

Bởi: Võ Văn Hải
Email: vovanhai@lth.edu.vn

Jakarta EE / Java EE

1
14/08/2023

1

mục tiêu

Tổng quan về kiến trúc JakartaEE

Tổng quan về container JakartaEE

Tổng quan về API JakartaEE

Dịch vụ Jakarta

- Vòng đời của Servlet
- Chia sẻ thông tin
- Xử lý không đồng bộ
- Bối cảnh và tiêm phụ thuộc (CDI)
- API REST
- JSP
- Ở các web

Trang máy chủ Java

14/08/2023

2

Lịch sử

Platform version	Released Date	Specification	Java SE support
Jakarta EE 10	13 September 2022	10	Java SE 17 / 11
Jakarta EE 9.1	25 May 2021	9.1	Java SE 11 / 8
Jakarta EE 9	08 December 2020	9	Java SE 8
Jakarta EE 8	10 September 2019	8	Java SE 8
Java EE 8	31 August 2017	JSR 366	Java SE 8
Java EE 7	28 May 2013	JSR 342	Java SE 7
Java EE 6	10 December 2009	JSR 316	Java SE 6
Java EE 5	11 May 2006	JSR 244	Java SE 5
J2EE 1.4	11 November 2003	JSR 151	J2SE 1.4
J2EE 1.3	24 September 2001	JSR 58	J2SE 1.3
J2EE 1.2	17 December 1997	1.2	J2SE 1.2

14/08/2023

3

Kiến trúc JavaEE

Ứng dụng đa tầng phân tán

Các phần ứng dụng Jakarta EE:

- Các thành phần cấp client chạy trên máy client.
- Các thành phần cấp web chạy trên máy chủ Jakarta EE.
- Các thành phần cấp doanh nghiệp chạy trên máy chủ Jakarta EE.
- Hệ thống thông tin doanh nghiệp (EIS)
 - tầng phần mềm chạy trên máy chủ EIS.

Mặc dù một ứng dụng Jakarta EE có thể bao gồm tất cả các tầng nhưng các ứng dụng nhiều tầng là Jakarta EE thường được coi là ứng dụng ba tầng vì chúng được phân phối trên ba địa điểm: máy khách, máy chủ Jakarta EE và cơ sở dữ liệu hoặc các máy kết thừa ở phía sau.

Hình 1-1 Ứng dụng nhiều tầng

14/08/2023

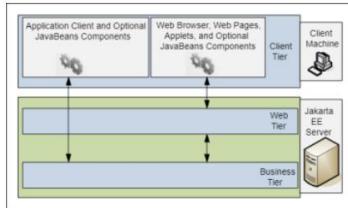
4

Kiến trúc JavaEE

Truyền thông máy chủ Jakarta EE

Các yếu tố khác nhau có thể tạo nên cấp độ khách hàng. Máy khách

liên lạc trực tiếp với tầng doanh nghiệp chạy trên máy chủ Jakarta EE hoặc, như trong trường hợp máy khách chạy trên trình duyệt, bằng cách đi qua các trang web hoặc máy chủ chạy trong tầng web.



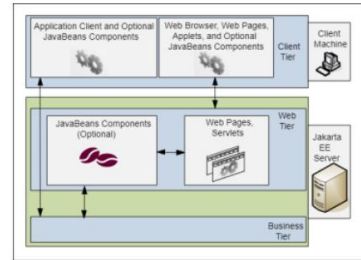
Hình 1-2 Giao tiếp máy chủ

14/08/2023

5

Kiến trúc JavaEE

Thành phần web



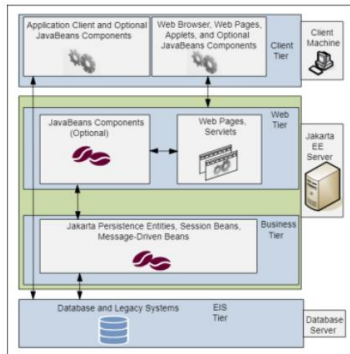
Hình 1-3 Tầng Web và các ứng dụng Jakarta EE

14/08/2023

6

Kiến trúc JavaEE

Thành phần kinh doanh



Hình 1-4 Cấp độ doanh nghiệp và EIS

14/08/2023

7

Container EE Jakarta

Dịch vụ container

Container là giao diện giữa một thành phần và chức năng cụ thể của nền tảng cấp thấp hỗ trợ thành phần đó. Trước khi có thể thực thi, một thành phần web, Enterprise Bean hoặc ứng dụng khách phải được lắp ráp thành mô-đun Jakarta EE và được triển khai vào vùng chứa của nó.

Quy trình lắp ráp bao gồm việc chỉ định cài đặt vùng chứa cho từng thành phần trong ứng dụng Jakarta EE và cho chính ứng dụng Jakarta EE. Cài đặt vùng chứa tùy chỉnh hỗ trợ cơ bản do máy chủ Jakarta EE cung cấp, bao gồm các dịch vụ như bảo mật, quản lý giao dịch, tra cứu API Giao diện thống nhất và đặt tên Java (JNDI) cũ ng như kết nối từ xa. Đây là một số điểm đáng chú ý.

- Mô hình bảo mật Jakarta EE cho phép bạn định cấu hình thành phần web hoặc doanh nghiệp để tài nguyên hệ thống chỉ được truy cập bởi người dùng được ủy quyền.
- Mô hình giao dịch Jakarta EE cho phép bạn chỉ định mối quan hệ giữa các phương thức tạo nên một giao dịch duy nhất để tất cả các phương thức trong một giao dịch được coi là một đơn vị duy nhất.
- Dịch vụ tra cứu JNDI cung cấp giao diện thống nhất cho nhiều dịch vụ đặt tên và thư mục trong doanh nghiệp để các thành phần ứng dụng có thể truy cập các dịch vụ này.
- Mô hình kết nối từ xa Jakarta EE quản lý thông tin liên lạc cấp thấp giữa khách hàng và đầu doanh nghiệp. Sau khi một bean doanh nghiệp được tạo, một máy khách sẽ gọi các phương thức trên nó như thể nó nằm trong cùng một máy ảo.

