



Платформа Java и IDE Netbeans 6 для разработки корпоративных приложений

Андрей Родионов
Sun Campus Ambassador
Sun Microsystems



Перечень тем

- Обзор Java платформы: Java SE, Java EE, Java ME
- IDE NetBeans как среда разработки разных типов приложений
- Двух- и трех– уровневая архитектура приложений
- Возможности Java EE для построения трех(много)-уровневых приложений
- Glassfish Application Server - обзор, демонстрация работы
- Технологии для разработки Web-уровня
- Технологии для разработки бизнес-уровня
- Современные подходы для разработки Web-приложений приложений

Java SE, Java EE, Java ME

J2SE

Desktop Solutions



- Standalone applications
- Distributed applications
- Applets

J2EE

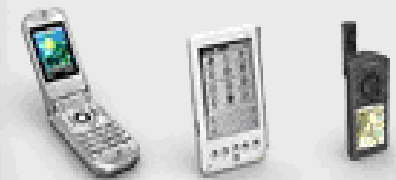
Enterprise Solutions



- Multi-tier Enterprise Apps
- eCommerce
- Web Services Support

J2ME

Consumer Solutions



- Cell phones
- PDAs
- TV set top boxes
- Car navigation Systems

Java Technology Product Groups

Java SE, Java EE, Java ME

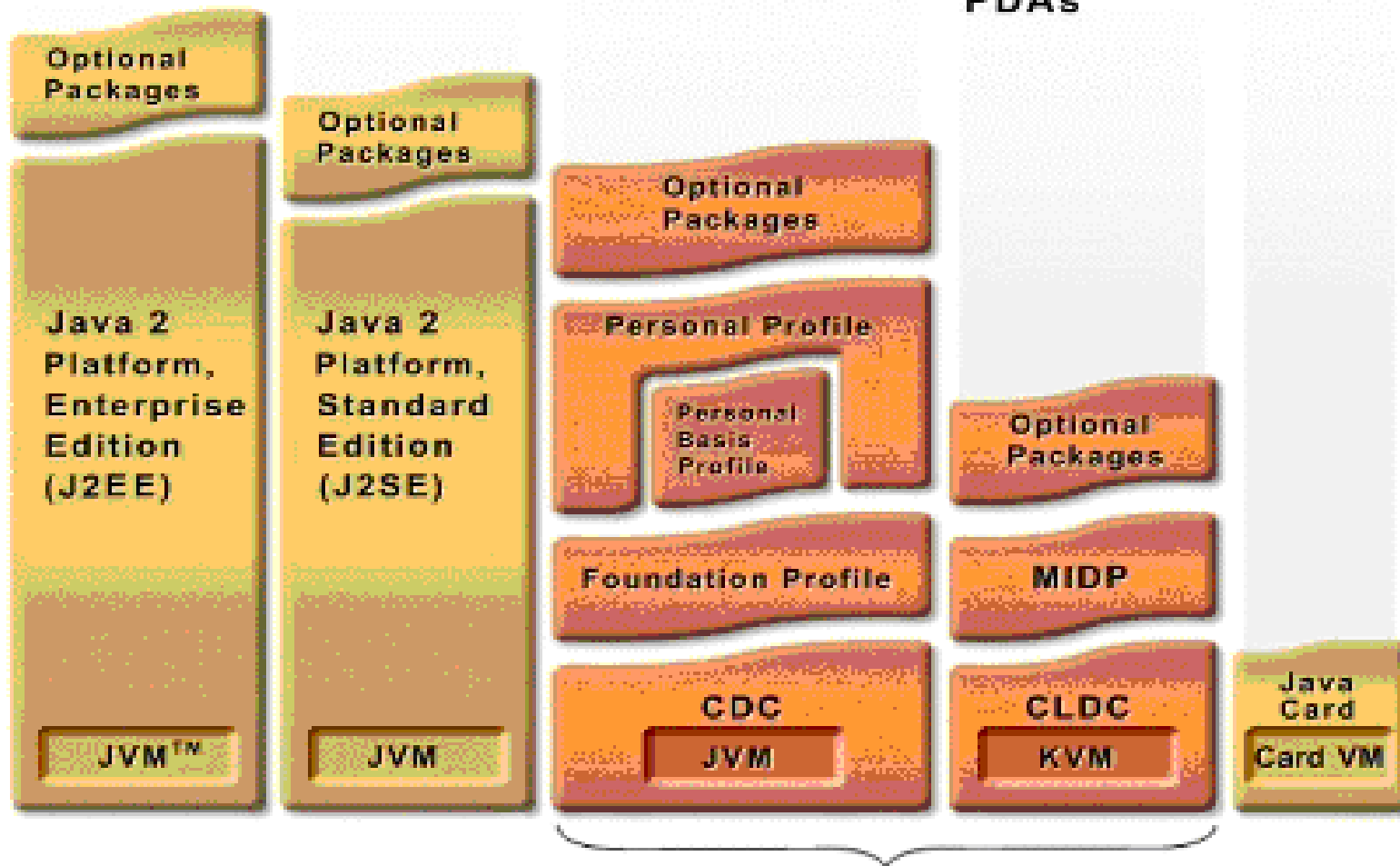
Servers & enterprise computers

Servers & personal computers

**High-end PDAs
TV set-top boxes
Embedded devices**

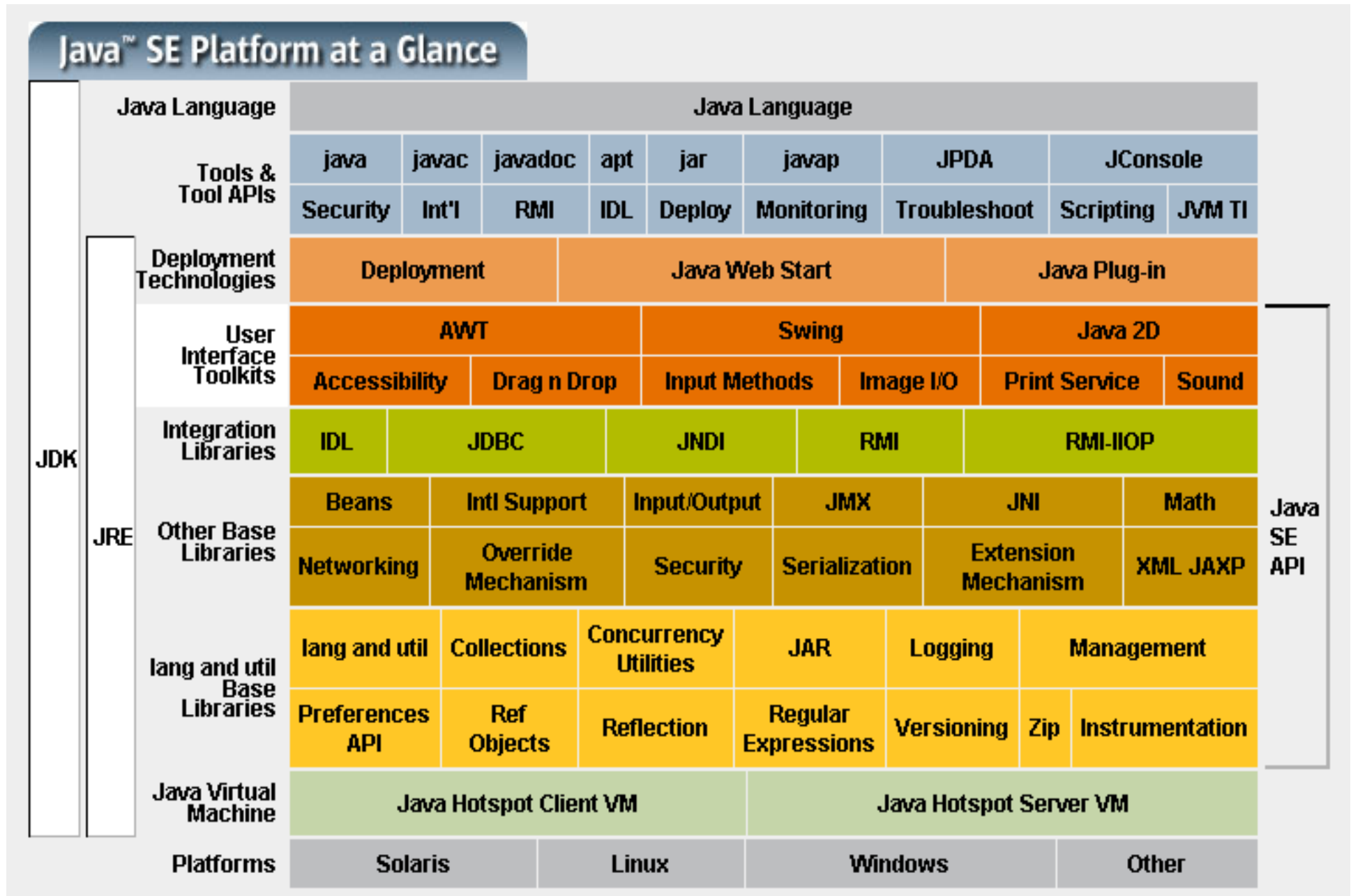
