

RESTful Web сервисы и Java



JAX-RS, JSP+RESTful

Детали:

https://ru.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition

Jakarta EE

ЧТО ВНУТРИ?

EJB (Enterprise JavaBeans) спецификация технологии серверных компонентов, содержащих бизнес-логику

JPA (Java Persistence API) управление постоянством и объектно-реляционное отображение

Servlet Обслуживание запросов веб-клиентов

JSP (JavaServer Pages) динамическая генерация веб-страниц на стороне сервера

JAX-WS Java API for XML Web Services — создание веб-сервисов

JAX-RS Java API for RESTful Web Services — создание RESTful веб-сервисов

JSON-P Java API for JSON Processing — разбор и генерация JSON

JSON-B Java API for JSON Binding — преобразование Java объектов в/из JSON

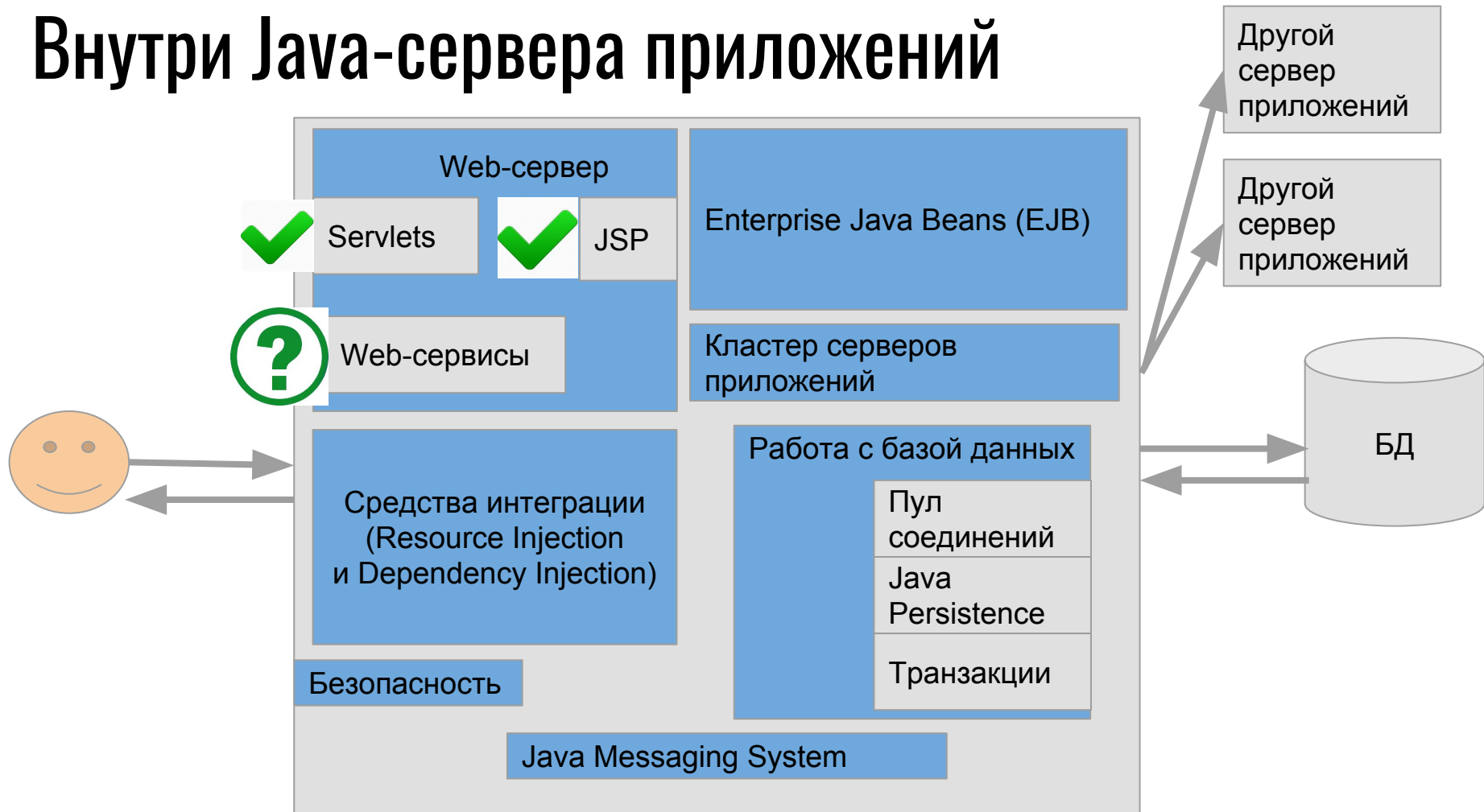
JNDI Java Naming and Directory Interface — служба каталогов

JavaMail Получение и отправка электронной почты

JACC Java Authorization Contract for Containers

.....