14/08/2023

8

Container EE Jakarta

Các loại vùng chứa

Hình 1-5 Máy chủ và container Jakarta EE

14/08/2023

9

Container EE Jakarta

Các loại vùng chứa

Máy chủ và container như sau:

- Máy chủ Jakarta EE: Phần thời gian chạy của sản phẩm Jakarta EE. Một máy chủ Jakarta EE cung cấp các thùng chứa web và doanh nghiệp.
- Jakarta Enterprise Bean container: Quản lý việc thực thi các hạt đậu doanh nghiệp cho các ứng dụng Jakarta EE. Jakarta Enterprise Beans và container của họ chạy trên máy chủ Jakarta EE.
- Bộ chứa web: Quản lý việc thực thi các trang web, servlet và một số thành phần Bean doanh nghiệp cho các ứng dụng Jakarta EE. Các thành phần web và vùng chứa của chúng chạy trên máy chủ Jakarta EE.
- Bộ chứa ứng dụng khách: Quản lý việc thực thi các thành phần ứng dụng khách. Ứng dụng khách và vùng chứa của chúng chạy trên máy khách.
- Thùng chứa Applet: Quản lý việc thực thi các applet. Bao gồm một trình duyệt web và một Plug-in Java chạy trên máy khách cùng nhau.

14/08/2023

10

Container Jakarta EE

Mối quan hệ giữa các container Jakarta EE

Hình 1-6 Các container Jakarta EE

14/08/2023

11

API EE của Jakarta

Web Container	WebSocket	Java SE
	Concurrency Utilities	
	Batch	
	JSON-P	
	Bean Validation	
	EJB Lite	
	EL	
	Jakarta Mail	
	JSP	
	Connectors	
	Jakarta Persistence	
	JMS	
	Management	
	WS Metadata	
	Web Services	
	JACC	
	JASPI	
	JAX-RS	
	JAX-WS	
	JSR	
	JTA	
	CDI	
	Dependency Injection	

New in Jakarta EE

EJB Container	Concurrency Utilities	Java SE
	Batch	
	JSON-P	
	Bean Validation	
	Jakarta Mail	
	Connectors	
	Jakarta Persistence	
	JMS	
	Management	
	WS Metadata	
	Web Services	
	JACC	
	JASPI	
	JAX-RS	
	JAX-WS	
	JTA	
	CDI	
	Dependency Injection	

New in Jakarta EE

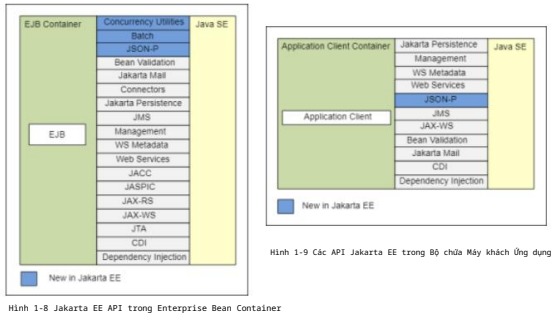
Hình 1-8 Jakarta EE API trong Enterprise Bean Container

Hình 1-7 API Jakarta EE trong Web Container

08/14/2023

12

API EE của Jakarta



13

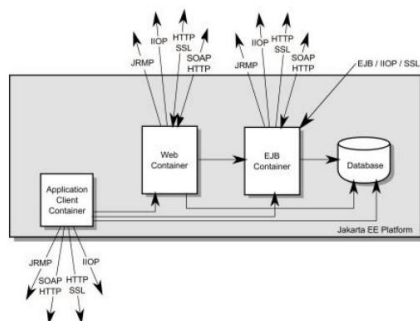
Công nghệ đầu doanh nghiệp Jakarta

Công nghệ đầu doanh nghiệp Jakarta	
Công nghệ Servlet Jakarta	Giấy phép Jakarta
Công nghệ khuôn mặt Jakarta	Xác thực Jakarta
Công nghệ trang máy chủ Jakarta	An ninh Jakarta
Thư viện thể tiêu chuẩn Jakarta	Ổ cắm web Jakarta
Sự kiện tri của Jakarta	Xử lý JSON ở Jakarta
Giao dịch Jakarta	Liên kết JSON của Jakarta
Dịch vụ web RESTful Jakarta	Đồng thời Jakarta
Đầu được quản lý Jakarta	Lô Jakarta
Bối cảnh Jakarta và nội dung phụ thuộc Kích hoạt Jakarta	
Tiêm phụ thuộc Jakarta	Liên kết XML Jakarta
Xác nhận đầu Jakarta	Dịch vụ Web XML của Jakarta
Tin nhắn Jakarta	Jakarta SOAP có tệp đính kèm
Đầu nối Jakarta	Chủ thích Jakarta
thư Jakarta	

