BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC HUẾ

**TRƯỜNG ĐẠI HỌC KHOA HỌC**

HỒ NGUYỄN THÀNH NHÂN

TÌM HIỂU VỀ KIẾN TRÚC HƯỚNG DỊCH VỤ TRONG LĨNH VỰC CÔNG NGHỆ PHẦN MỀM VÀ ỨNG DỤNG

CHUYÊN NGÀNH:

MÃ SỐ:

KHOA HỌC MÁY TÍNH

60 48 01 01

LUẬN VĂN THẠC SĨ KHOA HỌC   
CÔNG NGHỆ THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC

PGS. TS. HOÀNG HỮU HẠNH

Huế, 2016

# MỞ ĐẦU

Với sự phát triển của internet và với xu thế hội nhập chung của toàn thế giới, các tổ chức, doanh nghiệp bắt tay, phối hợp hoạt động và chia sẻ tài nguyên với nhau để nâng cao hiệu quả hoạt động. Khi đó các sản phẩm sẽ có độ phức tạp lớn hơn, kéo theo các vấn đề liên quan như chi phí sản xuất, chi phí quản lý và bảo trì. Bên cạnh đó, ngành công nghệ phần mềm còn phải đối mặt với các khó khăn trong xu thế mới như vấn đề tái sử dụng và mở rộng các hệ thống sẵn có, vấn đề về sự không tương thích giữa các hệ thống khác nhau của nhiều tổ chức...

Để giải quyết vấn đề trên, nhiều giải pháp đã được nghiên cứu và ứng dụng, nhưng hầu hết đều không giải quyết các khó khăn một cách triệt để và kết quả đạt được cũng không như mong đợi. Một giải pháp đang được cộng đồng CNTT rất quan tâm, đó là “Kiến trúc hướng dịch vụ” (Service-oriented Architecture – SOA. Đó là lý do tôi chọn hướng nghiên cứu của luận văn là **“Tìm hiểu về kiến trúc hướng dịch vụ trong lĩnh vực công nghệ phần mềm và ứng dụng”.**

Mục đích chính của luận văn là tìm một giải pháp để hỗ trợ cho việc lập trình một ứng dụng theo kiến trúc hướng dịch vụ.

Trong luận văn này, tôi đề xuất một phương pháp để hỗ trợ việc lập trình theo kiến trúc hướng dịch vụ đó là xây dựng một Services Bus Plug-in dựa trên công nghệ Eclipse.

**CẤU TRÚC CỦA LUẬN VĂN**

**Chương 1: Tổng quan về kiến trúc hướng dịch vụ**

Chương này trình bày các vấn đề cơ bản về kiến trúc hướng dịch vụ và các công nghệ liên quan. Đồng thời cũng nêu ra những nét chính, cái nhìn tổng quan về kiến trúc hướng dịch vụ.

**Chương 2: Khung ứng dụng hỗ trợ lập trình SOA**

Đây là chương trình bày về kiến trúc nền tảng Eclipse, các cơ chế và kiến trúc plug-in của Eclipse hỗ trợ cho việc lập trình theo kiến trúc hướng dịch vụ.

**Chương 3: Xây dựng ứng dụng trên nền tảng Eclipse**

Để xây dựng nên một ứng dụng theo kiến trúc hướng dịch vụ thì cách phân tích, thiết kế xây dựng các ứng dụng phần mềm lấy dịch vụ (service) làm trung tâm, làm thành phần cơ bản và cốt lõi. Kiến trúc này hỗ trợ service giao tiếp với nhau, có thể tái sử dụng và kết hợp với nhau để tạo thành các quy trình nghiệp vụ phục vụ cho lợi ích của người sử dụng. Vậy để xây dựng nên một ứng dụng theo kiến trúc hướng dịch vụ thì ta cần và có thể sử dụng nền tảng hay khung ứng dụng nào phù hợp? Ngôn ngữ lập trình nào? Chương này sẽ giới thiệu một giải pháp để xây dựng ứng dụng theo hướng kiến trúc hướng dịch vụ.

