BỘ CÔNG THƯƠNG

**TRƯỜNG ĐẠI HỌC KINH TẾ KỸ THUẬT CÔNG NGHIỆP**

**KHOA CÔNG NGHỆ THÔNG TIN**



MÔN:

**THỰC TẬP LẬP TRÌNH MẠNG**

**Họ tên sinh viên : Bùi Như Trường**

**Nguyễn Đình Thắng**

**Nguyễn Văn Nhân**

**Giảng viên: Trần Hồng Việt**

**Năm 2021**

**Giáo viên hướng dẫn: Trần Hồng Việt**

**Năm 2021**

Mục Lục

[**THỰC TẬP LẬP TRÌNH MẠNG** 1](https://d.docs.live.net/99010a34579522e4/Máy%20tính/BTL%20java.docx#_Toc89240566)

[Chương 1. 1](#_Toc89240567)

[Chapter 16.Protocol Handlers 1](#_Toc89240568)

[I.1 Thế nào là Protocol Handler 1](#_Toc89240569)

[1.2 Lớp URLStreamHandler 7](#_Toc89240570)

[1.2.1 The Constructor 8](#_Toc89240571)

[1.2.2 Phương pháp phân tích cú pháp URL 8](#_Toc89240572)

[1.2.2.1 Protected void parseURL(URL u, String spec, int start, int limit) 8](#_Toc89240573)

[1.2.2.2 Protected String toExternalForm(URL u) 11](#_Toc89240574)

[1.2.2.3 Protected void setURL(URL u, String protocol, String host, int port, String file, String ref ) 12](#_Toc89240575)

[1.2.2.4 Protected int getDefaultPort( ) // Java 1.3 14](#_Toc89240576)

[1.2.2.5 Protected InetAddress getHostAddress(URL u) // Java 1.3 14](#_Toc89240577)

[1.2.2.6 Protected boolean hostsEqual(URL u1, URL u2) // Java 1.3 15](#_Toc89240578)

[16.2.2.7 Protected boolean sameFile(URL u1, URL u2) // Java 1.3 15](#_Toc89240579)

[16.2.2.8 Protected boolean equals(URL u1, URL u2) // Java 1.3 16](#_Toc89240580)

[16.2.2.9 Protected int hashCode(URL u) // Java 1.3 16](#_Toc89240581)

[16.2.3 Phương pháp kết nối 17](#_Toc89240582)

[16.2.3.1 Protected abstract URLConnection openConnection(URL u) throws IOException 17](#_Toc89240583)

[16.3 Viết một trình xử lý giao thức 19](#_Toc89240584)

[1.4 Thêm các ví dụ và kỹ thuật về xử lý giao thức 25](#_Toc89240585)

[1.4.1 Trình xử lý giao thức daytime 27](#_Toc89240586)

[Example 16.4. The DaytimeURLConnection Class 28](#_Toc89240587)

[Ví dụ 16.5. Lớp DaytimeURLStreamHandler 30](#_Toc89240588)

[16.4.2 Trình xử lý giao thức hàng loạt 31](#_Toc89240589)

[Ví dụ 16.6. FiniteInputStream 33](#_Toc89240590)

[Ví dụ 16.8. Lớp xử lý chargen 37](#_Toc89240591)

[16.5 Giao diện URLStreamHandlerFactory 38](#_Toc89240592)

[Ví dụ 16.9. URLStreamHandlerFactory cho finger, daytime và chargen 39](#_Toc89240593)

[Ví dụ 16.10. Chương trình SourceViewer đặt URLStreamHandlerFactory 40](#_Toc89240594)

[Chương 2. 42](#_Toc89240595)

[Chapter17. Trình xử lí nội dung 42](#_Toc89240596)

[17. 1 Trình xử lý nội dung là gi? 45](#_Toc89240597)

[17.2 Lớp ContentHandler 48](#_Toc89240598)

[17.2.1 Trình xử lý nội dung cho các giá trị được phân tách bằng tab 49](#_Toc89240599)

[17.2 2 Sử dụng trình xử lý nội dung 52](#_Toc89240600)

[17.2.3 Chọn loại trả lại 59](#_Toc89240601)