Đọc thêm: <https://eclipse-ee4j.github.io/jakartaee-tutorial/>

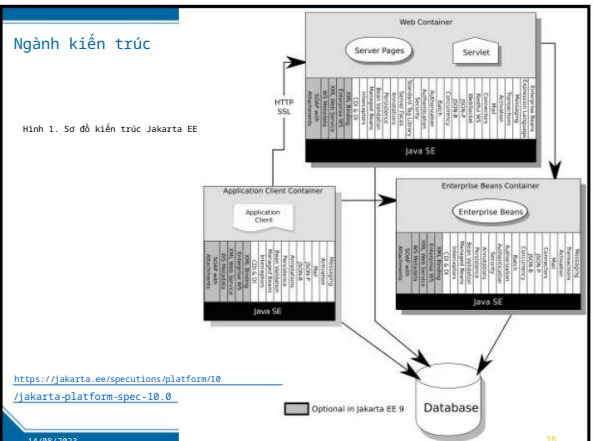
14

Khả năng tương tác API Jakarta EE



15

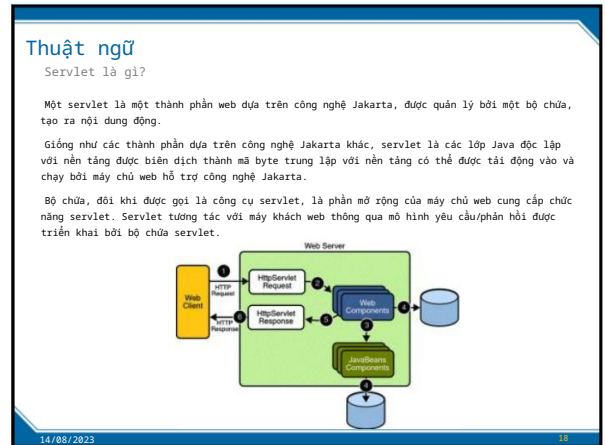
Ngành kiến trúc



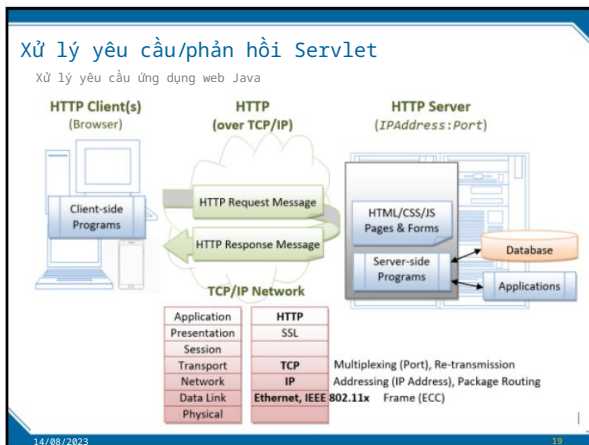
16



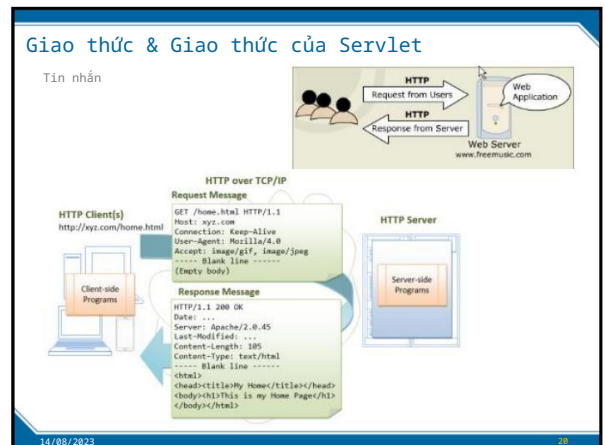
17



18



19



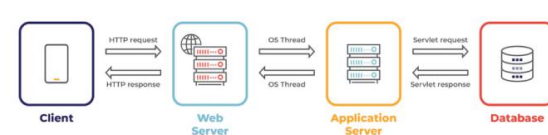
20

Phương thức Http và phương thức Servlet		
HTTP - Ảnh xạ các phương thức Servlet		
Phương thức http	Mô tả	Servlet Phương pháp
Lấy	Phương thức GET yêu cầu biểu diễn tài nguyên đã chỉ định. Các yêu cầu sử dụng GET chỉ nên truy xuất dữ liệu.	doGet
Đưa kiến	Phương thức POST gửi một thực thể tới tài nguyên được chỉ định, thường gây ra thay đổi về trạng thái hoặc tác dụng phụ trên máy chủ.	doPost
Đặt	Phương thức PUT thay thế tất cả các biểu diễn hiện tại của tài nguyên đích bằng tài trọng yêu cầu.	doPut
Xóa bỏ	Phương thức DELETE xóa tài nguyên đã chỉ định.	doDelete
Cất đầu	Phương thức HEAD yêu cầu phản hồi giống hệt với yêu cầu GET, nhưng không có nội dung phản hồi.	doHead
Tùy chọn	Phương thức OPTIONS mô tả các tùy chọn liên lạc cho tài nguyên đích.	doOptions
Dấu vết	Phương thức TRACE thực hiện kiểm tra vòng lặp thông báo dọc theo đường dẫn đến tài nguyên đích.	doTrace
Kết nối	Phương thức CONNECT thiết lập một đường hầm tới máy chủ được xác định bởi tài nguyên đích.	
Và	Phương thức PATCH áp dụng sửa đổi một phần cho tài nguyên.	

21

Thuật ngữ	
Thùng chứa Servlet là gì?	
<p>Bộ chứa servlet là một phần của máy chủ web hoặc máy chủ ứng dụng dùng để cung cấp các dịch vụ mạng qua đó các yêu cầu và phản hồi được thực hiện được gửi, giải mã các yêu cầu dựa trên MIME và định dạng các phản hồi dựa trên MIME. Một thùng chứa servlet cũ ng chứa và quản lý các servlet thông qua vòng đời.</p> <p>Một bộ chứa servlet có thể được tích hợp vào một máy chủ web lưu trữ hoặc được cài đặt như một thành phần tiện ích bổ sung cho máy chủ web thông qua API tiện ích mở rộng riêng của máy chủ đó. Các thùng chứa Servlet cũ ng có thể được tích hợp hoặc có thể được cài đặt vào các máy chủ ứng dụng hỗ trợ web.</p> <p>Tất cả các thùng chứa servlet phải hỗ trợ HTTP làm giao thức cho các yêu cầu và phản hồi, nhưng các giao thức dựa trên yêu cầu/phản hồi bổ sung như HTTPS (HTTP qua SSL) có thể được hỗ trợ.</p>	

22

Thuật ngữ	
Máy chủ web Java so với Máy chủ ứng dụng	
<p>Web Server là một chương trình máy tính chấp nhận yêu cầu dữ liệu và gửi các tài liệu quy định. Máy chủ web có thể là một máy tính chứa nội dung trực tuyến đã giữ. Về cơ bản, máy chủ internet được sử dụng để lưu trữ các trang web tuy nhiên có những khác biệt máy chủ web cũ ng như giải trí, lưu trữ, FTP, email, v.v.</p> <p>Máy chủ ứng dụng bao gồm bộ chứa Web cũ ng như bộ chứa EJB. Ứng dụng máy chủ tổ chức bầu không khí chạy cho các ứng dụng doanh nghiệp. máy chủ ứng dụng có thể là một máy chủ hợp lý nghĩa là cách đặt hệ điều hành, lưu trữ các ứng dụng và dịch vụ cho người dùng, dịch vụ CNTT và tổ chức. Trong đó, giao diện người dùng tương tự như giao thức và giao thức RPC/RMI được sử dụng.</p> 	