# Chương 1: Tổng quan về kiến trúc hướng dịch vụ

## Công nghệ Web Services

### Tổng quan về Web Services

Web service là một hệ thống phần mềm được thiết kế để hỗ trợ khả năng tương tác giữa các ứng dụng trên các máy tính khác nhau thông qua mạng Internet, giao diện chung và sự gắn kết của nó được mô tả bằng XML.

Xuất bản

Gởi thông điệp

Tìm kiếm

**Đăng ký dịch vụ (UDDI)**

**Dịch vụ yêu cầu**

**Dịch vụ cung cấp**

Mô tả dịch vụ (WSDL)

Hình 1.1 Cơ chế hoạt động của Web Services

### Kiến trúc của Web Services

Kiến trúc của Web Services bao gồm các tầng như sau:



Hình 1.2 Kiến trúc của Web Services

### Các thành phần của Web Services

#### XML

#### WSDL – Web Services Description Language

#### UDDI – Universal Description, Discovery, and Integration

#### SOAP – Simple Object Access Protocol

## Kiến trúc hướng dịch vụ

### Kiến trúc hướng dịch vụ (SOA) là gì?

Kiến trúc hướng dịch vụ - SOA (Service Oriented Architecture) là một cách tiếp cận hay một phương pháp luận để thiết kế và tích hợp các thành phần khác nhau, bao gồm các phần mềm và các chức năng riêng lẻ lại thành một hệ thống hoàn chỉnh. Kiến trúc SOA rất giống với cấu trúc của các phần mềm hướng đối tượng gồm nhiều module. Tuy nhiên khái niệm module trong SOA không đơn thuần là một gói phần mềm, hay một bộ thư viện nào đó. Thay vào đó, mỗi module trong một ứng dụng SOA là một dịch vụ được cung cấp rải rác ở nhiều nơi khác nhau và có thể truy cập thông qua môi trường mạng, và mỗi module đóng vai trò là một “dịch vụ có tính kết nối lõng lẻo”. Nói một cách ngắn gọn, một hệ thống SOA là một tập hợp nhiều dịch vụ được cung cấp trên mạng, được tích hợp lại với nhau để cùng cộng tác thực hiện các tác vụ nào đó theo yêu cầu của khác hàng.

Dịch vụ (Service) là yếu tố then chốt trong SOA. Có thể hiểu dịch vụ như là một loại module thực hiện một quy trình nghiệp vụ nào đó

Register

Find

**Bind,**

**Execute**

Hình 1.10 Mô hình tổng quan của SOA

### Các nguyên tắc chính của hệ thống SOA

#### Sự phân định rạch ròi giữa các dịch vụ

#### Các dịch vụ tự hoạt động

#### Các dịch vụ chia sẻ lược đồ

#### Tính tương thích của các dịch vụ dựa trên chính sách

### Các tính chất của một hệ thống SOA

#### Kết nối lõng lẻo

#### Tái sử dụng dịch vụ

#### Quản lý chính sách

#### Tự động dò tìm và rành buộc động

#### Khả năng tự phục hồi

#### Khả năng cộng tác

### Kiến trúc phân tầng chi tiết của SOA

Nhấn mạnh rằng, SOA là một phương pháp luận giúp chúng ta tận dụng sức mạnh của các nguồn lực, nguồn tài nguyên khác nhau trong mạng máy tính để trở thành một hệ thống nhất. Nên hiện tại không có mô hình thống nhất cho các thành phần của hệ thống SOA, mỗi công ty, tổ chức khi phát triển một hệ thống SOA có thể đưa ra mô hình các thành phần của SOA khác nhau. Đây là mô hình các thành phần của hệ thống SOA theo quan điểm của công ty IBM và đây cũng là một mô hình khá phổ biến cho kiến trúc của hệ thống SOA.