[17.3 Giao diện ContentHandlerFactory 67](#_Toc89240602)

[17.3.1 Phương thức createContentHandler () 67](#_Toc89240603)

[17.3.2 Cài đặt các nhà máy xử lý nội dung 69](#_Toc89240604)

[17.4 Trình xử lý nội dung cho định dạng hình ảnh: image / x-fit 71](#_Toc89240605)

[Chương 3. 87](#_Toc89240606)

[Dịch vụ lập trình công cụ crawl dữ liệu website tự động. 87](#_Toc89240607)

[1.Crawl dữ liệu là gì. 87](#_Toc89240608)

[2.Crawl phù hợp với những doanh nghiệp nào. 88](#_Toc89240609)

[3.Lợi ích của việc crawl data là gì? 88](#_Toc89240610)

[4.Công nghệ sử dụng là gì? 89](#_Toc89240611)

[5.Có bị Google phạt không? 89](#_Toc89240612)

[6.Kết luận: 90](#_Toc89240613)

# Chương 1.

## Chapter 16.Protocol Handlers

Khi thiết kế một kiến ​​trúc cho phép họ xây dựng một trình duyệt, các kỹ sư tại Sun đã chia vấn đề thành hai phần: Protocol Handlers và Content Handlers. Xử lý một giao thức có nghĩa là quan tâm đến sự tương tác giữa máy khách và máy chủ: tạo yêu cầu ở định dạng chính xác, diễn giải các tiêu đề quay trở lại với dữ liệu, xác nhận rằng dữ liệu đã được nhận, v.v. Xử lý nội dung có nghĩa là chuyển đổi dữ liệu thô thành định dạng Java hiểu, ví dụ, một InputStream hoặc một AudioClip. Hai vấn đề này, các giao thức xử lý và nội dung xử lý khác nhau. Phần mềm hiển thị GIF hình ảnh không quan tâm liệu hình ảnh có được truy xuất qua FTP, HTTP, gopher hay một số giao thức mới. Tương tự như vậy, trình xử lý giao thức, quản lý kết nối và tương tác với máy chủ, không quan tâm đến việc nó đang nhận tệp HTML hay MPEG tập phim; tối đa, nó sẽ trích xuất một loại nội dung từ các tiêu đề để chuyển đến trình xử lý nội dung.

Java chia nhiệm vụ xử lý các giao thức thành một số phần. Kết quả là, không có lớp đơn lẻ nào được gọi là ProtocolHandler. Thay vào đó, các phần của trình xử lý giao thức cơ chế được thực hiện bởi bốn lớp khác nhau trong gói java.net: URL,

URLStreamHandler, URLConnection và URLStreamHandlerFactory. URL là

chỉ lớp cụ thể trong nhóm này; URLStreamHandler và URLConnection đều là

các lớp trừu tượng và URLStreamHandlerFactory là một giao diện. Do đó, nếu bạn

để triển khai một trình xử lý giao thức mới, bạn phải viết các lớp con cụ thể cho

URLStreamHandler và URLConnection. Để sử dụng các lớp này, bạn cũng có thể

phải viết một lớp triển khai giao diện URLStreamHandlerFactory.

I.1 Thế nào là Protocol Handler

Cách URL, URLStreamHandler, URLConnection và Các lớp URLStreamHandlerFactory hoạt động cùng nhau có thể gây nhầm lẫn. Mọi điều

bắt đầu bằng một URL, đại diện cho một con trỏ đến một tài nguyên Internet cụ thể. Mỗi URL chỉ định giao thức được sử dụng để truy cập tài nguyên; các giá trị điển hình cho giao thức bao gồm mailto, http và ftp. Khi bạn tạo một đối tượng URL từ URL của biểu diễn chuỗi, hàm tạo tách trường giao thức và chuyển nó đến

URLStreamHandlerFactory. Công việc của factory là thực hiện giao thức, xác định vị trí lớp con của URLStreamHandler cho giao thức và tạo một phiên bản mới của giao thức đó trình xử lý luồng, được lưu trữ dưới dạng một trường trong đối tượng URL. Mỗi ứng dụng có nhiều nhất một URLStreamHandlerFactory; khi factory đã được lắp đặt xong, cố gắng cài đặt một cái khác sẽ xuất hiện một Lỗi.