23

Máy chủ Web so với Máy chủ ứng dụng		
Loại	Máy chủ web	Máy chủ ứng dụng
1	Máy chủ web chỉ bao gồm vùng chứa web.	Trong khi máy chủ ứng dụng bao gồm bộ chứa Web cũ ng như bộ chứa EJB.
2	Máy chủ web hữu ích hoặc phù hợp với nội dung tĩnh.	Trong khi đó máy chủ ứng dụng được trang bị cho nội dung động.
3	Máy chủ web tiêu thụ hoặc sử dụng ít tài nguyên hơn.	Trong khi máy chủ ứng dụng sử dụng nhiều tài nguyên hơn.
4	Các máy chủ web tập xếp mỗi trường chạy cho các ứng dụng web.	Trong khi các máy chủ ứng dụng tập xếp mỗi trường chạy cho các ứng dụng của doanh nghiệp.
5	Máy chủ web hỗ trợ đa luồng.	Trong khi ở máy chủ ứng dụng, đa luồng không được hỗ trợ.
6	Máy chủ web có dung lượng thấp hơn máy chủ ứng dụng.	Trong khi dung lượng máy chủ ứng dụng cao hơn web máy chủ.
7	Trong máy chủ web, giao thức HTML và HTTP được sử dụng.	Trong khi đó, gửi cũ ng như các giao thức HTTP và RPC/RMI được sử dụng.
8	Các quy trình không sử dụng nhiều tài nguyên được hỗ trợ.	Các quy trình sử dụng nhiều tài nguyên được hỗ trợ.
9	Giao dịch và kết nối tổng hợp không được hỗ trợ. Giao dịch và kết nối tổng hợp được hỗ trợ.	
10	Khả năng chịu lỗi thấp so với các máy chủ ứng dụng.	Nó có khả năng chịu lỗi cao.
11	Ví dụ về Máy chủ Web là Máy chủ HTTP Apache, Tomcat, Jetty, Nginx,...	Ví dụ về Máy chủ ứng dụng là JBoss EAP, WildFly, Glassfish, Apache TomEE, IBM WebSphere,...

14/08/2023

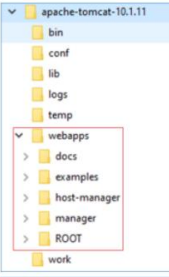
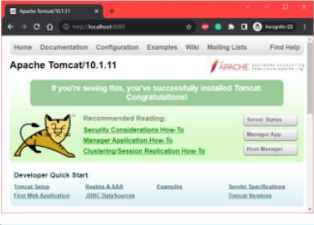
24

24

Máy chủ web Apache Tomcat

Tải xuống: https://tomcat.apache.org/download_10.cgi (v 10.1.11 trở lên)

Cấu hình: - Cài đặt JDK (11/8/17), thêm vào môi trường PATH -
Chỉnh sửa: `conf\tomcat-user.xml` - Khởi động Tomcat: `bin\startup.bat`

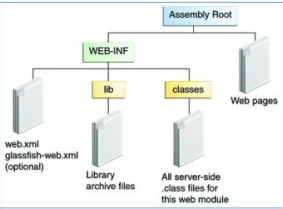


14/08/2023

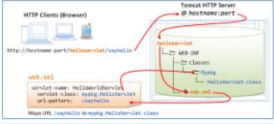
25

Phát triển

Cấu trúc thư mục ứng dụng web



Hình 1. Cấu trúc ứng dụng web Java



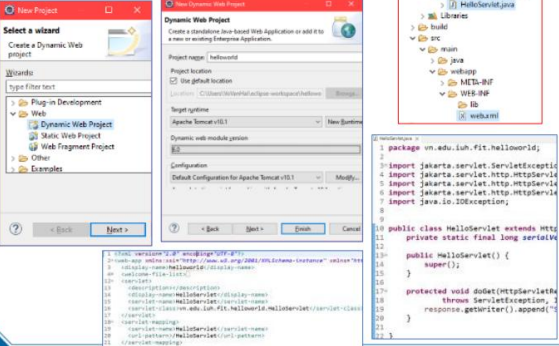
Hình2: Ứng dụng mẫu

14/08/2023

26

Tạo ứng dụng

Với IntelliJ thực

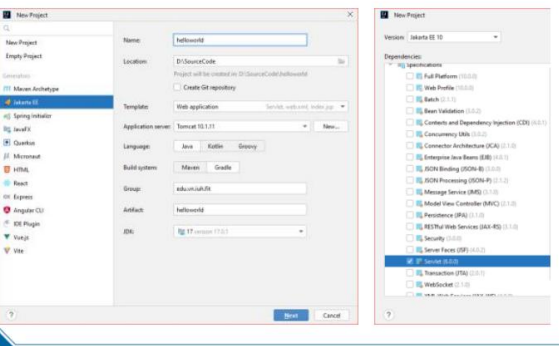


14/08/2023

27

Tạo ứng dụng

Với IntelliJ IDEA



14/08/2023

28

Servlet mẫu

```
1 package edu.vn.luh.fit.helloworld;
2
3 import java.io.*;
4 import jakarta.servlet.http.*;
5 import jakarta.servlet.annotation.*;
6
7 @WebServlet(name = "helloServlet", value = "/hello-servlet")
8 public class HelloServlet extends HttpServlet {
9     private String message;
10
11     public void init() { message = "Hello World!"; }
12
13     public void doGet(HttpServletRequest request,
14                       HttpServletResponse response) throws IOException {
15         response.setContentType("text/html");
16
17         // Hello
18         PrintWriter out = response.getWriter();
19         out.println("<html><body>");
20         out.println("<h1>");
21         out.println("</body></html>");
22     }
23
24     public void destroy() {
25     }
26 }
```

```
1 <? page contentType="text/html; charset=
2 <DOCTYPE html>
3 <html>
4 <head>
5 <title>JSP - Hello World</title>
6 </head>
7 <body>
8 <h1> "Hello World!" %>
9 </h1>
10 <br/>
11 <a href="hello-servlet">Hello Servlet</a>
12 </body>
13 </html>
```