Hình 1.12 Kiến trúc phân tầng của SOA

## Quy trình xây dựng SOA

Hiện nay, chưa có một quy trình cụ thể để phát triển các ứng dụng theo kiến trúc hướng dịch vụ. Tuy nhiên, dựa trên thực tế, 12 bước sau đã được đưa ra nhằm tham khảo khi quyết định chuyển sang định hướng dịch vụ.

12 bước trong quy trình xây dựng ứng dụng theo kiến trúc hướng dịch vụ:

1. Hiểu nghiệp vụ
2. Xác định phạm vi (miền) của vấn đề
3. Hiểu tất cả các ngữ nghĩa ứng dụng trong miền đó
4. Hiểu tất cả các dịch vụ hiện có trong miền
5. Hiểu tất cả các nguồn và đích của thông tin có trong miền
6. Hiểu tất cả các quy trình trong miền
7. Xác định và phân loại tất cả các giao diện bên ngoài miền cần thiết cho việc xây dựng ứng dụng (các dịch vụ và thông tin)
8. Định nghĩa các dịch vụ mới và các ràng buộc thông tin của các dịch vụ đó.
9. Định nghĩa các quy trình mới, cũng như các dịch vụ và ràng buộc thông tin cho các quy trình này.
10. Lựa chọn tập công nghệ.
11. Triển khai công nghệ SOA.
12. Kiểm thử và đánh giá.

## Ngôn ngữ thi hành quy trình nghiệp vụ - BPEL

### Giới thiệu

Web Service Business Process Execution Language (viết tắt là WS-BPEL hay được gọi là BPEL) là một ngôn ngữ thi hành quy trình nghiệp vụ dùng để hỗ trợ phát triển các ứng dụng phức tạp, lớn đòi hỏi phải tổng hợp nhiều web services khác nhau.

### Các khái niệm cơ bản

#### Nguyên tắc hoạt động của một tiến trình BPEL

#### Cấu trúc của một tiến trình

#### Các thành phần và ký hiệu trong BPEL

## Tiểu kết chương 1

Qua các kiến thức tổng quan ta đã thấy rõ ràng kiến trúc hướng dịch vụ (SOA) là một kiểu kiến trúc có khả năng tái sử dụng lại các tài nguyên sẵn có, khả năng mở rộng và liên kết tốt với các hệ thống mới để tạo nên một môi trường đồng nhất, nó bao gồm các dịch vụ nghiệp vụ độc lập, không đồng nhất được kết hợp với nhau trong quy trình nghiệp vụ linh hoạt mềm dẻo. Công nghệ Web service cùng với ngôn ngữ thi hành quy trình nghiệp vụ - BPEL đã hiện thực hóa kiến trúc hướng dịch vụ (SOA), cho phép kết hợp các dịch vụ đơn lẻ và các hệ thống ứng dụng thành một quy trình nghiệp vụ đầy đủ.

# Chương 2: Khung ứng dụng hỗ trợ lập trình SOA



## Nền tảng Eclipse

### Giới thiệu

Eclipse là một nền tảng phát triển phần mềm mã nguồn mở, bao gồm một IDE (Intergrated Development Environment) và một hệ thống plug-in có khả năng mở rộng được. Trong khi hầu hết người dùng sử dụng Eclipse như là môi trường phát triển tích hợp Java (IDE), một số khác lại có nhu cầu mở rộng Eclipse. Do đó, Eclipse xây dựng PDE (Plug-in Development Environment) dành cho người dùng muốn mở rộng Eclipse. PDE cho phép người xây dựng các công cụ tích hợp vào môi trường Eclipse và mọi thứ trong PDE đều là plug-in.

Download, cài đặt Eclipse: [*http://www.Eclipse.org/downloads/*](http://www.eclipse.org/downloads/)

### Các thành phần và kiến trúc

Kiến trúc của Eclipse được xây dựng dựa trên hai thành phần chính: thành phần lõi (*core*) và các thành phần gắn thêm (*plug-in*).