Bây giờ đối tượng URL có một trình xử lý luồng, nó yêu cầu trình xử lý luồng hoàn thành phân tích cú pháp chuỗi URL và tạo một lớp con của URLConnection biết cách nói chuyện với các máy chủ sử dụng giao thức này. Các lớp con URLStreamHandler và các lớp con của URLConnection luôn đi theo cặp; trình xử lý luồng cho một giao thức luôn biết cách tìm một URLConnection thích hợp cho giao thức của nó. Nó có giá trị lưu ý rằng trình xử lý luồng thực hiện hầu hết công việc phân tích cú pháp URL. Định dạng của URL, nó là tiêu chuẩn, phụ thuộc vào giao thức; do đó, nó phải được phân tích cú pháp bởi URLStreamHandler, biết về một giao thức cụ thể, chứ không phải bởi đối tượng URL, là đối tượng chung chung và do đó không được biết về các giao thức. Điều này cũng có nghĩa là nếu bạn đang viết một trình xử lý luồng mới, bạn có thể xác định một định dạng URL mới phù hợp với nhiệm vụ của bạn.

Lớp URLConnection đại diện cho một kết nối đang hoạt động với tài nguyên Internet. Nó chịu trách nhiệm cho tương tác với máy chủ. Một URLConnection biết cách tạo các yêu cầu và diễn giải các tiêu đề mà máy chủ trả về. Đầu ra từ URLConnection là dữ liệu thô được yêu cầu với tất cả các dấu vết của giao thức (tiêu đề, v.v.) được loại bỏ, sẵn sàng cho xử lý bởi trình xử lý nội dung.

Trong hầu hết các ứng dụng, bạn không cần phải lo lắng về các đối tượng URLConnection và stream handlers; chúng bị ẩn bởi lớp URL, cung cấp một giao diện đơn giản với các phương pháp bạn cần. Khi bạn gọi getInputStream (), phương thức getOutputStream () và getContent () của lớp URL, bạn thực sự gọi các phương thức có tên tương tự trong lớp URLConnection. Chúng tôi đã thấy rằng tương tác trực tiếp với URLConnection có thể thuận tiện khi bạn cần một chút

kiểm soát nhiều hơn giao tiếp với máy chủ, phổ biến nhất là khi tải xuống tệp nhị phân.

Tuy nhiên, các lớp URLConnection và URLStreamHandler thậm chí còn nhiều quan trọng hơn khi bạn cần thêm các giao thức mới. Bằng cách viết các lớp con của các lớp này, bạn có thể thêm hỗ trợ cho các giao thức tiêu chuẩn như ngón tay, whois hoặc NTP mà Java không hỗ trợ ra khỏi hộp. Hơn nữa, bạn không bị giới hạn ở các giao thức đã thiết lập với các dịch vụ nổi tiếng. Bạn có thể tạo các giao thức mới để thực hiện cơ sở dữ liệu truy vấn, tìm kiếm trên nhiều công cụ tìm kiếm trên Internet, xem ảnh từ hệ nhị phân nhóm tin tức và hơn thế nữa. Bạn có thể thêm các loại URL mới nếu cần để đại diện cho các loại tài nguyên mới. Hơn nữa, các ứng dụng Java có thể được xây dựng để chúng có thể tải các trình xử lý giao thức mới trong thời gian chạy. Không giống như các trình duyệt hiện tại như Mozilla và

Internet Explorer, chứa kiến ​​thức rõ ràng về tất cả các giao thức và nội dung các loại mà chúng có thể xử lý, một trình duyệt Java có thể là một bộ xương tương đối nhẹ tải các trình xử lý mới nếu cần. Hỗ trợ một giao thức mới chỉ có nghĩa là thêm một số các lớp mới ở các vị trí được xác định trước, không phải viết một bản phát hành hoàn toàn mới của trình duyệt.