29




Tạo dự án bằng công cụ Build

Maven:

```
<dependency>
<groupId>jakarta.servlet</groupId>
<artifactId>jakarta.servlet-api</
artifactId>
<version>6.0.0</version> </
dependency>
```

Lớp:

```
biên dịchOnly('jakarta.servlet:jakarta.servlet-api:6.0.0')
```



30

Vòng đời của Servlet Jakarta

Vòng đời chung của Servlet

Vòng đời được xác định bởi:

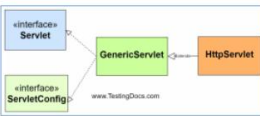
- init() - chỉ được gọi bởi máy chủ trong yêu cầu đầu tiên
- service() - xử lý yêu cầu của máy khách
- hủy() - được gọi sau khi tất cả các yêu cầu đã được xử lý hoặc một số cụ thể của máy chủ giấy đã trôi qua

Hình: Vòng đời của Servlet


31

Các loại Servlet

Jakarta.servlet.GenericServlet



```
<!-- Servlet -->
<!-- ServletConfig -->
<!-- HttpServlet -->
```



```
public void init(ServletConfig config) throws ServletException {
    super.init(config);
    initParam = config.getInitParameter("initParam");
}

@Override
public void service(ServletRequest request, ServletResponse response) throws ServletException {
    String name = request.getParameter("name");
    response.getWriter().println("initParam = " + initParam);
}
```

32

[illegible]

33

Xử lý các sự kiện trong vòng đời của Servlet

Bạn có thể theo dõi và phản ứng với các sự kiện trong vòng đời của servlet bằng cách xác định các đối tượng người nghe có phương thức được gọi khi các sự kiện trong đời xảy ra.

```

@WebServlet()
public class SimpleServletListener implements ServletContextListener,
    ServletContextAttributeListener, HttpSessionListener {

```

Sự kiện trong đời của Servlet		
Sự vật	Sự kiện	Giao diện người nghe và lớp sự kiện
Bối cảnh web	Khởi tạo và phá hủy	<code>jakarta.servlet.ServletContextListener</code> và <code>ServletBối cảnh</code> kiện
Web bối cảnh	Thuộc tính được thêm, xóa hoặc thay thế	<code>jakarta.servlet.ServletContextAttributeListener</code> và <code>ServletContextAttributeEvent</code>
Phiên họp	Tạo, vô hiệu hóa, kích hoạt, thu đóng và hết thời gian	<code>jakarta.servlet.http.HttpSessionListener</code> , <code>jakarta.servlet.http.SessionCreationListener</code> và <code>HttpSessionEvent</code>
Phiên họp	Thuộc tính được thêm, xóa hoặc thay thế	<code>jakarta.servlet.http.HttpSessionAttributeListener</code> và <code>HttpSessionBindingEvent</code>
Lời yêu cầu	Một yêu cầu servlet đã bắt đầu được xử lý bởi các thành phần web	<code>jakarta.servlet.ServletRequestListener</code> và <code>ServletRequestYêu cầu</code> kiện
Lời yêu cầu	Thuộc tính được thêm, xóa hoặc thay thế	<code>jakarta.servlet.ServletRequestAttributeListener</code> và <code>ServletRequestAttributeEvent</code> kiện

34

Chia sẻ thông tin

Sử dụng đối tượng phạm vi

Các thành phần web cộng tác chia sẻ thông tin qua các đối tượng được duy trì như các thuộc tính của bốn đối tượng phạm vi. Ban truy cập những được trình bằng cách sử dụng các phương thức `getAttribute` và `setAttribute` của lớp đại diện cho phạm vi.

Đối tượng phạm vi	
Lớp đối tượng phạm vi	Có thể truy cập từ
Bối cảnh web	Các thành phần web trong ngữ cảnh web. Xem [truy cập-ngữ cảnh web].
Phiên họp	Các thành phần web xử lý một yêu cầu thuộc về phiên. Xem [duy trì-máy khách-trang thái].
Lời yêu cầu	Các thành phần web xử lý yêu cầu.
Trang	Trang Jakarta Server Pages tạo đối tượng.

35

Chia sẻ thông tin

Kiểm soát quyền truy cập đồng thời vào tài nguyên được chia sẻ

Trong một máy chủ đa luồng, các tài nguyên được chia sẻ có thể được truy cập đồng thời. TRONG Ngoài, các thuộc tính đối tượng phân vụ, các tài nguyên được chia sẻ bao gồm dữ liệu trong bộ nhớ, chẳng hạn như các biến thể hiện hoặc lớp và các đối tượng bên ngoài, chẳng hạn như các tệp, cơ sở dữ liệu kết nối và kết nối mạng.

Truy cập đồng thời có thể phát sinh trong một số trường hợp.

- Nhiều thành phần web truy cập các đối tượng được lưu trữ trong ngữ cảnh web.
- Nhiều thành phần web truy cập các đối tượng được lưu trữ trong một phần mềm.
- Nhiều thành phần web truy cập các đối tượng chia sẻ hiện diện.

Vùng chứa web thường sẽ tạo một luồng để xử lý từng yêu cầu. Băm báo rằng một phiên bản servlet chỉ xử lý một yêu cầu tại một thời điểm, các tài nguyên chia sẻ đơn SingleThreadModel. Nếu một servlet thực hiện điều này giao diện, không có bất kỳ luồng nào sẽ thực thi đồng thời trong dịch vụ của servlet phương pháp. Vùng chứa web có thể triển khai bảo đảm này bằng cách đồng bộ hóa quyền truy cập đến một phiên bản duy nhất của servlet hoặc bằng cách duy trì một nhóm thành phần web các phiên bản và gửi từng yêu cầu mới đến một phiên bản miễn phí. Giao diện này làm không ngăn chặn các sự cố đồng bộ hóa do các thành phần web gây ra'.