Thành phần lõi (core): bao gồm các chức năng, dịch vụ mà các hệ phát triển ứng dụng phải có như chức năng cung cấp giao diện, trình soạn thảo văn bản, gỡ lỗi… cần cho mọi nền tảng lập trình (cần cho các plug-in).

Thành phần gắn thêm (plug-in): bao gồm nhiều thành phần dễ dàng tích hợp vào nhiều ứng dụng chạy trên nền Eclipse. Các chức năng của thành phần core tách biệt với các chức năng của phần giao diện.

*EclipseSDK* bao gồm 3 phần chính:

Platform

Java Development Toolkit (JDT)

Plug-in Development Environment (PDE)

Với *JDT*, *Eclipse* là một môi trường hỗ trợ phát triển Java. JDT cũng có thể được coi như là một *Plug-in* cho *Eclipse* như là một môi trường lập trình tích hợp (*Java IDE -Integrated Development Enviroment*). PDE hỗ trợ việc mở rộng *Eclipse*, tích hợp các *Plug-in* vào *Eclipse Platform*. *Eclipse Platform* là nền tảng của toàn bộ phần mềm *Eclipse*, cung cấp những dịch vụ cần thiết cho việc tích hợp những bộ công cụ phát triển phần mềm dưới dạng *Plug-in*. *Eclipse Platform* được xây dựng dựa trên cơ chế phát hiện, tích hợp và chạy các thành phần gắn thêm. *Plug-in* là thành phần gắn thêm có thể hoạt động độc lập hoặc cùng với các thành phần khác. Thông thường, một ứng dụng có thể tổ hợp nhiều thành phần gắn thêm, mỗi thành phần gắn thêm này lại nối với các thành phần gắn thêm khác tùy vào mục đích sử dụng



Hình 2.1 Kiến trúc tổng quan Eclipse

## Kiến trúc plug-in của Eclipse

### Giới thiệu về plug-in và extension point

Plug-in còn được gọi là trình cắm, hay phần bổ trợ là một thành phần giúp phần mềm ứng dụng thêm những tính năng cụ thể. Nếu được hỗ trợ, Plug-in cho phép tùy biến các chức năng của một ứng dụng. Plug-in có thể chứa mã nguồn, tài nguyên hay cả hai.

Extension point là 1 cơ chế cho phép 1 plug-in có thể thêm các chức năng từ 1 plug-in khác.

Extension là một chức năng

Mỗi Plug-in có một hay nhiều extension point và có thể tùy ý khai báo một điểm mở rộng mới (extension poit). Mỗi plug-in phụ thuộc vào một hoặc nhiều plug-in khác.

Hình 2.2 Plug-in và extension point trong eclipse

Về mặt kỹ thuật, một plug-in là một Java Archive (JAR) độc lập và tự khởi tạo. Plug-in độc lập vì plug-in chứa mã và tài nguyên các plug-in khác cần để chạy. Plug-in tự khởi tạo bởi vì nó bao gồm các thông tin của plug-in, tài nguyên plug-in, công dụng của nó. Trong một plug-in, hai tập tin mô tả là MANIFEST.MF và Plug-in.xml.

### Tập tin cấu hình (Manifest) của Plug-in

Thông tin mô tả chi tiết plug-in nằm trong tập tin manifest (plug-in.xml). Eclipse sử dụng tập tin manifest để tích hợp plug-in vào framework. 

Hình 2.3 Tập tin plug-in manifest

### Vòng đời của plug-in

Khi khởi động Eclipse sẽ xem trong thư mục plug-ins và tạo 1 danh sách các plug-in, danh sách này được gọi là plug-in registry (danh sách được tạo dựa trên các file manifest). Plug-in được load khi chúng ta gọi nó và chỉ unload khi Eclipse đóng.

