



<http://www.reksoft.com>

Владимир Сергеевич Бегларян

# Java Enterprise Edition



<http://www.oracle.com/technetwork/java/javaee>

## Пример готового простого приложения как старт для работы на Java EE.

[https://netbeans.org/kb/docs/javaee/javaee-entapp-ejb\\_ru.html](https://netbeans.org/kb/docs/javaee/javaee-entapp-ejb_ru.html)

- **Веб-уровень.** Веб-уровень содержит логику представления приложения и запускается на сервере Java EE. В приложении NewsApp веб-уровень представлен веб-модулем и содержит сервлеты, через которые осуществляется доступ к бизнес-логике в модуле EJB.
- **Бизнес-уровень.** Приложения бизнес-уровня также выполняются на сервере Java EE и содержат бизнес-логику приложения. В приложении NewsApp бизнес-уровень представлен модулем EJB. Модуль EJB содержит код для обработки запросов от клиентов веб-уровня и для управления транзакциями и способами сохранения объектов в базе данных.
- **EIS-уровень.** EIS-уровень - это надежный уровень хранения приложения. В приложении NewsApp этот уровень представлен базой данных для сохранения сообщений.

Версия	Полное имя	Дата публикации
1.0	Java 2 Platform Enterprise Edition, v 1.0	декабрь 1999
1.2	Java 2 Platform Enterprise Edition, v 1.2	2000
1.2.1	Java 2 Platform Enterprise Edition, v 1.2.1	23 мая 2000
1.3	Java 2 Platform Enterprise Edition, v 1.3	24 сентября 2001
1.4	Java 2 Platform Enterprise Edition, v 1.4	24 ноября 2003
5.0	Java Platform, Enterprise Edition, v 5	11 мая 2006
6.0	Java Platform, Enterprise Edition, v 6	6 декабря 2009

# Зачем нужна Java EE?

Интеграция с  
другими сервисами

Скорость разработки и  
развития

БД, транзакции,  
синхронизация

Безопасность

Бизнес - логика

Поддержка  
развёртывания,  
конфигурирования  
, мониторинга

Производительность  
масштабируемость

Поддержка  
Web

## Java Platform, Enterprise Edition

**Java Platform, Enterprise Edition** or **Java EE** is Oracle's enterprise Java computing platform. The platform provides an API and runtime environment for developing and running enterprise software, including network and web services, and other large-scale, multi-tiered, scalable, reliable, and secure network applications. Java EE extends the Java Platform, Standard Edition (Java SE), providing an API for object-relational mapping, distributed and multi-tier architectures, and web services. The platform incorporates a design based largely on modular components running on an application server. Software for Java EE is primarily developed in the Java programming language. The platform emphasizes **convention over configuration** and **annotations for configuration**.

```
import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Address implements Serializable {
    @Id
    protected long id;
    String street1;
    String street2;
    String city;
    String province;
    ...
}
```

**package logic;**

**import java.util.Set;**

**import java.util.HashSet;**

**public class Bus {**

**private Long id;**

**private String number;**

**public Bus() { }**

**public void setId(Long id) {**

**this.id = id;**

**}**

**public void setNumber(String number) {**

**this.number = number;**

**}**

**public Long getId() {**

**return id;**

**}**

**public String getNumber() {**

**return number;**

**}**

**}**

```
<hibernate-mapping>
  <class name=«logic.Bus» table=«busses»>
    <id column=«bus_id» name=«id» type=«java.lang.Long»>
      <generator class=«increment»/>
    </id>
    <property column=«number» name=«number» type=«java.lang.String»/>

    <set name=«drivers» table=«busDriver» lazy=«false»>
      <key column=«bus_id»/>
      <many-to-many column=«driver_id» class=«logic.Driver»/>
    </set>

  </class>
</hibernate-mapping>
```



<http://www.ashep.org/2013/что-такое-dependency-injection/>

```
public interface IOnlineBrokerageService {  
    String[] getStockSymbols();  
    double getAskingPrice(String stockSymbol);  
    double getOfferPrice(String stockSymbol);  
    void putBuyOrder(String stockSymbol, int shares, double bidPrice);  
    void putSellOrder(String stockSymbol, int shares, double offerPrice);  
}
```

```
public interface IStockAnalysisService {  
    double getEstimatedValue(String stockSymbol);  
}
```

```
public interface IAutomatedStockTrader { void executeTrades(); }
```

## Декларативность: dependency injection

```
public class VerySimpleStockTraderImpl implements IAutomatedStockTrader {  
private IStockAnalysisService analysisService = new  
StockAnalysisServiceImpl();
```

```
private IOnlineBrokerageService brokerageService = new  
NewYorkStockExchangeBrokerageServiceImpl();
```

```
public void executeTrades() { ... }  
}
```

```
public class MyApplication {  
public static void main(String[] args) {  
IAutomatedStockTrader stockTrader = new VerySimpleStockTraderImpl();  
stockTrader.executeTrades();  
}  
}
```

```
<contract id="IAutomatedStockTrader">  
<implementation>VerySimpleStockTraderImpl</implementation>  
</contract>
```

```
<contract id="IStockAnalysisService" singleton="true">  
<implementation>StockAnalysisServiceImpl</implementation>  
</contract>
```

```
<contract id="IOnlineBrokerageService" singleton="true">  
<implementation> NewYorkStockExchangeBrokerageServiceImpl  
</implementation>  
</contract>
```