Внутри Java-сервера приложений



Технология Web сервисов

Web-сервис

- программная система со стандартизированными интерфейсами, доступная по Web-адресу (URL)

Технология Web-сервисов

технология создания распределенных систем, составленных из взаимодействующих между собой программных продуктов, созданных и работающих на основе различных платформ

Web-сервисы согласовывают работу больших, состоящих из множества частей приложений, предоставляя для приложений бизнес-функции обмена данными

Взаимодействие Web-сервисов

Веб-сервисы могут взаимодействовать друг с другом и со сторонними приложениями посредством сообщений, основанных на определённых протоколах:

SOAP, XML-RPC и т. д.

и соглашениях:

REST

Группы Web-сервисов

- **SOAP** Web-сервисы, ориентированные на модель RPC — вызов удаленных процедур (Remote Procedure Call)

протокол - **SOAP**, интерфейсы описаны на **WSDL**, сообщения - **в XML**

- **XML** Web-сервисы, ориентированные на сообщения

протокол - **SOAP**, сообщения - **в XML**

- **RESTful** Web-сервисы

представляют удаленные ресурсы, доступные с помощью HTTP-запросов

протокол - HTTP: GET, POST, PUT, DELETE, PATCH

REST - Representational State Transfer

Впервые описан
Roy Fielding в
2000 г.

REST - архитектурный стиль, основанный на Web-стандартах и протоколе HTTP.

В архитектуре стиля REST “всё - ресурс”.

Ресурс доступен другим через стандартный интерфейс, основанный на стандартных методах HTTP.

В архитектуре стиля REST есть REST сервер, предоставляющий доступ к ресурсам.

REST клиент может обращаться и изменять REST ресурсы.

REST - Representational State Transfer(2)

Каждый ресурс должен поддерживать стандартные операции протокола HTTP (GET, POST, PUT, DELETE, PATCH).

Ресурсы идентифицируются глобальными идентификаторами, типично это - их URI.

REST допускает, что ресурсы имеют разное представление: текст, XML, JSON. REST клиент может запросить конкретное представление ресурса, используя возможности протокола

HTTP

```
GET /contacts HTTP/1.1 Accept: application/json HTTP/1.1
200 OK Content-Type: application/json [ { "id": 1,
"lastName": "Fielding", "firstName": "Roy" } ]
```

<http://restlet.com/company/blog/2015/12/10/understanding-http-content-negotiation/>
<https://www.w3.org/Protocols/rfc2616/rfc2616-sec12.html> (HTTP content negotiation)

RESTful Web-сервис - аннотации

<https://docs.oracle.com/cd/E19776-01/820-4867/ggnxo/index.html>

Работы по созданию RESTful Web-сервиса

Работы по созданию RESTful Web-сервиса

1. Выбрать фреймворк, реализующий JAX-RS API
2. Выбрать Web-сервер или контейнер сервлетов
3. Создать Java класс, реализующий Web-сервис: в нем определить методы реагирования на конкретные виды HTTP запросов (GET, POST,...) и возвращения содержимого разных типов (XML, TEXT, JSON,...)

JERSEY = свободный JAX-RS API



RESTful Web Services in Java.

Реализует интерфейсы JAX-RS
и предоставляет
дополнительные возможности

Реализация Web-сервисов

ПАКЕТЫ

javax.ws.rs.DELETE

javax.ws.rs.GET

javax.ws.rs.POST

javax.ws.rs.PUT

javax.ws.rs.Path

javax.ws.rs.PathParam

javax.ws.rs.Consumes

javax.ws.rs.Produces

javax.ws.rs.core.MediaType

javax.xml.bind.JAXBElement

...

Пример Web- сервиса

```
package 4325webservice;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
@Path("/hello") //Установить путь к базовому URL + /hello
public class Hello {
// Этот метод вызывается, когда пришёл запрос на TEXT_PLAIN
@GET
    @Produces(MediaType.TEXT_PLAIN)
    public String sayPlainTextHello() { return "Hello Jersey"; }
// Этот метод вызывается, когда пришёл запрос на HTML
@GET
    @Produces(MediaType.TEXT_HTML)
    public String sayHtmlHello() { return "<html> " + "<title>" + "Hello
Jersey" + "</title>" + "<body><h1>" + "Hello Jersey" + "</body></h1>"
+ "</html> "; } }
```


web.xml для Web- сервиса

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID"
version="3.0">
  <display-name>4325</display-name>
  <servlet>
    <servlet-name>Jersey REST Service</servlet-name>
    <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
    <init-param>
      <param-name>jersey.config.server.provider.packages</param-name>
      <param-value>4325webservice</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Jersey REST Service</servlet-name>
    <url-pattern>/rest/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Вызовы Web- сервиса

<http://localhost:8080/4325webservice/rest/hello>

Полезные программы для тестирования HTTP запросов и форматы разметки



Postman

<https://www.getpostman.com/>



<https://www.telerik.com/fiddler>

YAML

более дружелюбный, чем JSON и XML,
язык разметки