## Tiểu kết chương 2

Kiến trúc nền tảng Eclipse bao gồm nhiều thành phần. Các chức năng của thành phần lõi tách biệt với các chức năng của phần giao diện. Eclipse cung cấp các công cụ cùng cơ chế sử dụng và quy định để có thể tích hợp các công cụ một cách dễ dàng. Những cơ chế này được thể hiện thông qua giao diện API, lớp và phương thức. Ngoài ra, nền tảng Eclipse còn cung cấp các framework để dễ dàng phát triển các công cụ mới.

Eclipse SDK (Software development kit) gồm Eclipse Platform, JDT và PDE. JDT và PDE là các plug-in được gắn vào Platform. Eclipse Platform được xây dựng dựa trên cơ chế “*phát hiện, tích hợp và chạy*”. Plug-in là đơn vị chức năng nhỏ nhất có thể được phát triển và phân phối một cách riêng biệt. Thông thường, công cụ nhỏ được viết trong một plug-in riêng biệt và công cụ phức tạp được tách ra thành nhiều plug-in.

Từ kiến trúc của Eclipse có thể dễ dàng thấy các ưu điểm của nó :

Có khả năng mở rộng, phụ thuộc vào các thành phần gắn thêm như cho ngôn ngữ mới, cho bộ xử lý mới

Ứng dụng được cho việc phát triển mọi kiểu ứng dụng, từ ứng dụng trong doanh nghiệp, ứng dụng trên máy tính cá nhân cho đến các ứng dụng nhúng cho các thiết bị

Mọi người có thể tự làm thêm các thành phần gắn thêm theo yêu cầu riêng của mình

Tuy nhiên, kiến trúc này cũng có nhược điểm là người phát triển cần phải hiểu biết về nó để biết lúc nào thì gắn thêm (hay gỡ ra) và gắn thêm cái gì. Ngoài ra, các thành phần gắn thêm (các dự án) không ngừng phát triển nên phải biết lúc nào thì nâng cấp lên đời mới hơn (mặc dù Eclipse cũng có khả năng tự động tìm thành phần gắn thêm).

# Chương 3: Xây dựng ứng dụng trên nền tảng Eclipse



## Bài toán điều phối các lời gọi dịch vụ trong kiến trúc SOA

### Mục tiêu

Nền tảng Eclipse cho phép dễ kết hợp các thành phần để tạo ra các giao diện người sử dụng nâng cao. Mặt khác các web services dựa trên java cung cấp một nền tảng rất tốt để phát triển khung ứng dụng quy trình nghiệp vụ theo định hướng dịch vụ . Cả hai khía cạnh trên, giao diện người sử dụng dựa trên plug-in và quy trình nghiệp vụ theo định hướng dịch vụ (service-oriented business processes) là không thể thiếu cho sự thành công của một sản phẩm phần mềm doanh nghiệp mạnh mẽ và tích hợp đầy đủ. Chính vì vậy, trước hết, giải pháp này tập trung vào mở rộng nền tảng Eclipse cho phép phát triển và triển khai các web services như các plug-in khác trong Eclipse. Do đó, các dịch vụ là được tích hợp trong một kịch bản định hướng đường ống để giải quyết các kịch bản nghiệp vụ cụ thể - dẫn tới việc ta xây dựng một kiến trúc hướng dịch vụ theo đường ống - Service-oriented Pipeline Architecture (SOPA)

### Giải pháp

Giải pháp là xây dựng một dịch vụ web đăng ký (web services registry), gọi là bus dịch vụ (Services Bus), phát triển như một plug-in Eclipse. Services Bus cung cấp các điểm mở rộng cho các nhà phát triển để xuất bản các lớp Java tiêu chuẩn của họ như các web services.