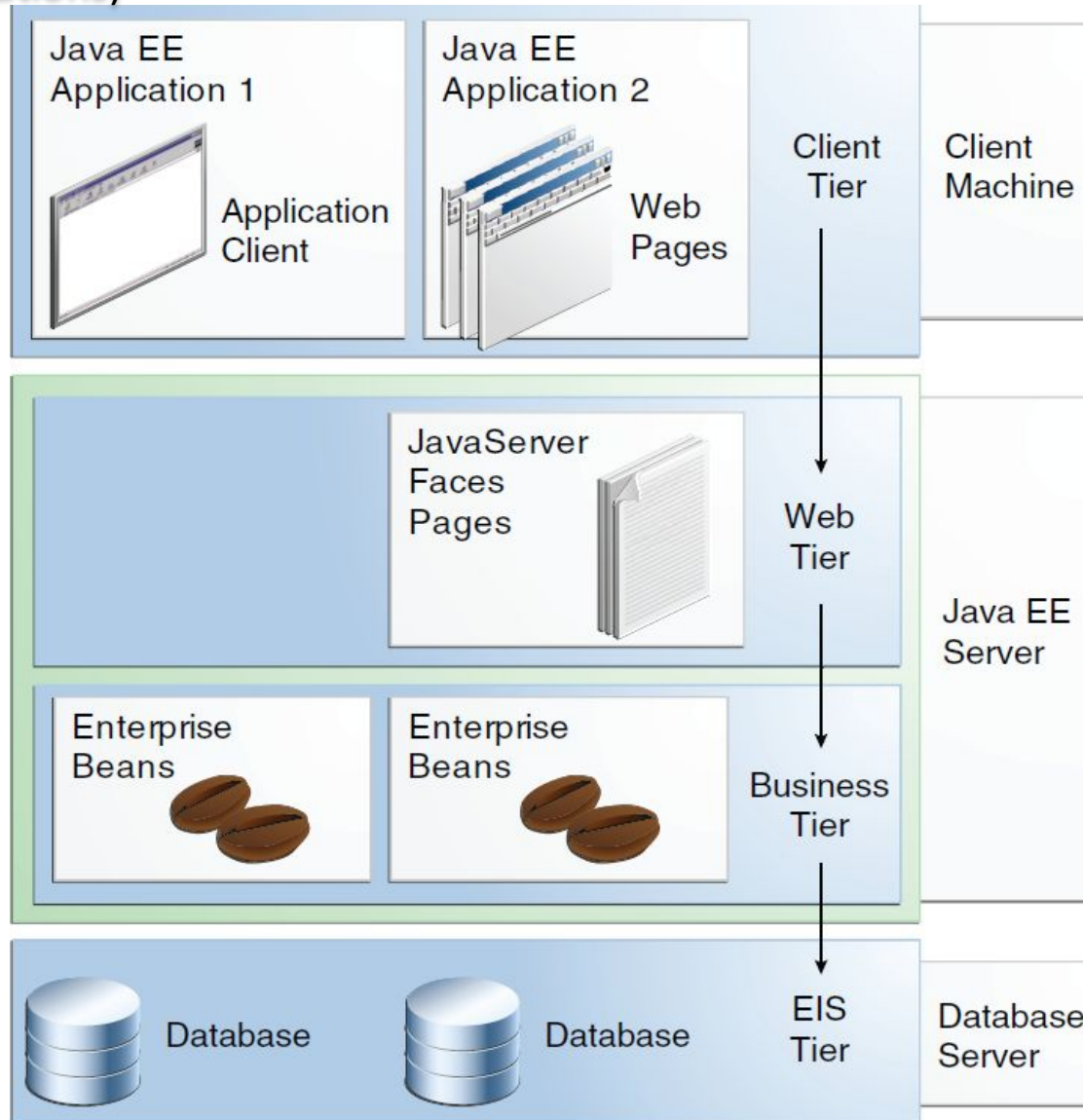
## Декларативность: dependency injection

```
public class VerySimpleStockTraderImpl implements IAutomatedStockTrader {  
    private IStockAnalysisService analysisService;  
    private IOnlineBrokerageService brokerageService;
```