Truy cập các tài nguyên được chia sẻ, chẳng hạn như các biến lớp tĩnh hoặc các đối tượng bên ngoài.

14/08/2022

35

36

Xử lý không đồng bộ

Giới thiệu

Bộ chứa web trong máy chủ ứng dụng thường sử dụng luồng máy chủ cho mỗi máy khách lời yêu cầu.

Trong điều kiện tải nặng, container cần số lượng thread lớn để phục vụ tất cả các yêu cầu của khách hàng.

Các hạn chế về khả năng mở rộng bao gồm hết bộ nhớ hoặc làm cạn kiệt nhóm luồng trong vùng chứa.

Để tạo các ứng dụng web có thể mở rộng, bạn phải đảm bảo rằng không có chuỗi nào liên quan đến yêu cầu đang ở trạng thái chờ, để bộ chứa có thể sử dụng chúng để xử lý các yêu cầu mới.

Có hai tình huống phổ biến trong đó một luồng được liên kết với yêu cầu có thể ở trạng thái không hoạt động:

- Chuỗi cần đợi tài nguyên khả dụng hoặc xử lý dữ liệu trước khi xây dựng phản hồi. Ví dụ: một ứng dụng có thể cần truy vấn cơ sở dữ liệu hoặc truy cập dữ liệu từ một dịch vụ web từ xa trước khi tạo phản hồi.
- Chuỗi cần đợi một sự kiện trước khi tạo phản hồi. Ví dụ: một ứng dụng có thể phải đợi một tín hiệu từ Jakarta Messaging, thông tin mới từ ứng dụng khách khác hoặc dữ liệu mới có sẵn trong hàng đợi trước khi tạo phản hồi.

14/08/2023

37

Xử lý không đồng bộ

Xử lý không đồng bộ trong Servlet

Để bật xử lý không đồng bộ trên servlet, hãy đặt tham số `asyncSupported` thành true trên chú thích `@WebServlet` như sau:

```
@WebServlet(urlPatterns={"/asyncservlet"}, asyncSupported=true)
public class AsyncServlet extends HttpServlet { ... }
```

Lớp `jakarta.servlet.AsyncContext` cung cấp chức năng mà bạn cần để thực hiện xử lý không đồng bộ bên trong các phương thức dịch vụ. Để lấy một phiên bản của `AsyncContext`, hãy gọi phương thức `startAsync()` trên đối tượng yêu cầu của phương thức dịch vụ của bạn:

```
public void doGet(HttpServletRequest req, HttpServletResponse resp) {
    ...
    AsyncContext acontext = req.startAsync();
    ...
}
```

Lớp `AsyncListener` cung cấp chức năng mà bạn có thể sử dụng để nghe

trạng thái của chủ đề không đồng bộ.

```
lớp công khai MyAsyncListener triển khai AsyncListener {
    //...
}
```

14/08/2023

38

Xử lý không đồng bộ

Ví dụ

```
@WebServlet(name = "myAsyncServlet", urlPatterns = {"/async-test"},
    asyncSupported = true)
public class MyAsyncServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req,
        HttpServletResponse resp) throws ServletException {
        PrintWriter writer = resp.getWriter();
        AsyncContext asyncContext = req.startAsync();
        asyncContext.addListener(new MyAsyncListener());

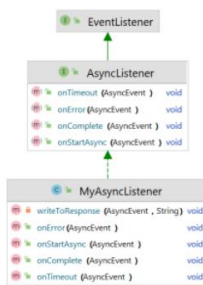
        asyncContext.start(new Runnable() {
            @Override
            public void run() {
                String msg = task();
                writer.println(msg);
                asyncContext.complete();
            }
        });
    }

    private String task() { ... }
}
```

Đọc thêm: <https://github.com/eclipse-ee4j/jakartaee-tutorial/blob/master/src/main/asciidoc/servlets/servlets1812.asciidoc>

14/08/2023

39



Bối cảnh và chèn phụ thuộc (CDI)

Giới thiệu

CDI (Contexts and Dependency Inject) là một khung chèn phụ thuộc tiêu chuẩn có trong Java EE 6 trở lên.

Nó cho phép chúng tôi quản lý vòng đời của các thành phần có trạng thái thông qua các bối cảnh vòng đời cụ thể của miền và đưa các thành phần (dịch vụ) vào các đối tượng máy khách theo cách an toàn về loại.

CDI

- Bối cảnh: Khả năng liên kết vòng đời và sự tương tác của các thành phần có trạng thái với bối cảnh vòng đời được xác định rõ ràng nhưng có thể mở rộng.
- Tiêm phụ thuộc: Khả năng đưa các thành phần vào một ứng dụng theo cách an toàn về loại, bao gồm khả năng chọn tại thời điểm triển khai thực hiện một giao diện cụ thể để đưa vào.

Trong Máy chủ Tomcat, cần thêm tham chiếu đến phụ thuộc mỗi hán jboss

```
dependencies {
    compileOnly('jakarta.servlet:jakarta.servlet-api:5.0.0')
    implementation('org.jboss.weld.servlet:weld-servlet-core:5.1.1.Final')
}

compileOnly('org.jboss.weld.servlet:weld-servlet-core:5.1.1.Final')
```

Đọc thêm:

<https://github.com/eclipse-ee4j/jakartaee-tutorial/tree/master/src/main/asciidoc/cdi-basic>

14/08/2023

40

Bối cảnh và chèn phụ thuộc (CDI)

Vi dụ

```
public class World {
    public String world() {
        return "World";
    }
}
```

```
import jakarta.inject.Inject;

public class Hello {
    @Inject
    private World world;

    public String helloWorld() {
        return "Hello " + world.world() + "!";
    }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/JavaServletWorld">
  <Resource name="BeanManager"
    auth="Container"
    type="jakarta.enterprise.inject.api.BeanManager"
    factory="org.jboss.weld.resources.ManagerObjectFactory" />
</Context>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="https://jakarta.ee/xml/ns/jakartaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
    https://jakarta.ee/xml/ns/jakartaee/beans_4.0.xsd"
  bean-discovery-mode="all">
</beans>
```

```
import jakarta.servlet.*;
import jakarta.servlet.http.*;

public class CiaoServlet extends HttpServlet {
    @Inject
    private Hello hello;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        try {
            out.println(hello.helloWorld());
        }
    }
}
```

Ciao Mondo

41

API REST

42

Dịch vụ web RESTful

Giới thiệu

Các dịch vụ web RESTful được kết hợp lỏng lẻo, các dịch vụ web nhẹ đặc biệt phù hợp để tạo API cho khách hàng trải rộng khắp Internet.