Một phần nữa trong giải pháp là triển khai một plug-in đường ống (Pipeline plug-in), đóng vai trò trung tâm trong việc điều phối dịch vụ của hệ thống và trong việc tạo ra các dịch vụ nghiệp vụ mới. Pipeline được định nghĩa bằng một cấu trúc XML để quy định các bước trong Pipeline và các chuyển đổi liên quan. Nó bao gồm một số câu lệnh điều kiện kiểu XSLT và trao đổi dữ liệu bởi biểu thức XPATH. Pipeline được thống nhất truy cập qua Web Service plug-in, quản lý toàn bộ hệ thống dịch vụ bao gồm plug-in services, pipelines và các dịch vụ web bên ngoài (external web services). Mô tả chi tiết

## Mô tả chi tiết

### Kiến trúc hướng dịch vụ theo đường ống

Services và pipelines là dựa trên các thành phần của kiến trúc hướng dịch vụ theo đường ống – Service-oriented Pipeline Architecture (SOPA). Các dịch vụ có thể bao gồm từ giao diện người sử dụng dịch vụ tới các thành phần quy trình nghiệp vụ (Web Services). Mặt khác, pipelines mô tả các thành phần của các dịch vụ để thực hiện các nhiệm vụ cụ thể. Chính thức thì SOPA={S,P}, nơi mà các dịch vụ S có thể là các giao diện người sử dụng dịch vụ (GUI services) Sv , internal web-services Sw , và external web-services Sx. Tức là S ={ Sv , Sw , Sx}. Pipelines P sẽ điều phối các dịch vụ nghiệp vụ khác nhau (Sw và Sx) và áp dụng các phép chuyển đổi T để trả lại kết quả cho người sử dụng hoặc dịch vụ khác, tức là P ={Sw , Sx ,T}

### Services Bus

Phần cốt lõi của khung ứng dụng Eclipse là plug-in và cơ chế mở rộng của nó. Một plug-in có thể coi là điểm mở rộng để các plug-in khác có thể kết nối đến. Eclipse cũng cung cấp cơ chế để quản lý tập hợp các điểm mở rộng cơ bản và các đăng ký tương ứng của nó. Ví dụ như ViewsRegistry trong Views. Mặc dù có một số plug-in của hãng thứ ba được phát triển cho web services và triển khai trên các máy chủ đã được cấu hình, tuy nhiên Eclipse vẫn chưa hỗ trợ cơ chế plug-n-play cho web services như đối với Views. Cơ chế này có thể làm quá trình phát triển web services trở nên đơn giản hơn bằng cách chuyển phần khó khăn của việc triển khai dịch vụ của lập trình viên sang cho phần quản lý các thành phần dịch vụ. Do đó, các lập trình viên chỉ cần phải tập trung vào việc phát triển các tính năng của dịch vụ thay vì phải quan tâm đến chi tiết vận hành của nó.

Services Bus cung cấp các điểm mở rộng cho các nhà phát triển để xuất các lớp Java tiêu chuẩn của họ như các dịch vụ web. Cơ chế điểm mở rộng của Eclipse tạo điều kiện thuận lợi cho việc cấu hình các phần mở rộng với nhà cung cấp phần mở rộng. Khi khởi chạy ứng dụng, services bus sẽ tải tất cả các dịch vụ đã được kết nối với nhau và tự động triển khai chúng bằng cách sử dụng máy chủ nhúng Jetty và Apache AXIS. Các script triển khai được tạo ra từ các mô tả dịch vụ. Ngoài ra, Services Bus cũng sử dụng tiêu chuẩn WSDD và WSDL để cấu hình dịch vụ.

### Plug-n-Play Web Services

Điểm cơ bản trong phát triển Service Bus chính là đạt được mục tiêu của plug-n-play web services sử dụng các plug-in và cơ chế mở rộng của nền tảng Eclipse. Đầu tiên, một điểm mở rộng phải được cấu hình theo các đặc tả dịch vụ và các tiêu chuẩn triển khai như WSDL và WSDD.