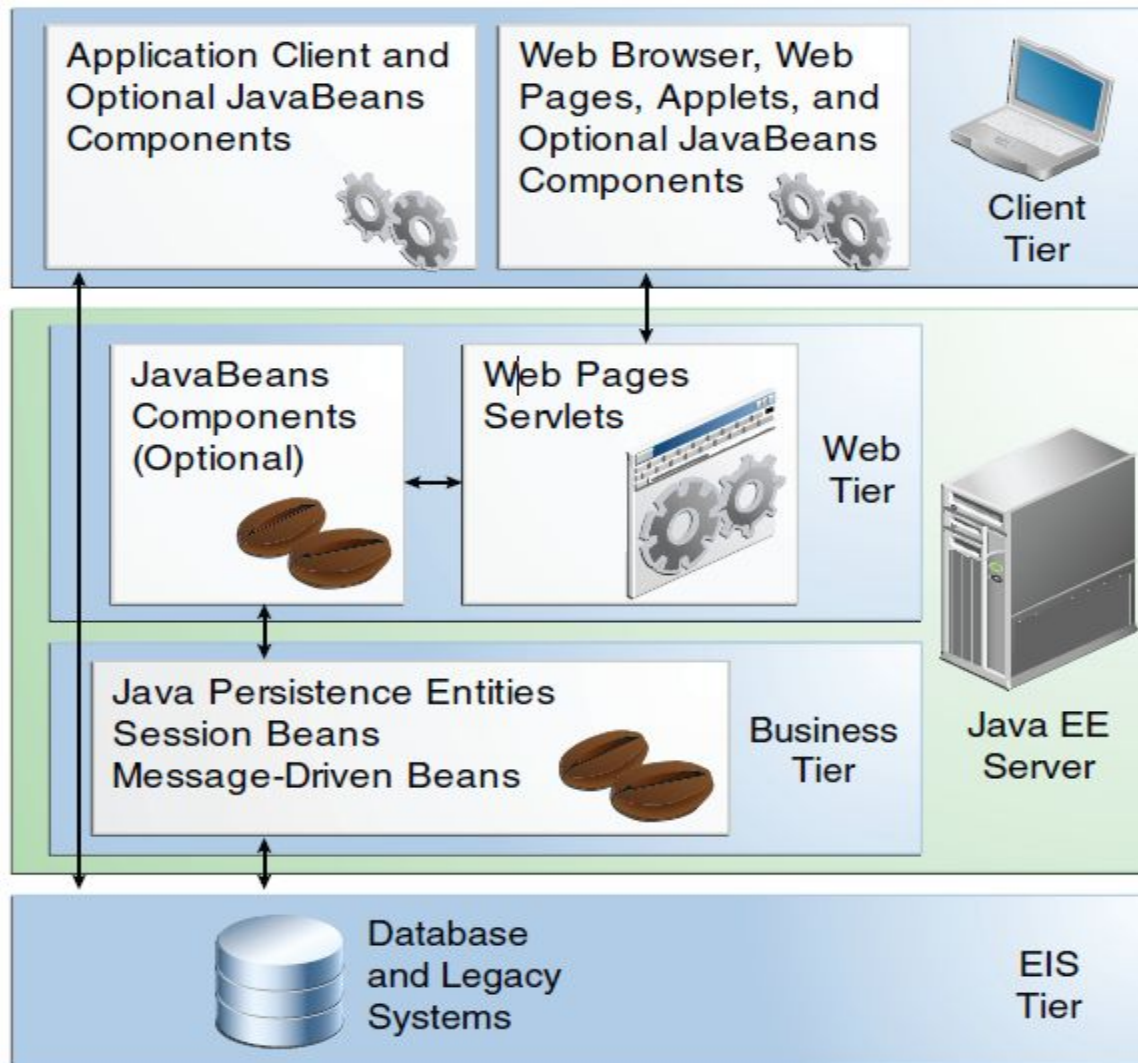
```
    public VerySimpleStockTraderImpl(  
        IStockAnalysisService analysisService, IOnlineBrokerageService brokerageService) {  
        this.analysisService = analysisService;  
        this.brokerageService = brokerageService;  
    }  
    public void executeTrades() { ... }  
}
```

```
public class MyApplication {  
    public static void main(String[] args) {  
        IAutomatedStockTrader stockTrader =  
            (IAutomatedStockTrader) DependencyManager.create(IAutomatedStockTrader.class);  
        stockTrader.executeTrades();  
    }  
}
```

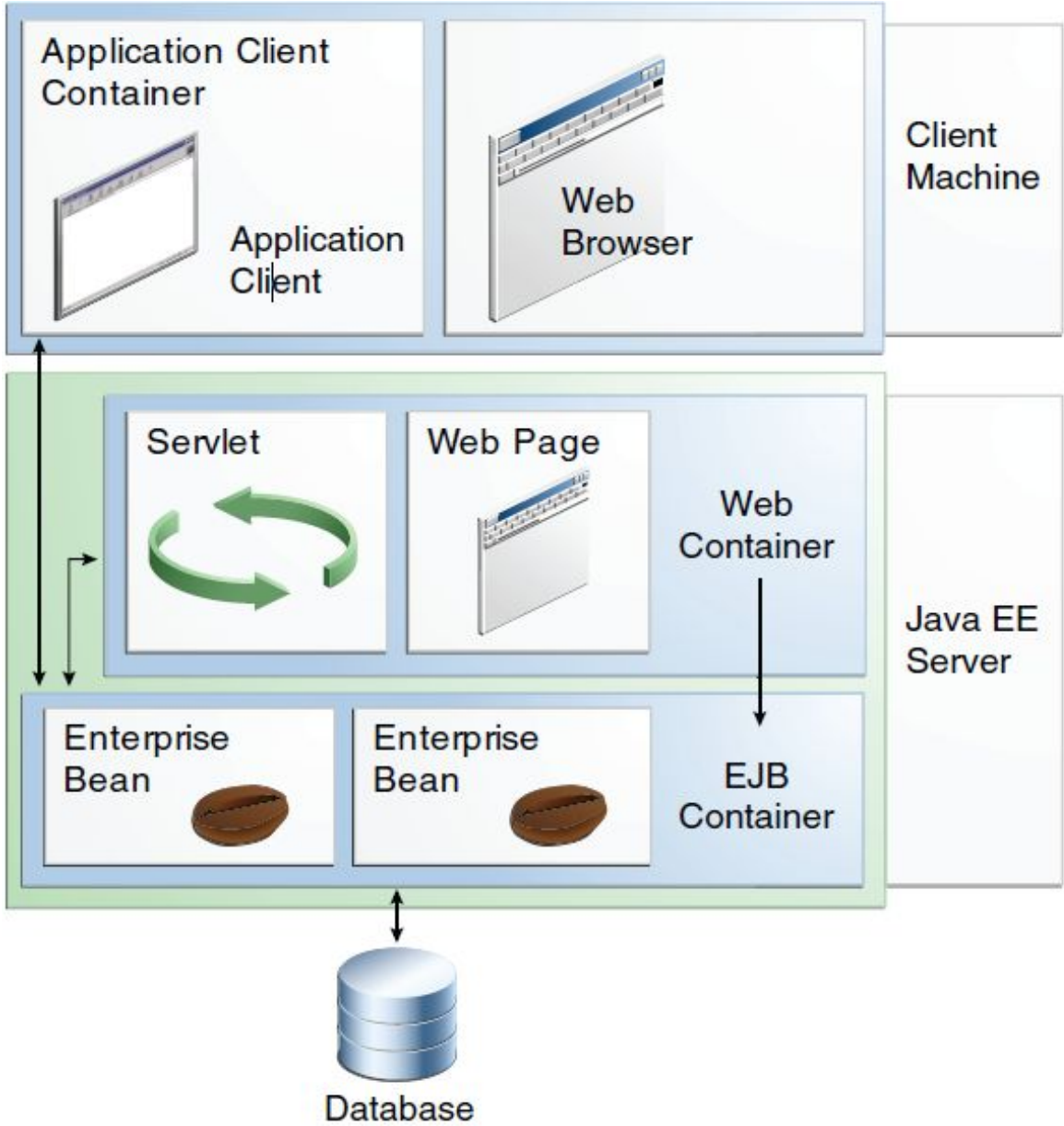
# Многоуровневая архитектура приложений (Multitiered Applications)