Mobile phones & entry-level PDAs

Smart cards



Java Platform, Micro Edition (Java ME)

Java Standard Edition



Что такое NetBeans IDE?

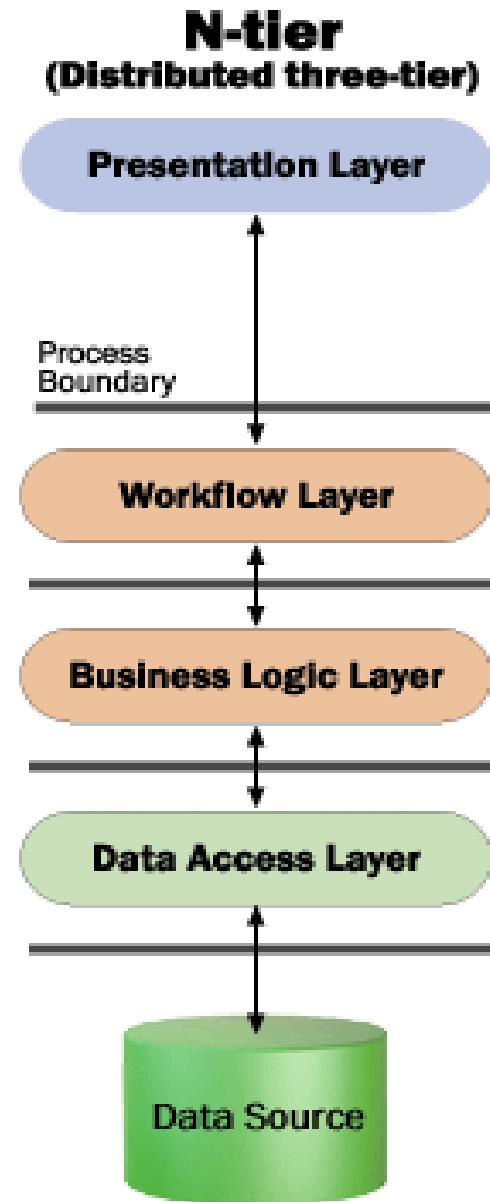
- Это бесплатная, модульная, с открытым исходным кодом интегрированная среда разработки, помогающая создавать, компилировать, развертывать (deployment) и отлаживать различные типы приложений
- Написана на языке программирования Java, однако может поддерживать разработку на других языках (C/C++, Ruby, PHP, ...)
- Работает под различными операционными системами (Windows, Linux, FreeBSD, Mac OS X и Solaris)
- Полная поддержка Java SE, Java EE, Java ME

NetBeans IDE 6.0 Features

- Java SE (Java Standard Edition)
- Web and Java EE (Java Enterprise Edition)
 - > Web Applications
 - > Java EE Applications
 - > Web Services
- Mobility (Java Micro Edition)
- Ruby, JRuby, Ruby on Rails
- C and C++
- UML
- SOA, XSLT Designer, WSDL and XML
- JavaFX

Двух- и трех– уровневая архитектура приложений

- Возможность использования различных языков программирования для различных компонентов
- Централизация компонентов
- Балансировка нагрузки
- Более эффективный доступ к данным
- Улучшенная защита
- Более простой доступ к внешним ресурсам