|  |
| --- |
| <schema targetNamespace="org.nhan.services">  <element name="**service**">  <complexType>  <sequence>  <element ref="operation" minOccurs="1" MaxOccurs="unbounded"/>  </sequence>  <attribute name="name" type="string" use="required">  </attribute>  <attribute name="class" type="string" use="required">  <annotation>  <appInfo>  <meta.attribute kind="java"/>  </appInfo>  </annotation>  </attribute>  </complexType>  </element>  <element name="**operation**">  <complexType>  <sequence>  <element ref="parameter" minOccurs="1" maxOccurs="unbounded"/>  </sequence>  <attribute name="name" type="string" use="required">  </attribute>  <attribute name="returns" use="required">  </attribute>  </complexType>  </element>  <element name="**parameter**">  <complexType>  <attribute name="type" use="required">  </attribute>  <attribute name="name" type="string">  </complexType>  </element>  </schema> |

Phiên bản tóm tắt của lược đồ điểm mở rộng

Điều quan trọng nhất của Service Bus đối với điểm mở rộng này là các lập trình viên phải xuất ra các class Java tiêu chuẩn như các dịch vụ. Điều này sẽ làm quá trình lập trình và bảo dưỡng web services thuận tiện hơn đối với các plug-in Eclipse khác. Ngoài ra, Services Bus đọc các chi tiết cấu hình của các dịch vụ được kết nối trong quá trình khởi động. Nó sẽ tự động tạo ra các script dựa trên WSDD và sử dụng máy chủ Jetty nhúng và Apache AXIS để hoàn thành nhiệm vụ.

|  |
| --- |
| <extension point="org.nhan.services.services">  <**service** name="org.example.arithmatics" **class**="org.example.Arithmatics" **namespace**="org.nhan.sbus.sarithmatics">  <**operation** name="multiply" returnType="xsd:double">  <parameter name="first" type="xsd:double"/>  <parameter name="second" type="xsd:double"/>  </operation>  </service>  <extension> |

Mô tả dịch vụ như phần mở rộng

### Tính trong suốt của lời gọi Service

Service Bus có thể được xem như cánh cửa để đi vào hệ thống SOPA dựa trên Eclipse. Nó có chức năng định tuyến các yêu cầu dịch vụ đến các dịch vụ được kết nối hiện thời. Một chức năng khác quan trọng của Service Bus đó là nâng cao tính linh hoạt của hệ thống SOPA. Từ đó, nó cung cấp một lớp truy cập đồng nhất và trong suốt đến các dịch vụ bên trong và dịch vụ bên ngoài hệ thống.

Lời gọi dịch vụ (Service call): Các dịch vụ cắm trong Service Bus có thể được gọi bởi các class tương ứng hoặc sử dụng máy chủ Apache AXIS. Cả hai cách này đều sử dụng các cách đặt tên truyền thống cho các class và các hàm, và trả về kết quả giống nhau. Ngoài ra, đối với các dịch vụ bên trong chỉ cần gọi tên là đủ, tuy nhiên các dịch vụ bên ngoài còn yêu cầu cung cấp đầy đủ end-point URI.

|  |
| --- |
| Object[] params = ...  Call client = new Call("at.slife.store");  Object result = client.invoke("sparkle", params); |

Gọi một dịch vụ được cắm vào trong Services Bus

### Dịch vụ đường ống – Services Pipeline

Một pipeline trong thuật ngữ SOPA là một tập hợp có tên duy nhất các lời gọi service và chuyển đổi trung gian. Pipeline plug-in cho phép hệ thống SOPA hiểu rõ kịch bản dựa trên dịch vụ cơ bản và các pipelines. Ý tưởng dựa trên các thành phần đường ống (component pipeline), mỗi thành phần trong pipelines chỉ định một hoạt động cụ thể, các thành phần móc nối với nhau vào pipelines mà không cần yêu cầu lập trình.