Điều gì liên quan đến việc thêm hỗ trợ cho một giao thức mới? Như tôi đã nói trước đó, bạn cần phải viết hai lớp mới: một lớp con của URLConnection và một lớp con của URLStreamHandler. Bạn cũng có thể cần viết một lớp triển khai

Giao diện URLStreamHandlerFactory. Lớp con URLConnection của bạn xử lý

tương tác với máy chủ, chuyển đổi bất cứ thứ gì máy chủ gửi thành Dòng đầu vào,

và chuyển đổi bất cứ thứ gì mà khách hàng gửi thành một Dòng ra. Lớp con này phải thực hiện phương thức trừu tượng connect (); nó cũng có thể đè lên các phương thức getInputStream (), getOutputStream () và getContent Type ().

Lớp con URLStreamHandler phân tích cú pháp biểu diễn chuỗi của URL thành các phần và tạo một đối tượng URLConnection mới hiểu giao thức của URL đó. Lớp con này phải triển khai phương thức trừu tượng openConnection (),

trả về URLConnection mới cho người gọi của nó. Nếu biểu diễn chuỗi của

URL trông không giống như một URL http chuẩn, thì bạn cũng nên ghi đè lên

các phương thức parseURL () và toExternalForm ().

Cuối cùng, bạn có thể cần tạo một lớp triển khai giao diện URLStreamHandlerFactory. URLStreamHandlerFactory giúp ứng dụng tìm trình xử lý giao thức phù hợp cho từng loại URL. Các giao diện URLStreamHandlerFactory có một phương pháp duy nhất, hãy tạo URLStreamHandler (), trả về một đối tượng URLStreamHandler. Phương pháp này phải tìm lớp con thích hợp của URLStreamHandler chỉ được cung cấp giao thức (ví dụ: ftp); nghĩa là, nó phải hiểu bất kỳ quy ước đặt tên gói và lớp nào mà bạn sử dụng cho trình xử lý luồng của bạn. Vì URLStreamHandlerFactory là một giao diện, bạn có thể đặt phương thức createURLStreamHandler () của bạn trong bất kỳ lớp nào thuận tiện, có lẽ là lớp chính của ứng dụng của bạn.

Khi lần đầu tiên gặp một giao thức, Java sẽ tìm kiếm các lớp URLStreamHandler trong đặt hàng:

1. Đầu tiên, Java kiểm tra xem URLStreamHandlerFactory đã được cài đặt hay chưa. Nếu như nó đã được cài đặt, factory được yêu cầu một URLStreamHandler cho giao thức.
2. Nếu URLStreamHandlerFactory không được cài đặt hoặc nếu Java không thể tìm thấy URLStreamHandler cho giao thức, sau đó Java tìm trong các gói có tên trong thuộc tính hệ thống java.protocol.handler.pkgs cho gói con chia sẻ tên giao thức và một lớp được gọi là Handler. Giá trị của cái này thuộc tính là một danh sách các tên gói được phân tách bằng một thanh dọc (|). Do đó, để chỉ ra rằng Java nên tìm kiếm các trình xử lý giao thức trong com.macfaq.net.www và gói org.cafeaulait.protocols, bạn sẽ thêm dòng này vào tệp thuộc tính:

java.protocol.handler.pkgs=com.macfaq.net.www|org.cafeaulait.protocols

Sau đó, để tìm một trình xử lý giao thức FTP (ví dụ), trước tiên Java sẽ tìm

kiếm lớp com.macfaq.net.www.ftp.Handler. Nếu điều đó không được tìm thấy,

Java tiếp theo sẽ cố gắng khởi tạo org.cafeaulait.protocols.ftp.Handler.

1. Cuối cùng, nếu vẫn thất bại, Java sẽ tìm kiếm URLStreamHandler có tên

sun.net.www.protocol.name.Handler, trong đó tên được thay thế bằng tên

của giao thức; ví dụ: sun.net.www.protocol.ftp.Handler.

Thông thường, người dùng cuối muốn cài đặt vĩnh viễn một giao thức bổ sung

trình xử lý trong một chương trình chẳng hạn như HotJava sẽ đặt các lớp cần thiết trong chương trình của đường dẫn lớp và thêm tiền tố gói vào thuộc tính java.protocol.handler.pkgs. Tuy nhiên, một lập trình viên chỉ muốn thêm một trình xử lý protocol tùy chỉnh cho cô ấy chương trình tại thời điểm biên dịch sẽ viết và cài đặt một URLStreamHandlerFactory biết cách tìm các trình xử lý giao thức tùy chỉnh của cô ấy. Nhà máy có thể yêu cầu một ứng dụng tìm kiếm các lớp URLStreamHandler ở bất kỳ nơi nào thuận tiện: trên trang web, trong cùng thư mục với ứng dụng hoặc ở đâu đó trong đường dẫn lớp của người dùng.