# Основные компоненты в архитектуре JEE приложения



# JEE Server и КОНТЕЙНЕРЫ (containers)



# Container сервис

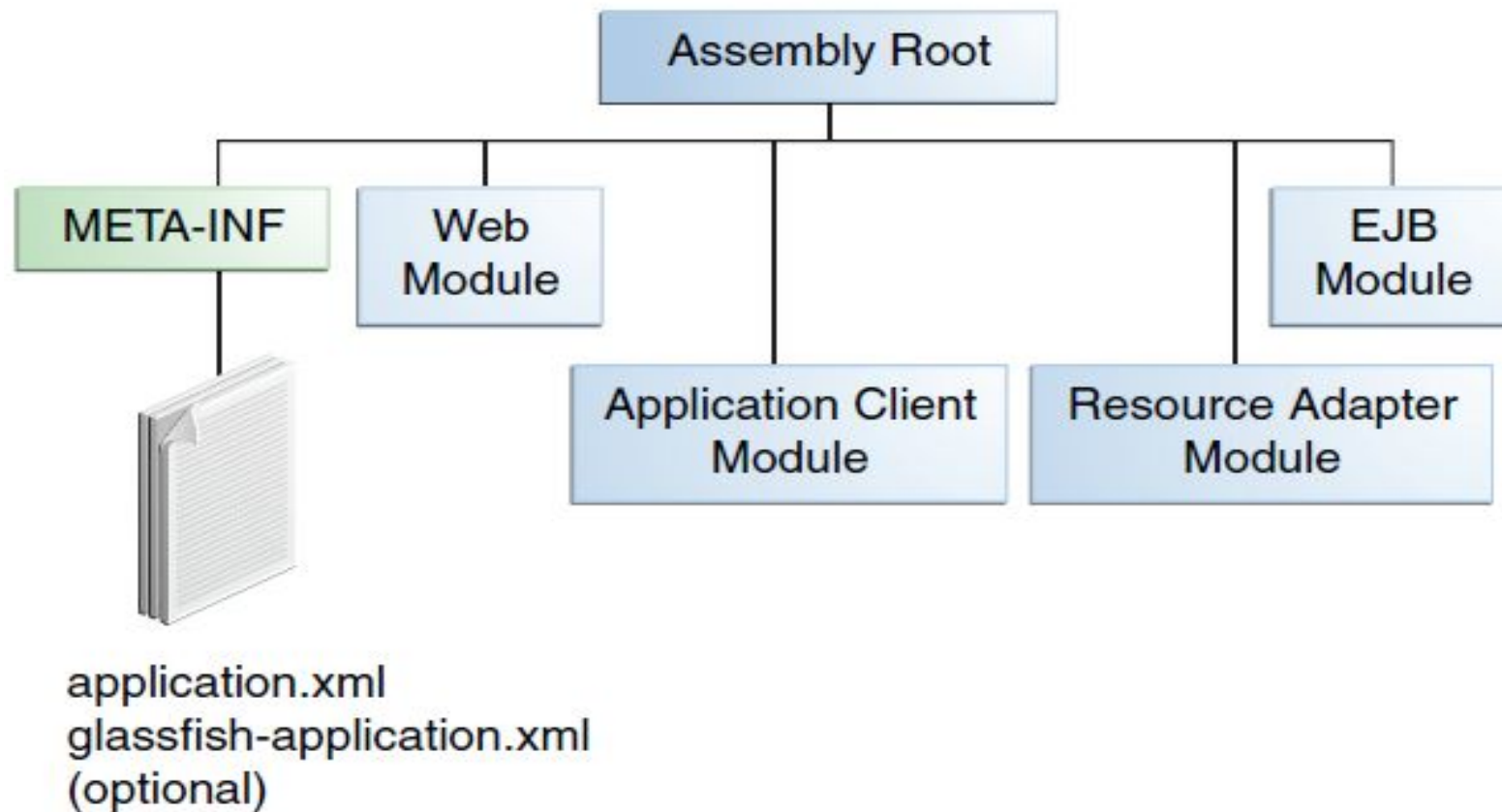
- The Java EE security model lets you configure a web component or enterprise bean so that system resources are accessed only by authorized users.
- The Java EE transaction model lets you specify relationships among methods that make up a single transaction so that all methods in one transaction are treated as a single unit.
- JNDI lookup services provide a unified interface to multiple naming and directory services in the enterprise so that application components can access these services.
- The Java EE remote connectivity model manages low-level communications between clients and enterprise beans. After an enterprise bean is created, a client invokes methods on it as if it were in the same virtual machine.