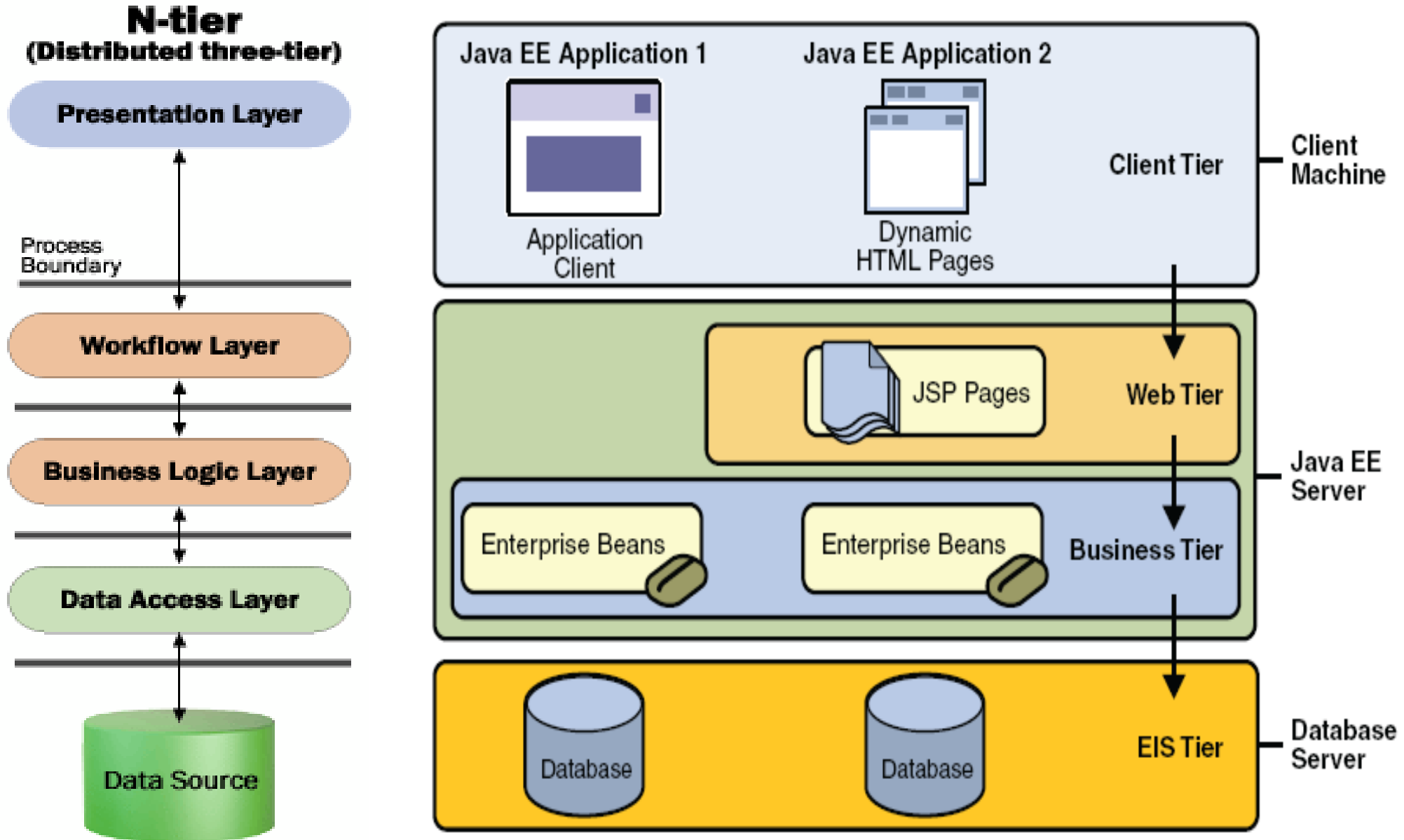


Java Enterprise Edition

- Построена на Java Standard Edition (Java SE)
- Web Application Technologies
- Web Services Technologies
- Enterprise Application Technologies
- Management and Security Technologies

- Sun Java System Application Server (Glassfish)