Khi mỗi lớp này đã được viết và biên dịch, bạn đã sẵn sàng để viết ứng dụng sử dụng trình xử lý giao thức mới của bạn. Giả sử rằng bạn đang sử dụng

URLStreamHandlerFactory, chuyển đối tượng gốc vào URL.setURL

Phương thức StreamHandlerFactory () như thế này:

URL.setURLStreamHandlerFactory(new MyURLStreamHandlerFactory( ));

Phương thức này chỉ có thể được gọi một lần trong thời gian tồn tại của một ứng dụng. Nếu nó được gọi lần thứ hai, nó sẽ xuất hiện một Lỗi. Các applet không đáng tin cậy nói chung sẽ không được phép để cài đặt các factory hoặc thay đổi thuộc tính java.protocol.handler.pkgs. Do đó, các trình xử lý giao thức chủ yếu được sử dụng cho các ứng dụng độc lập như như HotJava; Netscape và Internet Explorer sử dụng mã C gốc của riêng họ thay vì Java để xử lý các giao thức, vì vậy chúng bị giới hạn trong một tập hợp các giao thức cố định.

Tóm lại đây là chuỗi sự kiện

1. Chương trình xây dựng một đối tượng URL.
2. Hàm tạo sử dụng các đối số mà nó được truyền để xác định phần giao thức của URL, ví dụ: http.
3. Hàm tạo URL () cố gắng tìm một URLStreamHandler cho giao thức như thế này:
4. Nếu giao thức đã được sử dụng trước đó, thì URLStreamHandler

đối tượng được truy xuất từ ​​bộ nhớ cache.

1. Ngược lại, nếu URLStreamHandlerFactory đã được đặt, thì

chuỗi giao thức được chuyển đến phương pháp factory's

createURLStreamHandler( ).

1. Nếu giao thức chưa từng được nhìn thấy trước đây và không có URlStreamHandlerFactory, sau đó hàm tạo cố gắng khởi tạo một

đối tượng URLStreamHandler có tên là protocol.Handler ở một trong

các gói được liệt kê trong thuộc tính java.protocol.handler.pkgs.

1. Không thực hiện được, hàm tạo cố gắng khởi tạo một

đối tượng URLStreamHandler có tên là giao protocol.Handler trong

gói sun.net.www.protocol.

1. Nếu bất kỳ nỗ lực nào trong số này thành công trong việc truy xuất đối tượng URLStreamHandler, phương thức khởi tạo URL đặt trường xử lý của đối tượng URL. Nếu không có nỗ lực nào thành công, hàm tạo đưa ra một lỗi sai định dạng URLException.
2. Chương trình gọi phương thức openConnection () của đối tượng URL.
3. Đối tượng URL yêu cầu URLStreamHandler trả về một đối tượng URLConnection thích hợp cho URL này. Nếu có bất kỳ sự cố nào, một IOException sẽ được đưa ra. Nếu không, một đối tượng URLConnection được trả về.
4. Chương trình sử dụng các phương thức của lớp URLConnection để tương tác với tài nguyên từ xa.

Thay vì gọi openConnection () ở bước 4, chương trình có thể gọi getContent ()

hoặc getInputStream (). Trong trường hợp này, URLStreamHandler vẫn khởi tạo một đối tượng URLConnection của lớp thích hợp. Tuy nhiên, thay vì trả về bản thân đối tượng URLConnection, URLStreamHandler trả về kết quả là phương thức getContent () hoặc getInputStream () của URLConnection.

1.2 Lớp URLStreamHandler

Lớp URLStreamHandler trừu tượng là lớp cha dành cho các lớp xử lý giao thức — ví dụ: HTTP. Bạn hiếm khi gọi các phương thức của trực tiếp lớp URLStreamHandler; chúng được gọi bằng các phương thức khác trong URL và

các lớp URLConnection. Bằng cách ghi đè các phương thức URLStreamHandler lớp con của riêng bạn, bạn dạy lớp URL cách xử lý các giao thức mới. Do đó, chúng tôi sẽ tập trung ghi đè các phương thức của URLStreamHandler thay vì gọi các phương thức.