## Java App Server, сертифицированные для Java EE 6

- GlassFish server Open Source Edition
- Oracle GlassFish Server
- Oracle WebLogic Server
- JBoss Application Server
- Joss Enterprise Application Platform
- OW2 JOnAS
- IBM WebSphere Application Server
- IBM WebSphere Application Server Community Edition
- Apache Geronimo
- Fujitsu Interstage Application Server<sup>[16]</sup>
- TmaxSoft JEUS

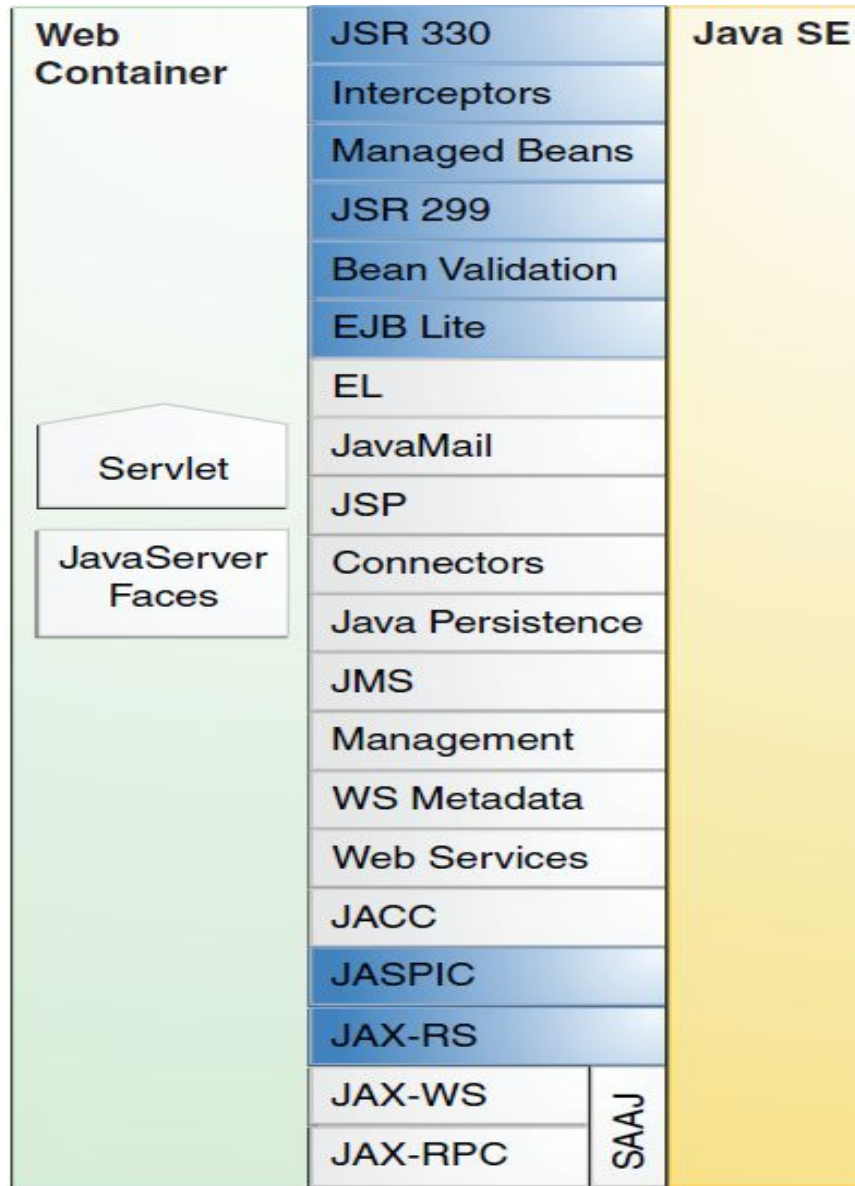
# Структура EAR файла



# Типы Java EE модулей

- EJB modules, which contain class files for enterprise beans and, optionally, an EJB deployment descriptor. EJB modules are packaged as JAR files with a .jar extension.
- Web modules, which contain servlet class files, web files, supporting class files, image and HTML files, and, optionally, a web application deployment descriptor. Web modules are packaged as JAR files with a .war (web archive) extension.
- Application client modules, which contain class files and, optionally, an application client deployment descriptor. Application client modules are packaged as JAR files with a .jar extension.
- Resource adapter modules, which contain all Java interfaces, classes, native libraries, and, optionally, a resource adapter deployment descriptor. Together, these implement the Connector architecture for a particular EIS. Resource adapter modules are packaged as JAR files with an .rar (resource adapter archive) extension.

# Java EE API для Web Container



# Java EE API для EJB Container

EJB Container	JSR 330	Java SE	
	Interceptors		
	Managed Beans		
	JSR 299		
	Bean Validation		
	JavaMail		
	Java Persistence		
	JTA		
	Connectors		
	EJB		JMS
	Management		
	WS Management		
	Web Services		
	JACC		
	JASPIC		
	JAXR		
	JAX-RS		
JAX-WS	SAAJ		
JAX-RPC			

# Java EE API для Application Client Container

Application Client Container	Java Persistence	Java SE	
	Management		
	WS Metadata		
	Web Services		
	JSR 299		
	JMS		
	JAXR		
	JAX-WS		SAAJ
	JAX-RPC		
	Application Client		

# Основные Java EE интерфейсы

JavaServer Faces

Servlet, включая JavaServer Pages  
contexts and Dependency Injection

Enterprise JavaBean

Java Persistence

Java Transaction (X/Open XA)

Java Message Service

Java Connector Architecture (лучше ESB)

# Web Tier

A *web application* is a dynamic extension of a web or application server. Web applications are of the following types:

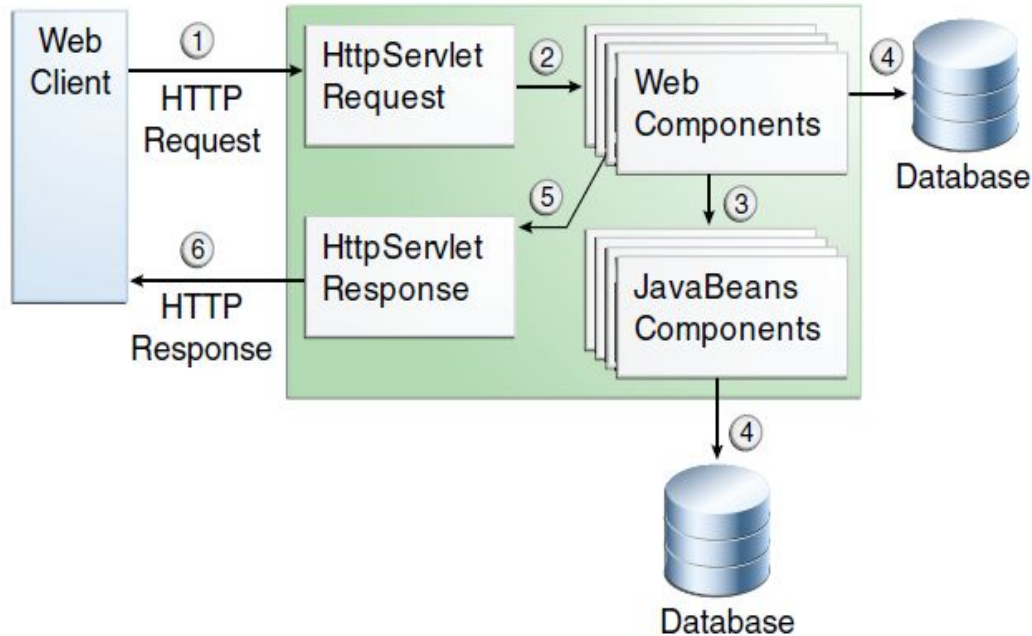
■ **Presentation-oriented:** A *presentation-oriented web application* generates interactive web pages containing various types of markup language (HTML, XHTML, XML, and so on) and dynamic content in response to requests.

■ **Service-oriented:** A *service-oriented web application* implements the endpoint of a web service.

Presentation-oriented applications are often clients of service-oriented web applications.



# Обработка HTTP запросов



**Web components:** Java servlets, web pages implemented with JavaServer Faces technology, JSP pages, web service (WS) endpoints.

**Web Services:** SOAP – based WS, RESTful WS

## Что даёт EJB:

1. Разработчик концентрируется на бизнес – логике.
2. Разработка клиентской части (Web) отделена от разработки бизнес – логики и работой с БД. Это особенно важно для малых устройств (смартфоны, планшеты).
3. Повторное использование кода.

## **Когда особенно следует использовать EJB:**

1. Масштабирование.
2. Транзакционность.
3. Разнообразие типов клиентов.

# Enterprise Bean

- **Session Bean: stateful, stateless, singleton (СИНХРОННЫЙ ВЫЗОВ).**
- **Message-driven Bean (АСИНХРОННЫЙ ВЫЗОВ)**
- **Entity Bean (Persistence КЛАССЫ)**

## **Statefull:**

- The bean's state represents the interaction between the bean and a specific client.
- The bean needs to hold information about the client across method invocations.
- The bean mediates between the client and the other components of the application, presenting a simplified view to the client.

## **Stateless:**

- The bean's state has no data for a specific client.
- In a single method invocation, the bean performs a generic task for all clients. For example, you might use a stateless session bean to send an email that confirms an online order.
- The bean implements a web service.

## **Singleton:**

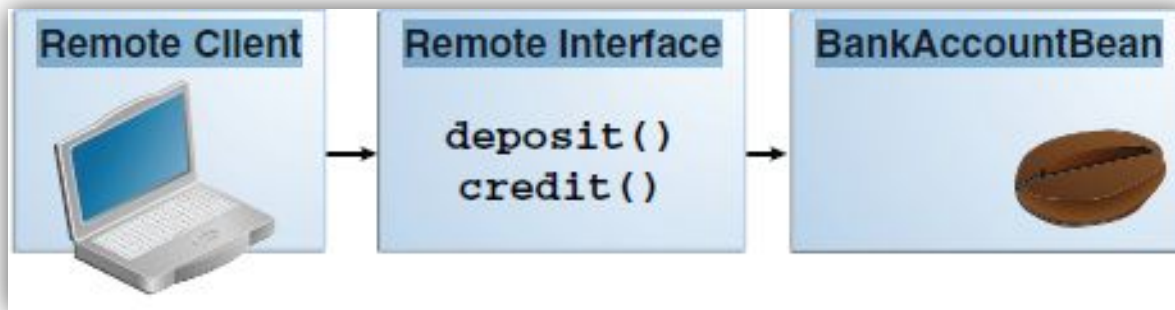
- State needs to be shared across the application.
- A single enterprise bean needs to be accessed by multiple threads concurrently.
- The application needs an enterprise bean to perform tasks upon application startup and shutdown.
- The bean implements a web service.