Java EE для построения трех(много)-уровневых приложений



Архитектура API Java EE

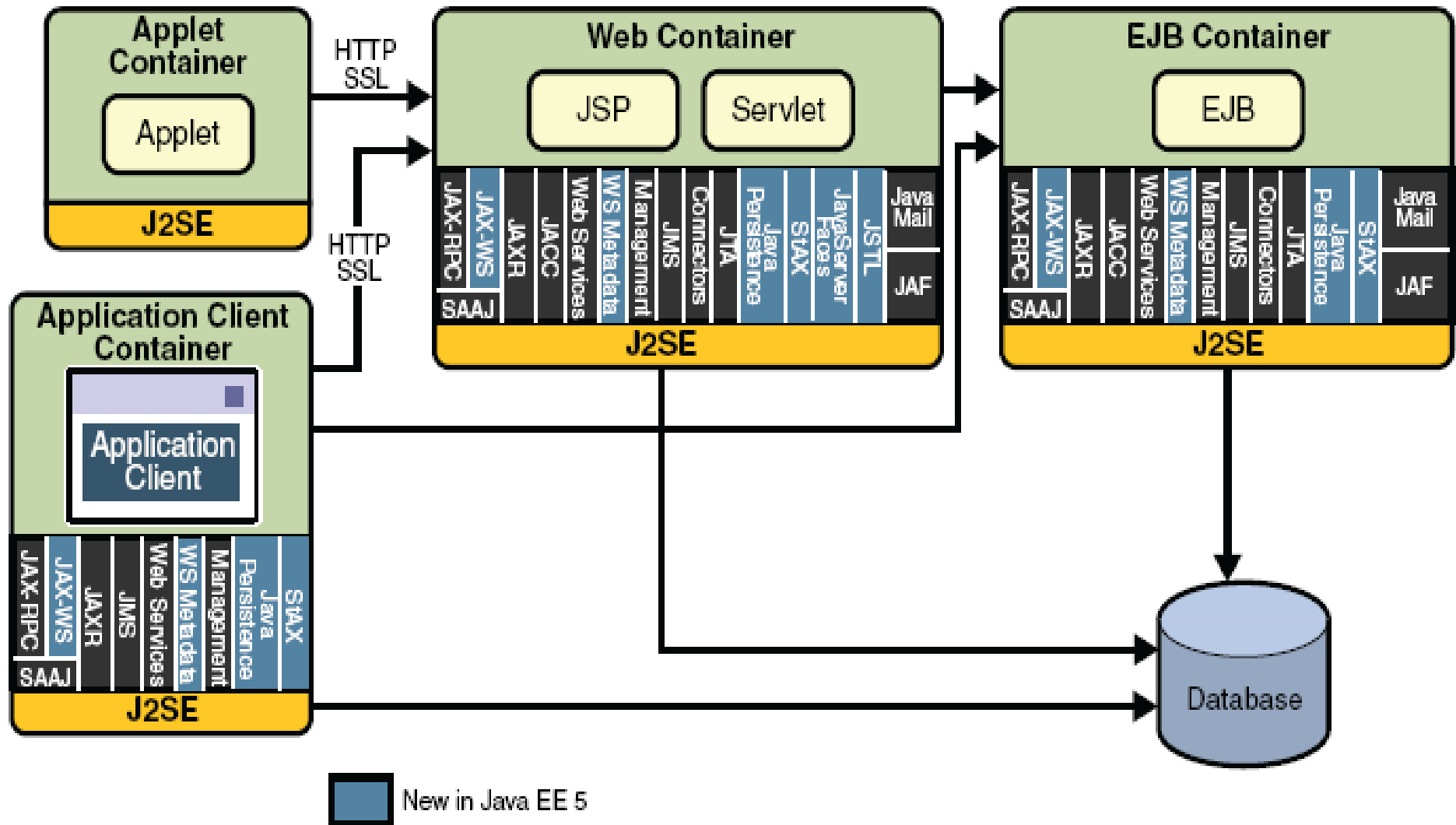


FIGURE 1-7 Java EE Platform APIs

Технологии для каждого из уровней

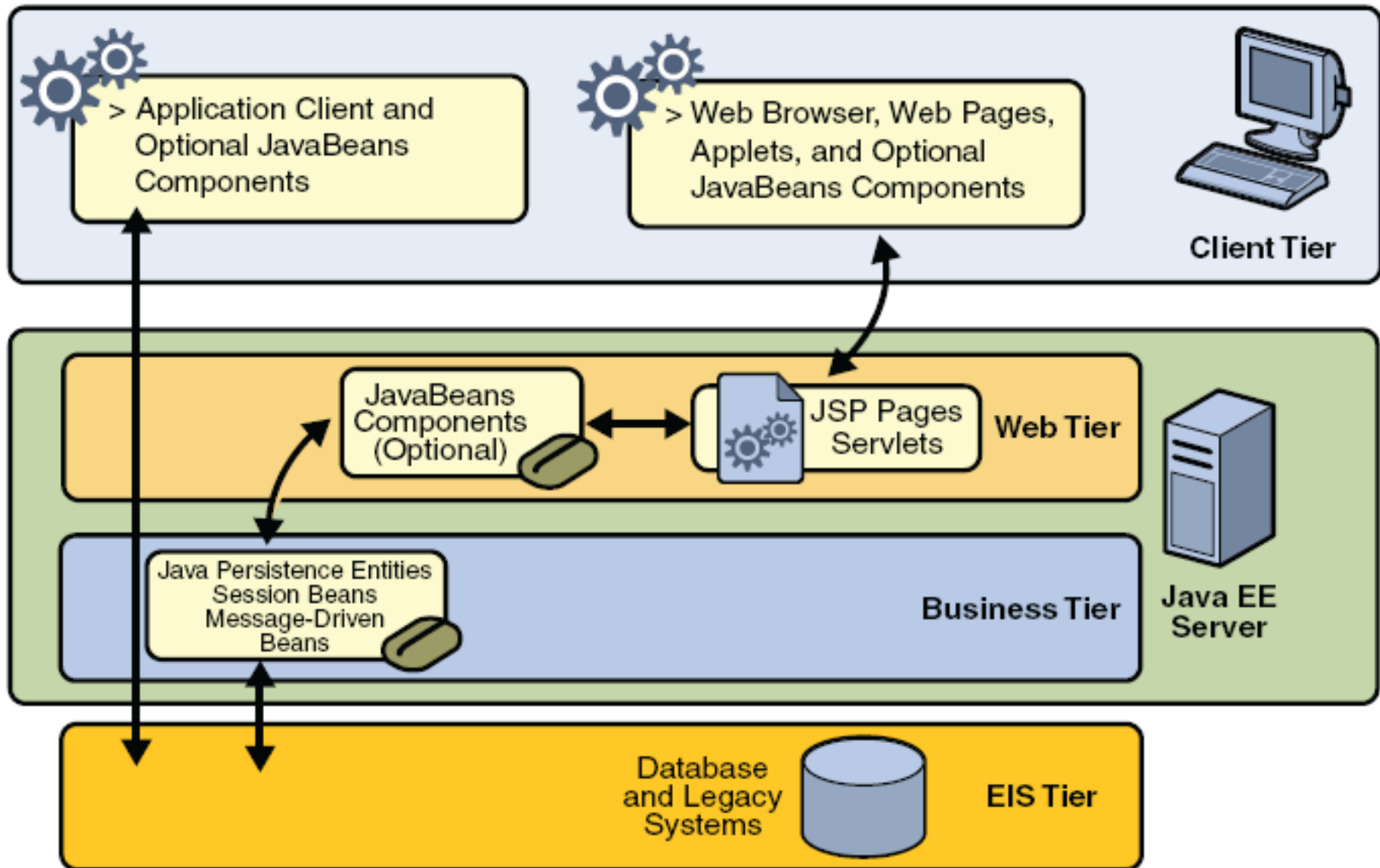


FIGURE 1-4 Business and EIS Tiers

Типы Web-приложений

- Presentation-oriented
 - > A presentation-oriented web application generates interactive web pages containing various types of markup language (HTML, 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