### 1.2.1 The Constructor

Bạn không trực tiếp tạo các đối tượng URLStreamHandler. Thay vào đó, khi một URL được xây dựng bằng một giao thức chưa từng thấy trước đây, Java yêu cầu ứng dụng URLStreamHandlerFactory để tạo lớp con URLStreamHandler thích hợp

cho giao thức. Nếu không thành công, Java sẽ đoán ở tên gói đủ điều kiện của

Lớp URLStreamHandler và sử dụng Class.forName () để cố gắng xây dựng một

sự vật. Điều này có nghĩa là các lớp con cứng phải có một hàm tạo noargs. Duy nhất hàm tạo cho URLStreamHandler không nhận bất kỳ đối số nào:

public URLStreamHandler( )

Bởi vì URLStreamHandler là một lớp trừu tượng, hàm tạo này không bao giờ được gọi trực tiếp; nó chỉ được gọi từ các hàm tạo của các lớp con.

### 1.2.2 Phương pháp phân tích cú pháp URL

Trách nhiệm đầu tiên của URLStreamHandler là phân tách một biểu diễn chuỗi của một URL vào các phần thành phần của nó và sử dụng các phần đó để đặt các trường khác nhau của URL sự vật. Phương thức parseURL () chia URL thành các phần, có thể sử dụng setURL () để gán giá trị cho các trường của URL. Rất khó hình dung một tình huống trong đó bạn sẽ gọi parseURL () trực tiếp; thay vào đó, bạn ghi đè nó để thay đổi hành vi của lớp URL.

### 1.2.2.1 Protected void parseURL(URL u, String spec, int start, int limit)

Phương thức này phân tích cú pháp một chuỗi đặc tả thành một URL u. Tất cả các ký tự trong chuỗi đặc tả trước start nên đã được phân tích cú pháp thành URL u. Các ký tự sau giới hạn là mặc kệ. Nói chung, giao thức sẽ đã được phân tích cú pháp và lưu trữ trong u trước đó phương thức này được gọi và bắt đầu sẽ được điều chỉnh để nó bắt đầu bằng ký tự sau dấu hai chấm phân định giao thức.

Nhiệm vụ của parseURL () là đặt giao thức, máy chủ, cổng, tệp và tham chiếu lĩnh vực của URL u. Nó có thể giả định rằng bất kỳ phần nào của Chuỗi trước khi bắt đầu và sau đó giới hạn đã được phân tích cú pháp hoặc có thể bị bỏ qua.

Phương thức parseURL () mà Java cung cấp giả định rằng URL trông giống hơn hoặc ít hơn như một URL http:

protocol://www.host.com:port/directory/another\_directory/file#ref

Điều này hoạt động cho các URL ftp và gopher. Nó không hoạt động đối với các URL mailto hoặc tin tức và có thể không phù hợp với bất kỳ loại URL mới nào mà bạn xác định. Nếu trình xử lý giao thức của bạn sử dụng URL phù hợp với biểu mẫu này, bạn không phải ghi đè parseURL (); NS phương thức kế thừa từ URLStreamHandler sẽ hoạt động tốt. Nếu URL của bạn là hoàn toàn khác, bạn phải cung cấp một phương thức parseURL () để phân tích cú pháp URL hoàn toàn. Tuy nhiên, thường có một nền tảng trung gian có thể làm cho nhiệm vụ của bạn dễ dàng hơn. Nếu như URL của bạn trông giống như một URL chuẩn, bạn có thể triển khai một parseURL () phương thức xử lý phần không chuẩn của URL của bạn và sau đó gọi super.parseURL () để thực hiện phần còn lại của công việc, đặt giá trị bù đắp và giới hạn đối số để chỉ ra phần URL mà bạn không phân tích cú pháp.

