nazewnictwa środowiska zgodny z rodzajem obiektu. Na przykład ziarna EJB są nazywane w podkontekście java: **comp / env / ejb** i referencje typu JDBC DataSource są nazywane w podkontekście java: **comp / env / jdbc.**

Java EE 6 APIs in the Java Platform, Standard Edition 6 and 7

Java API for XML Processing (JAXP)

Java API for XML Processing (JAXP), część platformy Java SE, obsługuje przetwarzania dokumentów XML za pomocą Document ObjectModel (DOM), Simple API for XML (SAX) i Extensible Stylesheet Language Transformations (XSLT). JAXP umożliwia aplikacjom analizowanie i przetwarzanie dokumentów XML niezależnie od konkretnej implementacji przetwarzania XML.
JAXP zapewnia także obsługę przestrzeni nazw, która pozwala na pracę ze schematami, które w przeciwnym razie mogłyby mieć konflikty nazewnictwa .Projektownie jest elastyczne, ponieważ JAXP pozwala używać właściwy parser XML lub procesor XSL w aplikacji i obsługuje schemat Worldwide Web Consortium (W3C).

Informacje na temat schematu W3C są pod adresem URL: http://www.w3.org/XML/Schema.

Java EE 6 APIs in the Java Platform, Standard Edition 6 and 7

Java Authentication and Authorization Service (JAAS)

Java Authentication and Service Authorization (JAAS) zapewnia aplikacji Java EE sposób uwierzytelniania i autoryzacji określonego użytkownika lub grupy użytkowników, aby go/ich uruchomić.
JAAS jest wersją standardowego modułu Pluggable Authentication Module (PAM), która rozszerza architekturę bezpieczeństwa platformy Java wspierającej autoryzację użytkownika.

Java EE 6 APIs in the Java Platform, Standard Edition 6 and 7

- JavaBeans Activation Framework (JAF)
- Java Architecture for XML Binding (JAXB)
- SOAP with Attachments API for Java (SAAJ)
- Java API for XMLWeb Services (JAX-WS)

Narzędzia serwera GlassFish

Narzędzie	Opis
Administration Console	Narzędzie z GUI wspierające administratora. Używane do zatrzymania serwera i do zarządzania zasobami użytkownika i aplikacji
asadmin	Narzędzie uruchamiane z linii poleceń wspierające administratora. Używane do uruchamiania i zatrzymania serwera i do zarządzania zasobami użytkownika i aplikacji
appclient	Narzędzie uruchamiane z linii poleceń do połączenia kontenera aplikacji klienckiej i do wywołania pakietu aplikacji klienckiej spakowanego w pliku jar
capture-schema	Narzędzie uruchamiane z linii poleceń do ekstrakcji shematu bazy danych z bazy danych w postaci pliku używanego przez serwera do trwałości obsługiwanej przez kontener
package-appclient	Narzędzie uruchamiane z linii poleceń do pakowania bibliotek kontenera aplikacji klienckiej i pików typu JAR

Narzędzia serwera GlassFish

Narzędzie	Opis
JavaDB database	Kopia serwera Java DB (baza danych Apache Derby firmy ORACLE typu Open Source)
xjc	Narzędzie uruchamiane z linii poleceń do transformacji lub połączeniu źródła schematu XML do zbioru klas z kontekstu JAXB
schemagen	Narzędzie uruchamiane z linii poleceń do tworzenia pliku ze schematem XML
wsimport	Narzędzie uruchamiane z linii poleceń do generowania przenośnych parametrów JAX-WS dla danego pliku WSDL. Po generacji, te parametry mogą być spakowane w pliku WAR, zawierającym WSDL oraz schemat dokumentu z implementacją typu endpoint i potem może być wdrożony
wsgen	Narzędzie uruchamiane z linii poleceń do czytania klasy typu endpoint usługi internetowej i generowania wszystkich przenośnych parametrów JAX-WS usługi internetowej do jej wdrożenia i wywołania