Chuyển trạng thái đại diện (REST) là một kiểu kiến trúc của ứng dụng máy chủ khách tập trung vào việc chuyển giao các đại diện của tài nguyên thông qua các yêu cầu và phản hồi.

Trong phong cách kiến trúc REST, dữ liệu và chức năng được xem xét tài nguyên và được truy cập bằng Mã định danh tài nguyên thống nhất (URI), thường là các liên kết trên Web.

Các tài nguyên được thể hiện bằng các tài liệu và được thực hiện bằng cách sử dụng một tập hợp các hoạt động đơn giản, được xác định rõ.

43

Tài nguyên gốc RESTful

Phát triển dịch vụ web RESTful với Jakarta REST

Jakarta REST là API ngôn ngữ lập trình Java được thiết kế để giúp dễ dàng để phát triển các ứng dụng sử dụng kiến trúc REST.

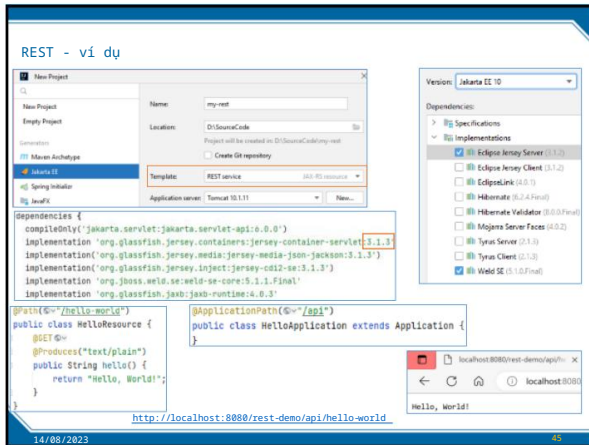
Jakarta REST API sử dụng chú thích ngôn ngữ lập trình Java để đơn giản hóa việc phát triển các dịch vụ web RESTful.

Các nhà phát triển trang trí file lớp ngôn ngữ lập trình Java bằng Jakarta Chú thích REST để xác định tài nguyên và các hành động có thể được được thực hiện trên các tài nguyên đó.

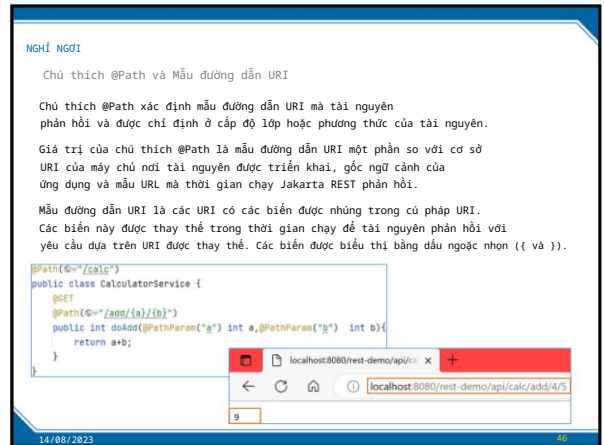
Jakarta REST Chú thích		
@Con đường	@VA	@Consumes
@Lấy	@CÁI ĐẦU	@Sản xuất
@ĐUỔI KIẾN	@TUY CHỌN	@Các nhà cung cấp
@ĐÀT	@PathParam	@ApplicationPath
@ĐỎA BỎ	@QueryParam	

<https://github.com/eclipse-ee4j/jakartaee-tutorial/blob/master/src/main/asciidoc/jaxrs/jaxrs002.adoc>

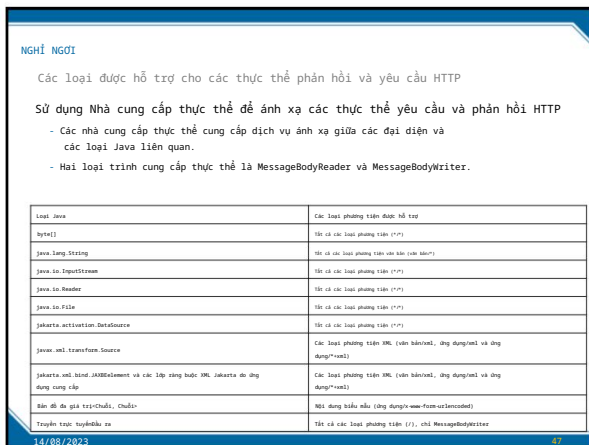
44



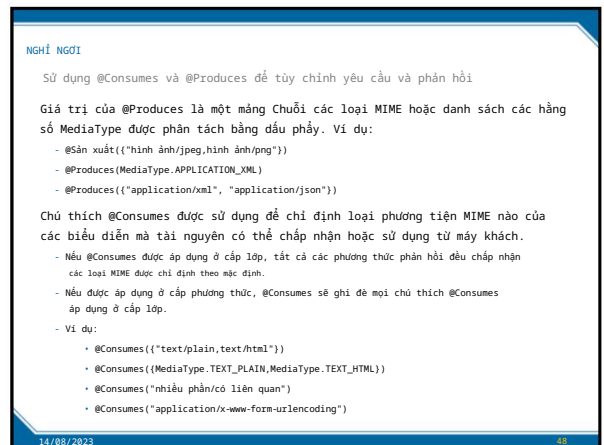
45



46



47



48

NGHĨ NGÔI

Trích xuất tham số yêu cầu

Các tham số của một phương thức tài nguyên có thể được chú thích bằng các chú thích dựa trên tham số để trích xuất thông tin từ một yêu cầu. Ví dụ trước đã trình bày cách sử dụng tham số `@PathParam` để trích xuất tham số đường dẫn từ thành phần đường dẫn của URL yêu cầu khớp với đường dẫn được khai báo trong `@Path`. Bạn có thể