*Ví dụ: một URL mailto trông giống như mailto: elharo@metalab.unc.edu. Đầu tiên bạn cần tìm ra cách ánh xạ điều này vào giao thức của lớp URL, máy chủ, cổng, tệp, và các trường giới thiệu. Giao thức rõ ràng là mailto. Mọi thứ sau ký tự @ đều có thể là máy chủ lưu trữ. Câu hỏi khó là phải làm gì với tên người dùng. Vì một URL mailto thực sự không có một phần tệp, chúng tôi sẽ sử dụng trường tệp của lớp URL để giữ tên người dùng. Các ref có thể được đặt thành chuỗi rỗng hoặc null. Phương thức parseURL () theo sau triển khai chương trình này:*

public void parseURL(URL u, String spec, int start, int limit) {

String protocol = u.getProtocol( );

String host = "";

int port = u.getPort( );

String file = ""; // really username

String ref = null;

if( start < limit) {

String address = spec.substring(start, limit);

int atSign = address.indexOf('@');

if (atSign >= 0) {

host = address.substring(atSign+1);

file = address.substring(0, atSign);

}

}

this.setURL(u, protocol, host, port, file, ref);

}

Thay vì mượn một trường không sử dụng từ đối tượng URL, đó có thể là một ý tưởng tốt hơn để lưu trữ các phần cụ thể theo giao thức của URL, chẳng hạn như tên người dùng, trong các trường của lớp con URLStreamHandler. Nhược điểm của phương pháp này là các trường như chỉ có thể được nhìn thấy bằng mã của riêng bạn; trong ví dụ này, bạn không thể sử dụng getFile () trong lớp URL để truy xuất tên người dùng. Đây là phiên bản của parseURL () lưu tên người dùng trong một trường của lớp con Trình xử lý. Khi kết nối đã mở, tên người dùng có thể được sao chép vào đối tượng Mailto URLConnection mà kết quả. Lớp đó sẽ cung cấp một số loại phương thức getUserName ():

String username = "";

public void parseURL(URL u, String spec, int start, int limit) {

String protocol = u.getProtocol( );

String host = "";

int port = u.getPort( );

String file = "";

String ref = null;

if( start < limit) {

String address = spec.substring(start, limit);

int atSign = address.indexOf('@');

if (atSign >= 0) {

host = address.substring(atSign+1);

this.username = address.substring(0, atSign);

}

}

This.setURL(u, protocol, host, port, file, ref);

}

### 1.2.2.2 Protected String toExternalForm(URL u)

Phương thức này đặt các phần của URL u — nghĩa là, giao thức, máy chủ, cổng, tệp, và các trường ref — trở lại với nhau trong một chuỗi. Nếu bạn ghi đè parseURL (), bạn cũng nên ghi đè thành toExternalForm (). Đây là một phương thức toExternalForm () cho một URL mailto; nó giả định rằng tên người dùng đã được lưu trữ trong trường máy chủ của URL:

protected String toExternalForm(URL u){

return "mailto:" + u.getFile () + "@" + u.getHost ();

}

Vì toExternalForm () được bảo vệ, có thể bạn sẽ không gọi phương thức này trực tiếp. Tuy nhiên, nó được gọi bởi phương thức public toExternalForm () và toString () của lớp URL, vì vậy bất kỳ thay đổi nào bạn thực hiện ở đây đều được phản ánh khi bạn chuyển đổi URL đối tượng thành chuỗi.

### 1.2.2.3 Protected void setURL(URL u, String protocol, String host, int port, String file, String ref )

Phương thức này đặt các trường máy chủ, cổng, tệp và tham chiếu của URL u thành các giá trị đã cho. Nó đặt trường giao thức thành đối số giao thức chỉ trong Java 1.1.x trở về trước. (Trong Java 1.2 trở lên, trường này được đặt trước khi phương thức này được gọi và không thể thay đổi. Đối số giao thức bị bỏ qua.) Phương thức này được parseURL () sử dụng để đặt các các giá trị mà nó đã tìm thấy bằng cách phân tích cú pháp URL. Bạn cần gọi phương thức này tại cuối phương thức parseURL () khi bạn phân lớp URLStreamHandler.