## **Message-driven (like stateless):**

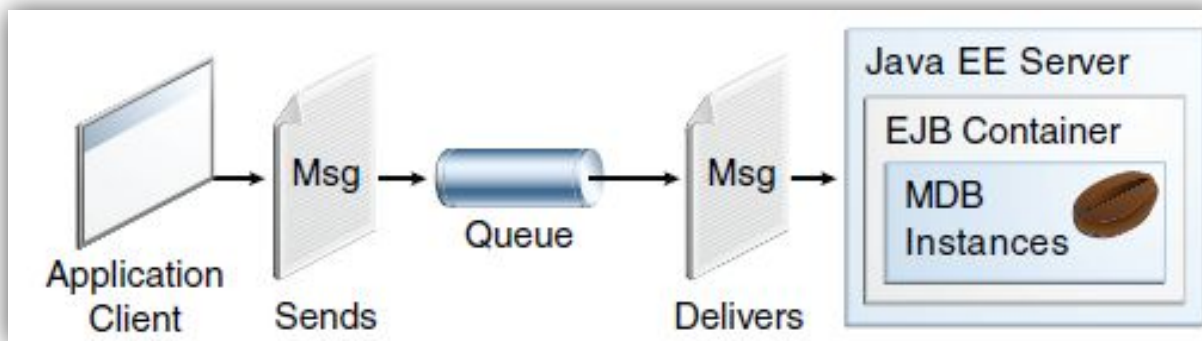
- To avoid tying up server resources, do not to use blocking synchronous; receives in a server-side component

# Enterprise Bean

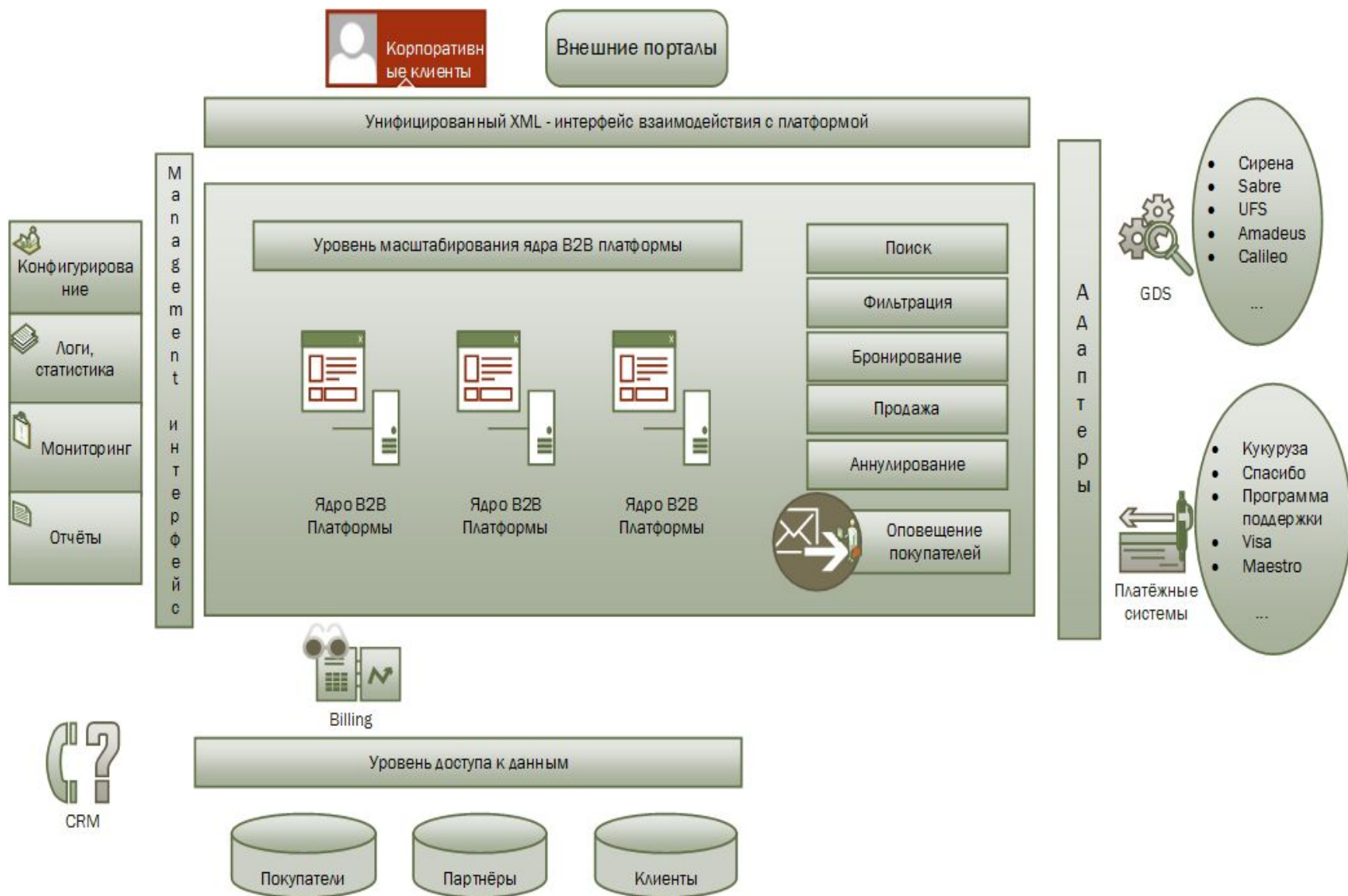
## Session Bean



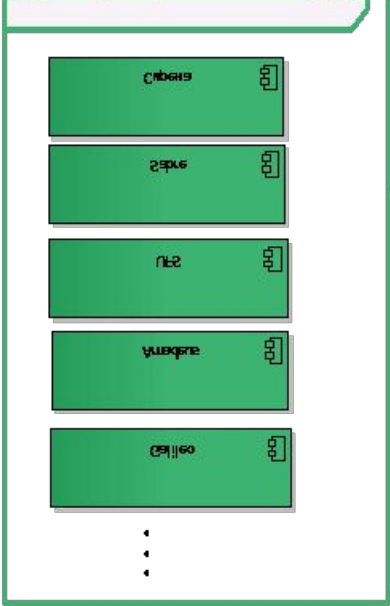
## Message-Driven Bean



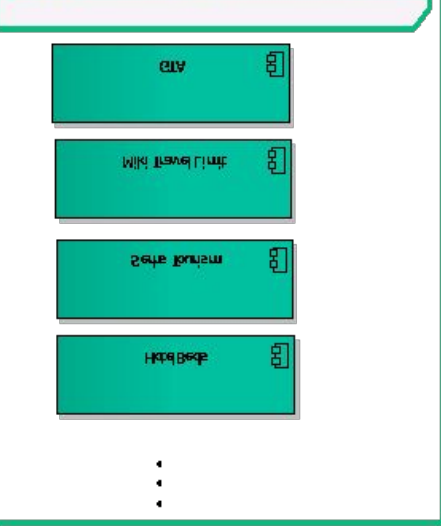
# Бизнес – архитектура одной системы online бронирования билетов