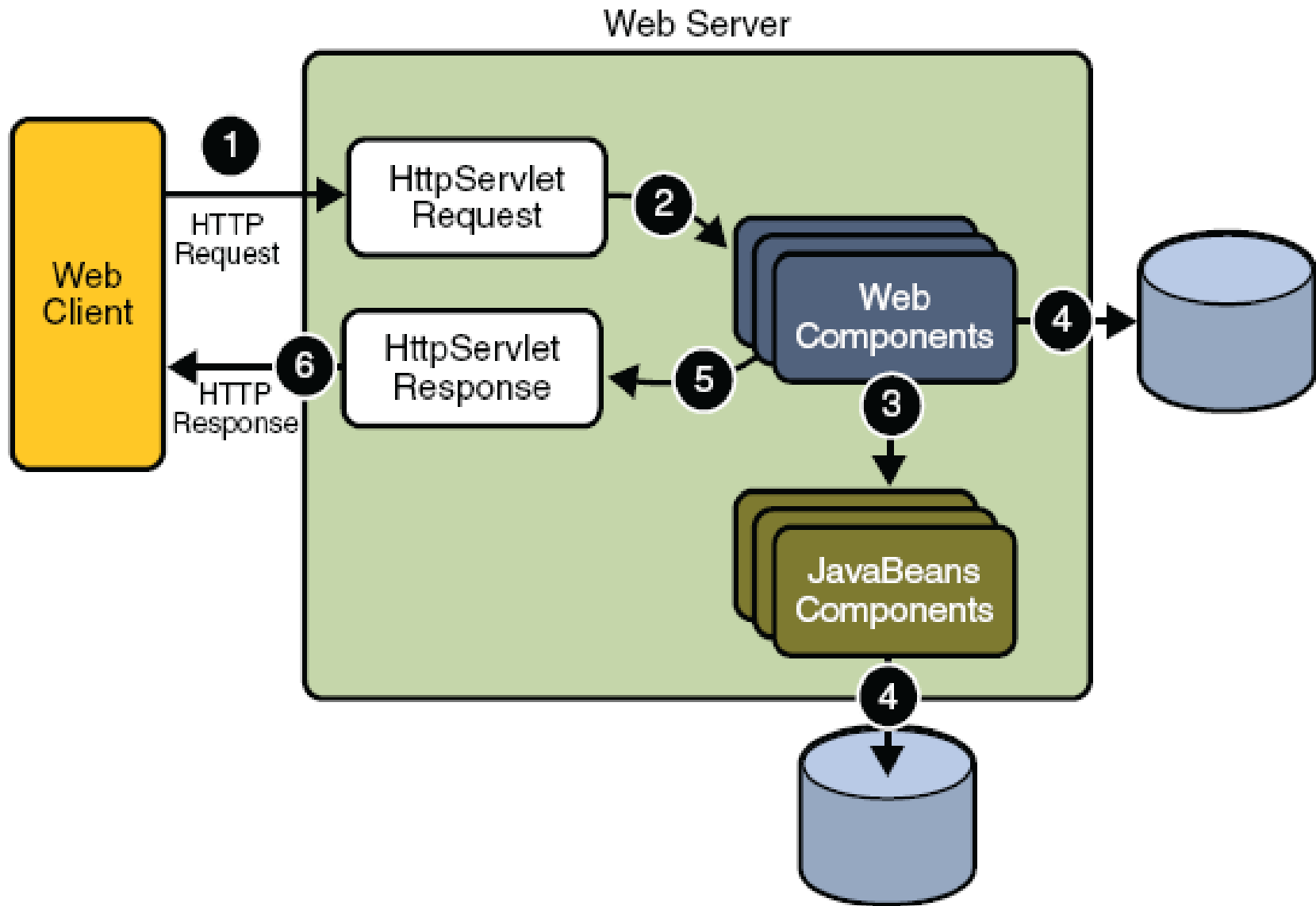


FIGURE 3-1 Java Web Application Request Handling

Технологии для разработки Web-уровня

- Java Servlet technology
- Java Server Pages (JSP)
- Java Server Faces

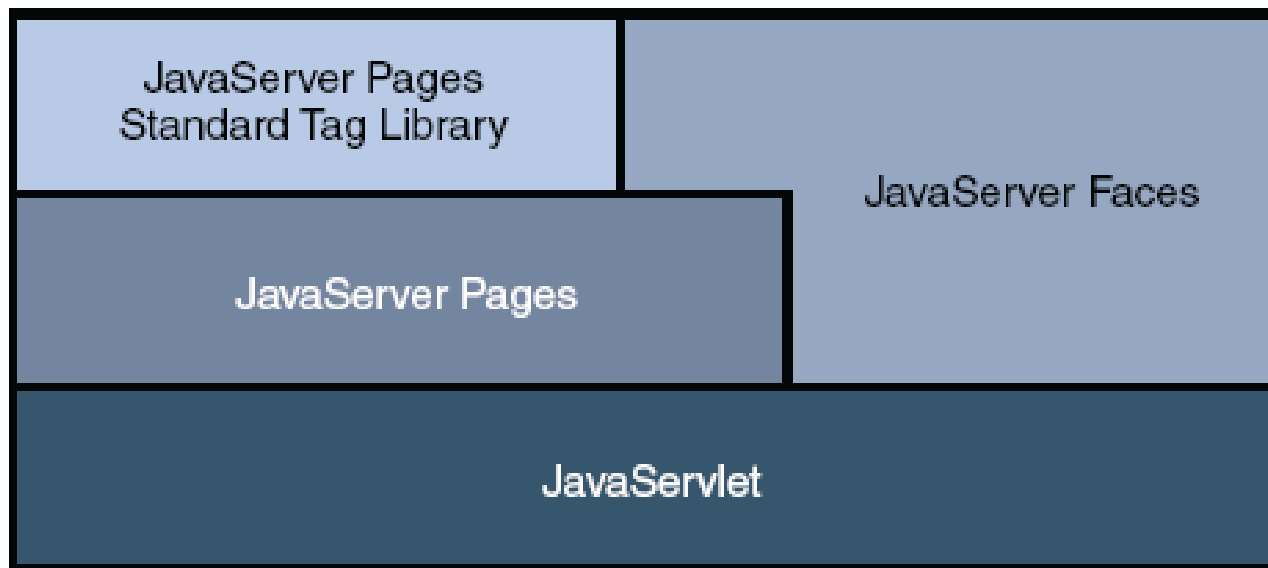


FIGURE 3-2 Java Web Application Technologies

Java Servlet technology

- A **servlet** is a Java programming language class

```
protected void processRequest (HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType ("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        /* TODO output your page here */
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet NewServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet NewServlet at " + request.getContextPath () + "</h1>");
        out.println("</body>");
        out.println("</html>");

    } finally {
        out.close();
    }
}
```

Java Server Pages (JSP)

- JavaServer Pages (JSP) technology allows you to easily create web content that has both static and dynamic components
- A JSP page is a text document that contains two types of text: static data, which can be expressed in any text-based format (such as HTML, SVG, WML, and XML), and JSP elements, which construct dynamic content

```

<body bgcolor="white">
<h1> Request Information </h1>
<font size="4">
JSP Request Method: <%= request.getMethod() %>
<br>
Request URI: <%= request.getRequestURI() %>
<br>
Request Protocol: <%= request.getProtocol() %>
<br>
Servlet path: <%= request.getServletPath() %>
<br>
Path info: <% out.print(util.HTMLFilter.filter(request.getPathInfo())); %>
<hr>
The browser you are using is <% out.print(util.HTMLFilter.filter(request.getHeader("User-Agent"))); %>
<hr>
</font>
</body>
</html>

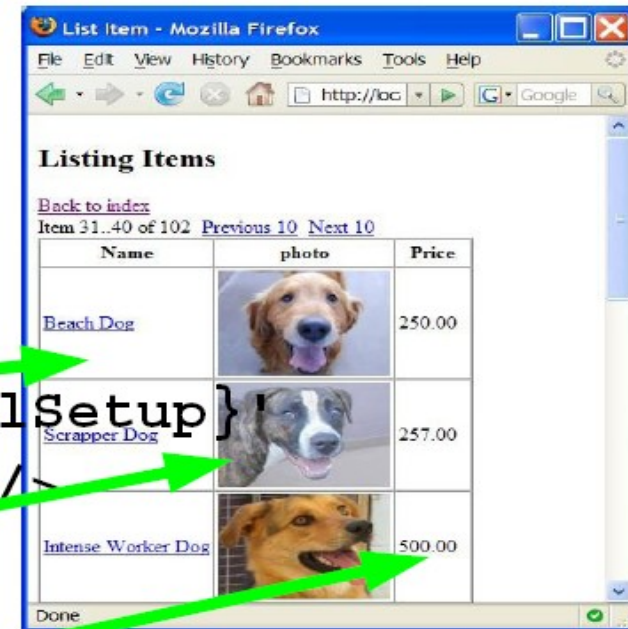
```




list.jsp Page

```

<h:dataTable value='#{itemController.items}'
  var='dataTableItem' >
  <h:column>
    <f:facet name='header'>
      <h:outputText value='Name' />
    </f:facet>
    <h:commandLink action='#{item.detailSetup}'
      value='#{dataTableItem.name}' />
  </h:column>
  ...
  <h:graphicImage url='#{dataTableItem.imageThumburl}' />
  ...
  <h:outputText value='#{dataTableItem.price}' />
  ...
</h:dataTable>

```



Name	photo	Price
Brach Dog		250.00
Scrapper Dog		257.00
Intense Worker Dog		500.00

Java Server Faces

- JavaServer Faces technology is a server-side user interface component framework for Java technology-based web applications