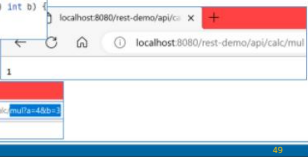
trích xuất các loại tham số sau để sử dụng trong tài nguyên của mình

Lớp: Truy vấn, đường dẫn URI, Biểu mẫu, Cookie, Tiêu đề, Mã trận

```

@GET
@Path("/{mul}")
public int doMul(@DefaultValue("1") @QueryParam("a") int a,
                @DefaultValue("1") @QueryParam("b") int b) {
    return a * b;
}

```



14/08/2023

49

NGHĨ NGÔI

Định cấu hình ứng dụng Jakarta REST

Tạo một lớp con của `jakarta.ws.rs.core.Application` để định cấu hình thủ công môi trường trong đó chạy các tài nguyên REST được xác định trong các lớp tài nguyên của bạn, bao gồm cả URI cơ sở. Thêm chú thích `@ApplicationPath` cấp lớp để đặt URI cơ sở.

```

@ApplicationPath("/{api}")
public class HelloApplication extends Application {}

```

tất cả tài nguyên được xác định trong ứng dụng đều liên quan đến /

api Theo mặc định, tất cả tài nguyên trong kho lưu trữ sẽ được xử lý cho tài nguyên. Ghi đè phương thức `getClasses` để đăng ký thủ công các lớp tài nguyên trong ứng dụng với thời gian chạy Jakarta REST.

```

@ApplicationPath("/{api}")
public class HelloApplication extends Application {
    @Override
    public Set<Class?> getClasses() {
        final Set<Class?> classes = new HashSet<>();
        // register root resource
        classes.add(HelloResource.class);
        return classes;
    }
}

```

14/08/2023

50

Máy khách REST

```

public static void main(String[] args) {
    Client client = ClientBuilder.newClient();

    // WebTarget wt1 = client.target("http://localhost:8080/rest-demo/api/calc/mul?a=4&b=3");
    WebTarget wt1 = client.target("http://localhost:8080/rest-demo/api/calc/mul")
        .queryParam("a", "4")
        .queryParam("b", "3");
    Response response1 = wt1.request().accept(MediaType.TEXT_PLAIN).get();
    String s1 = response1.readEntity(String.class);

    // WebTarget wt2 = client.target("http://localhost:8080/rest-demo/api/calc/add/1/4");
    WebTarget wt2 = client.target("http://localhost:8080/rest-demo/api/calc/add")
        .path("3")
        .path("4");
    Response response2 = wt2.request().accept(MediaType.TEXT_PLAIN).get();
    String s2 = response2.readEntity(String.class);

    System.out.println("-----" + s1);
    System.out.println("-----" + s2);
}

```

14/08/2023

51

Trang máy chủ Java

52

14/08/2023

52

Trang máy chủ Java

Giới thiệu

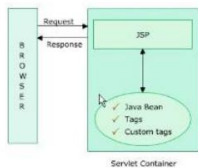
Trang máy chủ Java (JSP) là ngôn ngữ tập lệnh phía máy chủ

Được lưu với phần mở rộng .jsp

Một công nghệ Java đơn giản nhưng mạnh mẽ để tạo và duy trì các trang web có nội dung động

Trang JSP được bộ chứa web chuyển đổi thành một phiên bản Servlet

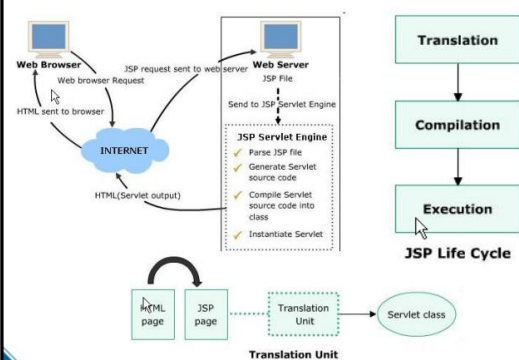
Tập trung vào logic trình bày của ứng dụng web



14/08/2023

53

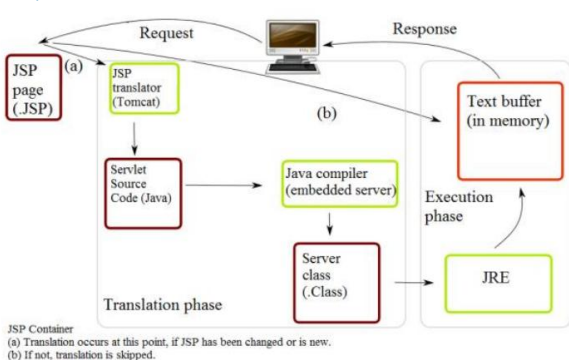
Kiến trúc JSP



14/08/2023

54

Thực thi JSP



14/08/2023

55

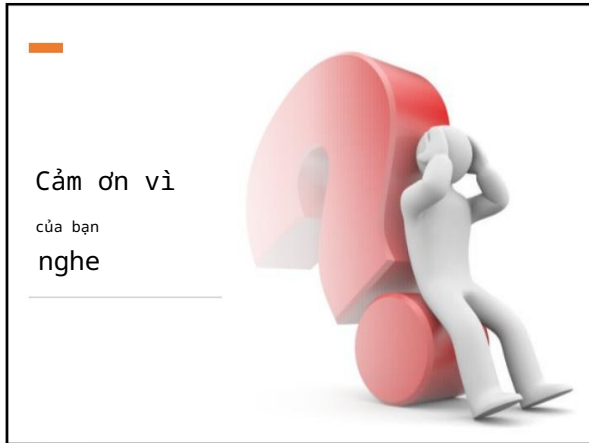
Tự học

Đọc thêm

<https://github.com/eclipse-ee4j/jakartaee-tutorial>

14/08/2023

56



57