Các pipeline và cấu trúc tương ứng của chứng được định nghĩa bằng cách sử dụng một cấu trúc XML để chỉ rõ các thành phần pipeline và các chuyển đổi liên quan.

|  |
| --- |
| 1. < pipeline name="square"> 2. < parameters> 3. < parameter name="num" type="xsd:double"/> 4. </parameters> 5. < call service="org.example.arithmatics" operation="multiply"/> 6. < parameter>{num}</parameter> 7. < parameter>{num}</parameter> 8. </call> 9. < transform method="xml" stylesheet="result.xsl"/> 10. </pipeline> |

Cấu trúc cơ bản của một pipeline đơn giản

Các dịch vụ có sẵn trong môi trường SOPA được định tuyến thông qua Services Bus plug-in, tức là tất cả dịch vụ sẽ được yêu cầu từ Services Bus, nơi chịu trách nhiệm cho việc tìm thấy và triệu gọi các dịch vụ tương ứng để làm nhiệm vụ. Tính năng này cung cấp sự trong suốt của dịch vụ trong toàn bộ môi trường SOPA. Các dịch vụ trong SOPA là không giới hạn để các dịch vụ tiếp xúc với nhau nhưng có thể tùy chọn bao gồm các pipeline, các Plug-in services và các dịch vụ web từ bên ngoài (External Web Services). Kết quả là hệ thống SOPA mang đến cho kịch bản tích hợp dịch vụ một chân trời mới. Các kịch bản nghiệp vụ phát triển theo khung lập trình Eclipse có thể kết hợp các thành phần từ nội bộ hoặc bên ngoài thông qua một dịch vụ định tuyến (Services Bus plug-in).

Plug-in Services

Services

Services

Pipelines



External

Web Services

Services Bus

Plug-in



Tính trong suốt của dịch vụ trong SOPA

### Tính năng kỹ thuật và các loại kịch bản của Pipeline

**Multiple Calls – Đa lời gọi**

**Nested Calls – Lời gọi lồng ghép**

**Conditional Calls – Lời gọi có điều kiện**

**XPath Extracted Parameters – Các tham số Xpath**

**Pipes within Pipes – Pipes trong pipes**

# 

# Phần kết luận và hướng phát triển

# TÀI LIỆU THAM KHẢO

**Tiếng Việt**

* 1. Nguyễn Trọng Dũng, *Phát triển Web Service với các công nghệ chuẩn của java* , Trường ĐHSP Hà Nội
  2. Hồ Bảo Thanh & Nguyễn Hoàng Long (2005), *Nghiên cứu kiến trúc hướng dịch vụ (Service-Oriented Architecture) và ứng dụng,* Luận văn cử nhân tin học khoa CNTT – ĐH KHTN Tp.HCM

**Tiếng Anh**

* 1. Abdaldhem Albreshne, Patrik Fuhrer, Jacque Pasquier Spasquier (9-2009), *Web Services Technologies,* State of the Art
  2. Dr Alex Blewitt (June 2013), *Eclipse 4 Plug-in Development by Example*
  3. [Bertrand Portier](http://www.ibm.com/developerworks/webservices/library/ws-soa-term1/index.html#authorN1001E) (May 2007), *SOA terminology overview*, IT Architect IBM
  4. David s. Linthicum (2004), *12 Steps to implementing a Service-Oriented Architecture*
  5. David Gallardo, *Developing Eclipse plug-ins,*  <http://www.ibm.com/developerworks/opensource/library/os-ecplug/>
  6. Deepal Jayasinghe (May 2008), *Quickstart Apache Axis2*
  7. Eric Clayberg – Dan Rubel (2008), *Eclipse Plug-ins – Third Edition*, Addison Wesley
  8. Hartwig Gunzer (March 2002), *Introduction to Web Services*, Sales Engineer, Borland
  9. Kiet T. Tran (2013), *Introduction to Web Services with Java*
  10. Mark D. Hansen (May 2007), *SOA Using Java Web Services*
  11. Martin Kalin (2013), *Java Web Services : Up and Running, Second Edition*
  12. Yuli Vasiliev (September 2007), *SOA and WS-BPEL*
  13. Chris Aniszczyk, David Gallardo, *Get Started with the Eclipse platform* - <http://www.ibm.com/developerworks/opensource/library/os-eclipse-platform/#N101FF>
  14. Eclipse – <http://www.eclipse.org/>
  15. *Web Services Business Process Execution Language Version 2.0*, OASIS (2007), <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html>