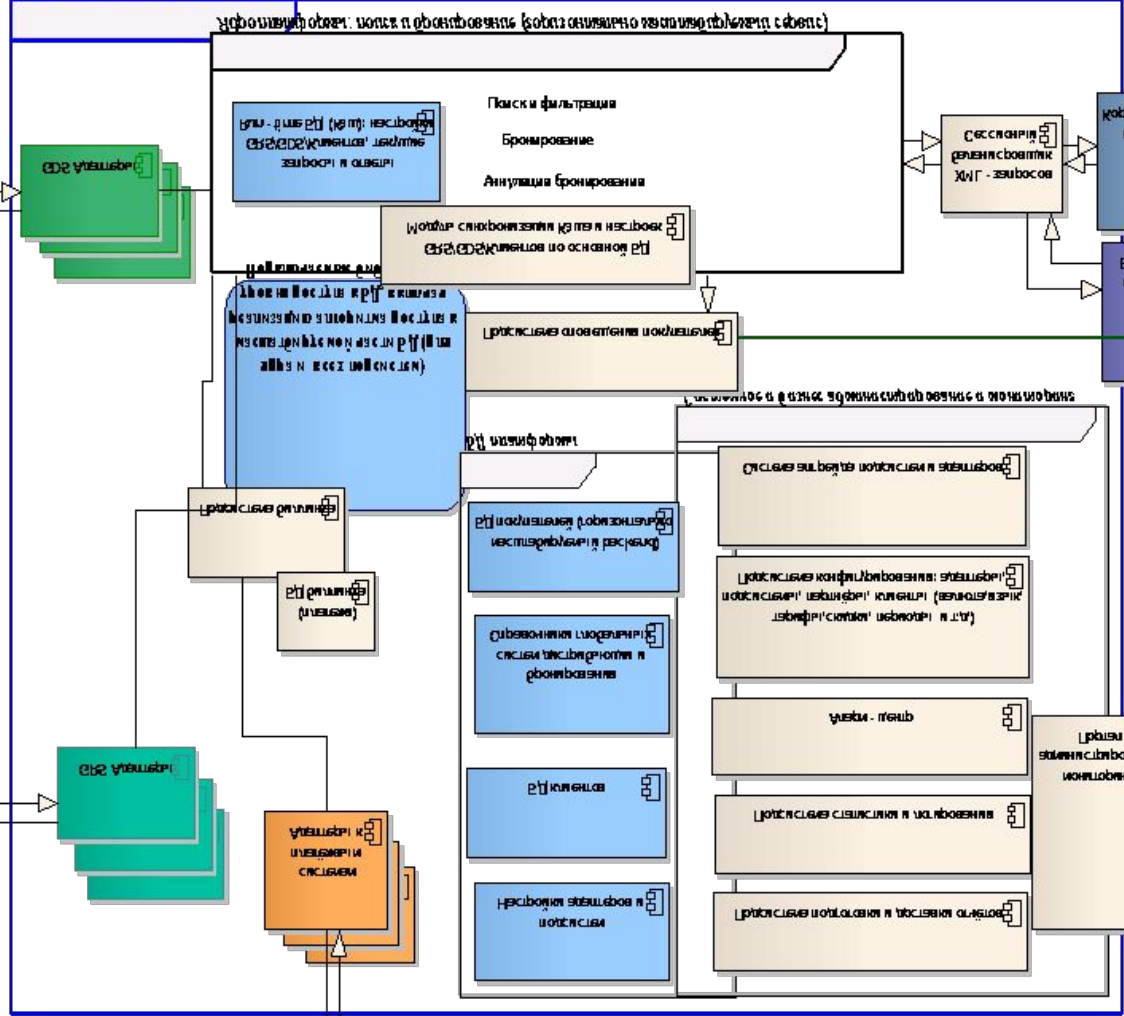
1. Информационная система (ИС)



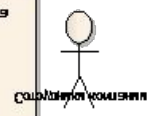
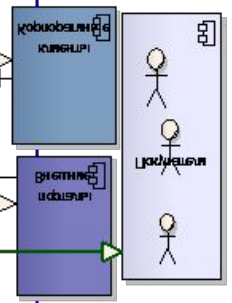
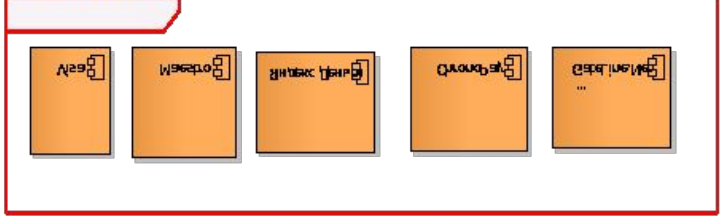
2. Информационная система (ИС) для бронирования и размещения услуг



БЭР микроформа бронирования



3. Информационная система





**Спасибо за  
внимание !**