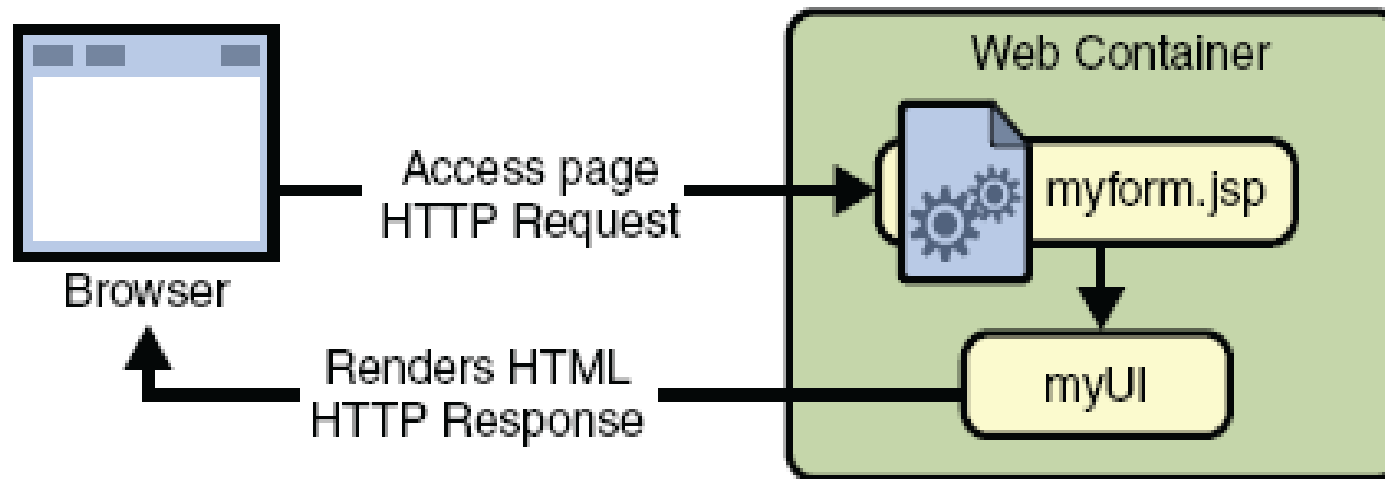


FIGURE 10-1 The UI Runs on the Server

Технологии для разработки бизнес-уровня

- Web Services
- Enterprise JavaBeans (EJB)
 - > Session Bean (Stateful, Stateless)
 - > Message-Driven Beans
- Java Persistence

Web Services

- Дает возможность удаленного вызова функций по протоколу HTTP
- Клиент и сервер могут базироваться на разных программных платформах
- Стандарт передачи данных SOAP (формат XML)

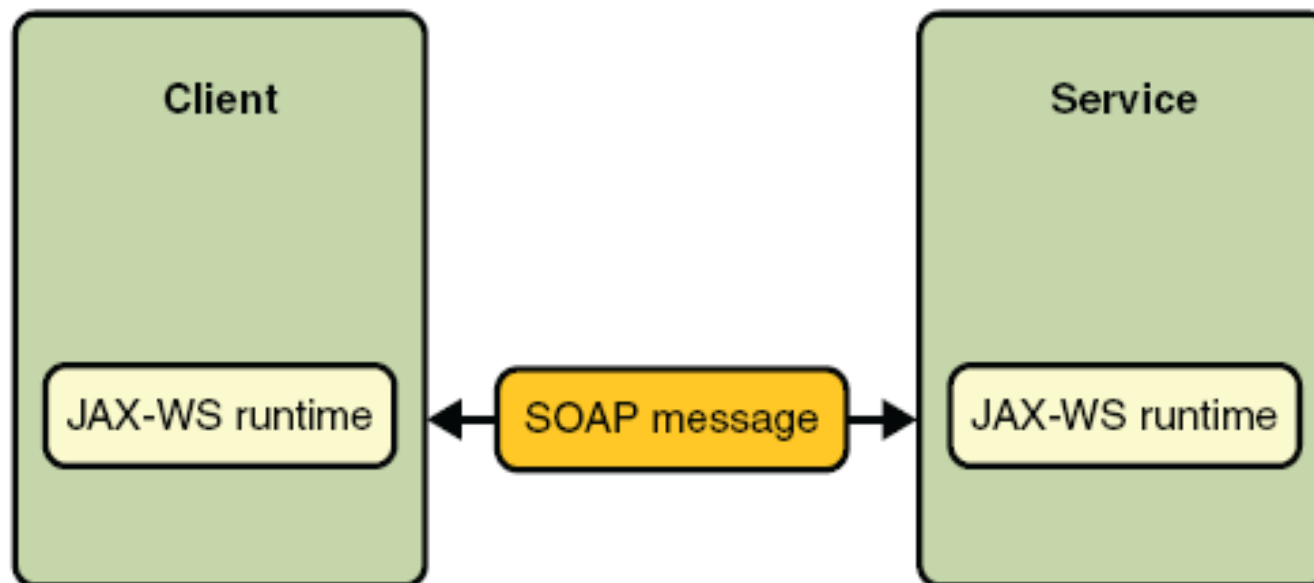


FIGURE 16-1 Communication between a JAX-WS Web Service and a Client

Web Service – Example

// Create a POJO

```
public class Calculator {  
    public int add(int a, int b) {  
        return a+b;  
    }  
}
```

Web Service – Example

```
// Create a service implementation
```

```
@WebService
```

```
public class Calculator {  
    public int add(int a, int b) {  
        return a+b;  
    }  
}
```

```
// Create and publish the endpoint
```

```
Calculator calculator = new Calculator();  
Endpoint endpoint = Endpoint.publish  
    ("http://localhost/calculator", calculator);
```

Web Service Client

```
// Create a Service object  
CalculatorService svc =  
    new CalculatorService();
```

```
// Create a proxy from the Service object  
Calculator proxy =  
    svc.getCalculatorPort();
```

```
// Invoke a Web service operation  
int answer = proxy.add(35, 7);
```

Демонстрация

- Создание веб-сервиса реализующего калькулятор
- Создание веб-клиента с помощью JSF
- Создание обычного клиента

When to Use Enterprise Beans?

- The application must be scalable. To accommodate a growing number of users, you may need to distribute an application's components across multiple machines.
- Transactions must ensure data integrity. Enterprise beans support transactions, the mechanisms that manage the concurrent access of shared objects.
- The application will have a variety of clients. With only a few lines of code, remote clients can easily locate enterprise beans. These clients can be thin, various, and numerous.

Types of Enterprise Beans

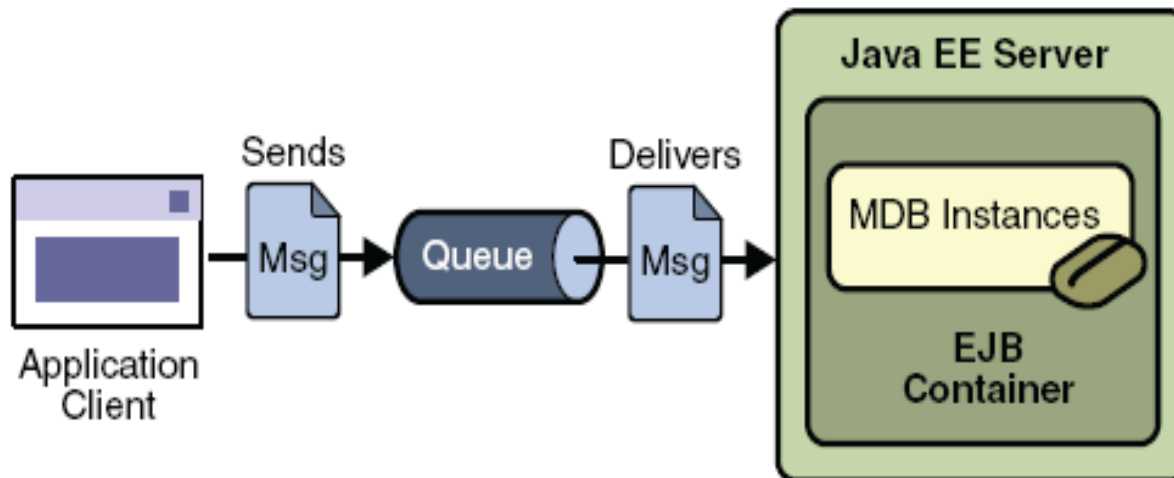
Session Bean - a session bean represents a single client inside the Application Server. To access an application that is deployed on the server, the client invokes the session bean's methods

- > Stateful
- > Stateless

- Message-Driven Beans - a message-driven bean is an enterprise bean that allows Java EE applications to process messages asynchronously. It normally acts as a JMS message listener, which is similar to an event listener except that it receives JMS messages instead of events.

Message-Driven Beans

Session beans allow you to send JMS messages and to receive them synchronously, but not asynchronously. To avoid tying up server resources, do not to use blocking synchronous receives in a server-side component, and in general JMS messages should not be sent or received synchronously. To receive messages asynchronously, use a message-driven bean.



Java Persistence

The Java Persistence API provides an object/relational mapping facility to Java developers for managing relational data in Java applications.

Java Persistence consists of three areas:

- The Java Persistence API
- The query language
- Object/relational mapping metadata

Entity

- An entity is a lightweight persistence domain object. Typically an entity represents a table in a relational database, and each entity instance corresponds to a row in that table. The primary programming artifact of an entity is the entity class

Simple Mappings

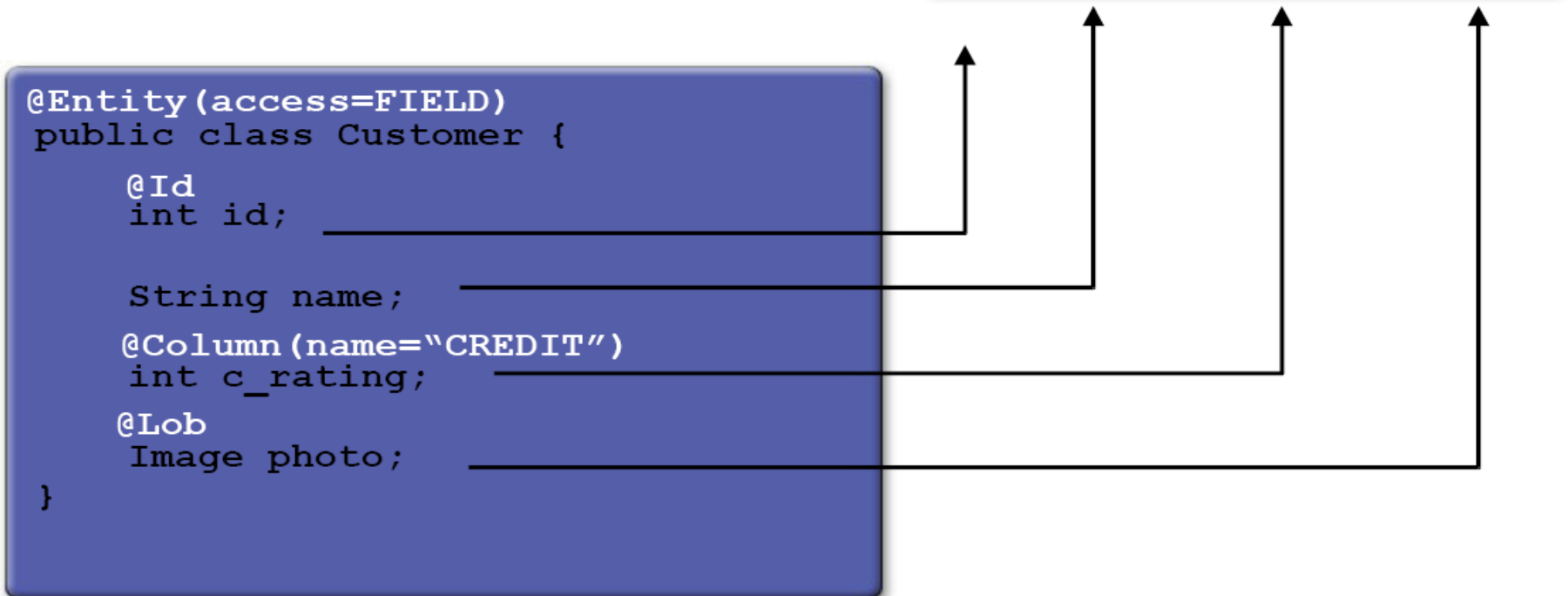
```
@Entity(access=FIELD)
public class Customer {
    @Id
    int id;

    String name;

    @Column(name="CREDIT")
    int c_rating;

    @Lob
    Image photo;
}
```

CUSTOMER			
ID	NAME	CREDIT	PHOTO



Entity

Annotated as "Entity"

@Id denotes primary key

@Entity

```
public class Customer implements Serializable {
    @Id protected Long id;
    protected String name;

    public Customer() {}

    public Long getId() {return id;}

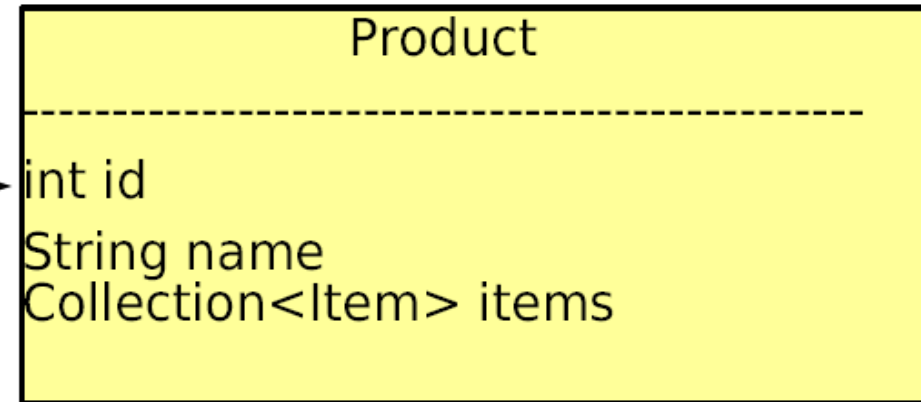
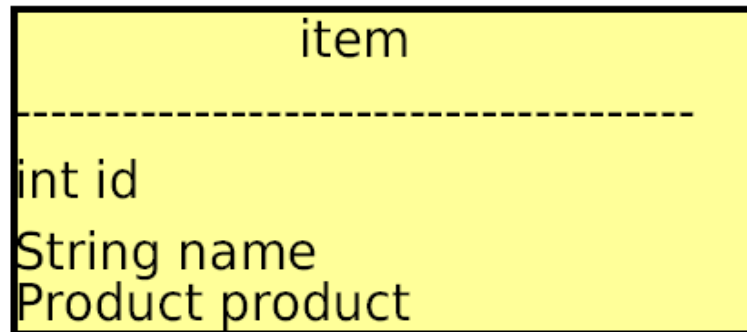
    public String getName() {return name;}
    public void setName(String name) {this.name = name;}

    ...
}
```

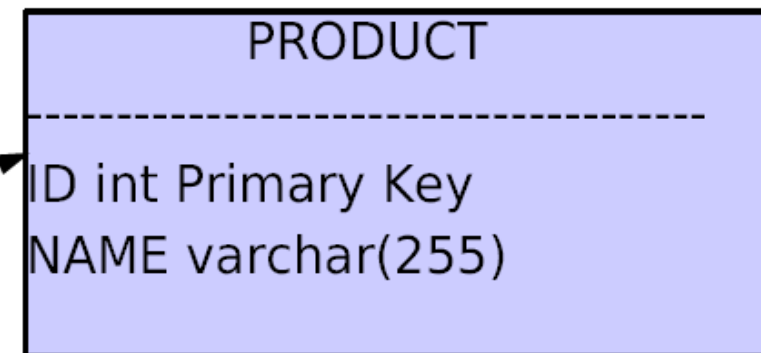
O-R mapping

Example Data Model

CLASSES



TABLES



Relationship Mappings – ManyToMany

owner
of Relationship

```
@Entity(access=FIELD)
public class Customer {
    @Id
    int id;
    ...
    @ManyToMany
    Collection<Phone> phones;
}
```

Inverse
of Relationship

```
@Entity(access=FIELD)
public class Phone {
    @Id
    int id;
    ...
    @ManyToMany(mappedBy="phones")
    Collection<Customer> custs;
}
```



Relationship Mappings – OneToMany

Bidirectional

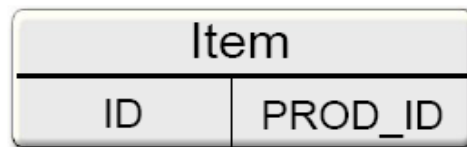
```
@Entity
public class Item {
    @Id
    int id;
    ...

    @ManyToOne
    Product product;
}
```

```
@Entity
public class Product {
    @Id
    int id;
    ...

    @OneToMany(mappedBy="product"
    Collection<Item> items;
}
```

Relationship
Owner



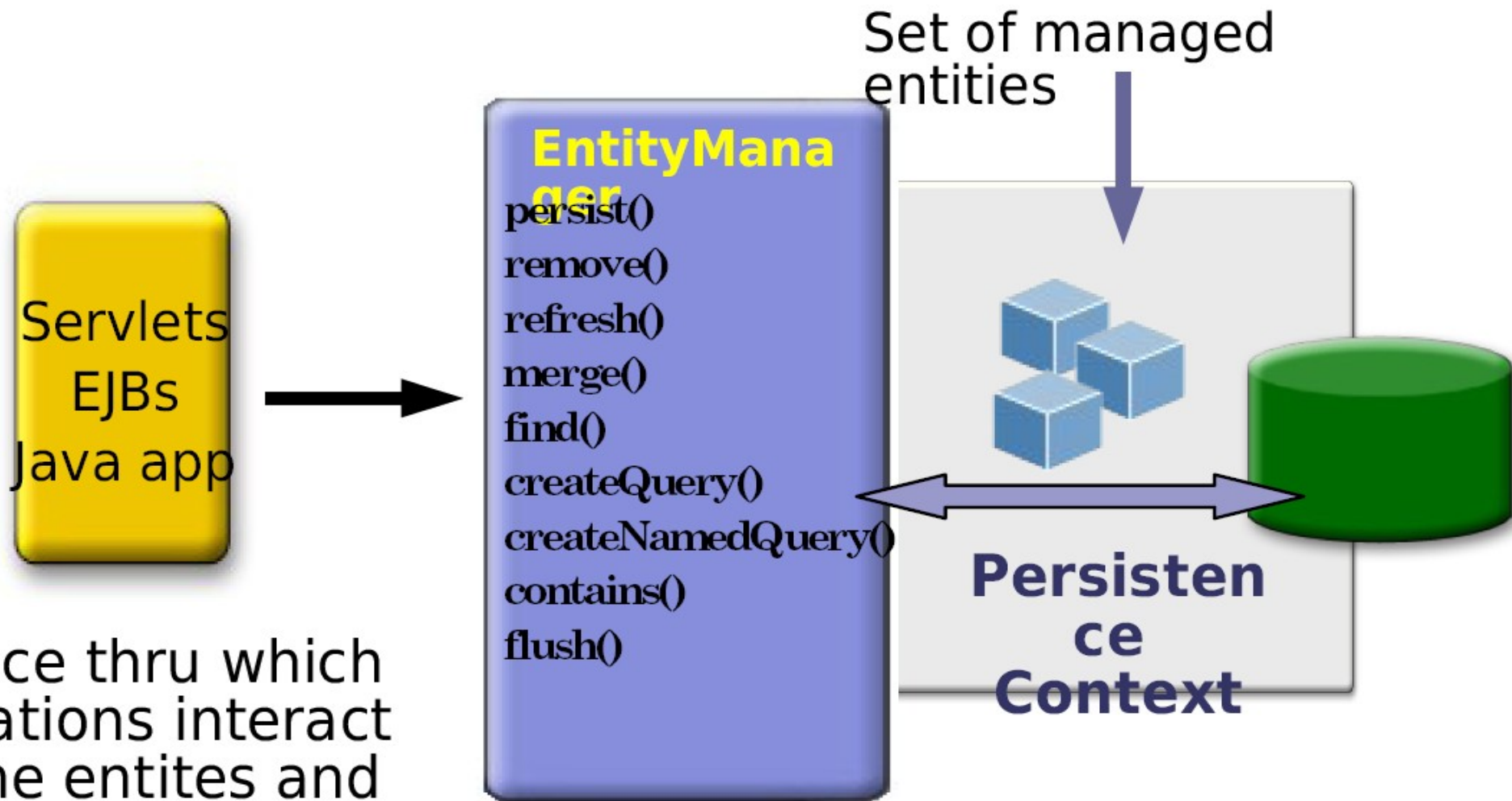
has to say **where the foreign key is using mappedBy**



1

Inverse
of Relationship

What is the EntityManager?

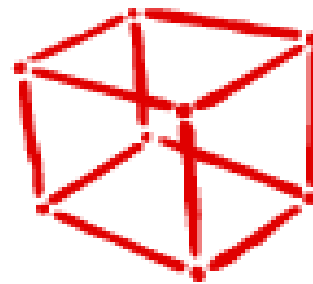


Interface thru which applications interact with the entites and the persistence engine

Поддерживаемые в NetBeans IDE 6.0 технологии для веб-разработки

- JavaServer Faces
- Ruby and Rails
- Google Web Toolkit Framework
- Struts Web Framework
- Spring Web Framework
- jMaki Framework

- JavaScript Editor
- CSS Editor



built on
NetBeans



СПАСИБО!

– Andrii.Rodionov@sun.com

–

> <http://blogs.sun.com/ntuu>

> <http://osug.org.ua/>

>

>

>

>

>

>

>

>

>

>

>