Phương thức này không được chấp nhận trong Java 1.3 (mặc dù không có trong Java 1.2). Lý do là Java 1.3 cũng cho phép bạn thêm chuỗi truy vấn, thông tin người dùng và thẩm quyền vào URL. Vì vậy, Java 1.3 thích phương thức setURL () thay thế này, hỗ trợ các tính năng sau:

protected void setURL(URL u, String protocol, String host, int port,

// Java 1.3

String authority, String userInfo, String path, String query, String ref)

Trong Java 1.2 trở về trước, thông tin người dùng và quyền hạn có thể được bao gồm như một phần của máy chủ lưu trữ tranh luận. Chuỗi truy vấn có thể được đưa vào cuối tệp. Ngay cả trong Java 1.3, phương pháp này hơi phức tạp, vì máy chủ, cổng và thông tin người dùng cùng nhau tạo nên chính quyền. Trong trường hợp có xung đột giữa chúng, tất cả chúng đều được lưu trữ riêng biệt, nhưng

máy chủ, cổng và thông tin người dùng được sử dụng tùy theo cơ quan có thẩm quyền khi quyết định trang web để kết nối.

Điều này thực sự khá liên quan đến ví dụ mailto, vì các URL mailto thường có

chuỗi truy vấn cho biết chủ đề hoặc tiêu đề khác; Ví dụ, mailto: elharo@metalab.unc.edu? subject = JavaReading. Đây là chuỗi truy vấn chủ đề = JavaReading. Nếu chúng ta viết lại phương thức parseURL () ở trên thành

hỗ trợ các URL mailto ở định dạng này, kết quả sẽ như sau:

public void parseURL(URL u, String spec, int start, int limit) {

String protocol = u.getProtocol( );

String host = ""; int port = u.getPort( );

String file = "";

String userInfo = null;

String query = null;

String ref = null;

if (start < limit) {

String address = spec.substring(start, limit);

int atSign = address.indexOf('@');

int questionMark = address.indexOf('?');

int hostEnd = questionMark >= 0 ? questionMark : address.length( );

if (atSign >= 0) {

host = address.substring(atSign+1, hostEnd);

userInfo = address.substring(0, atSign);

}

if (questionMark >= 0 && questionMark > atSign) {

query = address.substring(questionMark + 1);

}

}

String authority = "";

if (userInfo != null)

authority += userInfo + '@'; authority += host;

if (port >= 0)

authority += ":" + port;

this.setURL(u, protocol, host, port, authority, userInfo, file, query, ref);

}

Lưu ý rằng điều này chỉ hoạt động trong Java 1.3 trở lên. Nó sẽ không hoạt động trong Java 1.1 hoặc 1.2.

### 1.2.2.4 Protected int getDefaultPort( ) // Java 1.3

Java 1.3 thêm một phương thức getDefaultPort () vào lớp URLStreamHandler có trách nhiệm là trả lại cổng mặc định cho giao thức; ví dụ: 80 cho HTTP. Các

việc triển khai mặc định của phương thức này chỉ trả về -1, nhưng mỗi lớp con phải ghi đè điều đó bằng cổng mặc định thích hợp cho giao thức mà nó xử lý. Ví dụ, đây là phương thức getDefaultPort () cho giao thức Finger thường hoạt động trên cổng 79:

public int getDefaultPort( ) {

return 79;

}

Ngoài việc cung cấp cổng phù hợp cho Finger, phương pháp ghi đè này cũng làm cho getDefaultPort () công khai. Mặc dù chỉ có một triển khai mặc định của điều này trong Java 1.3, không có lý do gì bạn không thể cung cấp nó trong các lớp con của riêng bạn trong bất kỳ phiên bản nào của Java. Bạn chỉ đơn giản là sẽ không thể gọi nó một cách đa hình từ một tham chiếu được nhập dưới dạng lớp cha.

### 1.2.2.5 Protected InetAddress getHostAddress(URL u) // Java 1.3

Java 1.3 cũng thêm một phương thức getHostAddress () vào lớp URLStreamHandler có trách nhiệm trả về một đối tượng InetAddress trỏ đến máy chủ trong URL. Điều này yêu cầu tra cứu DNS và phương pháp này không chặn trong khi tra cứu làm ra. Tuy nhiên, nó không ném ra bất kỳ trường hợp ngoại lệ nào. Nếu không xác định được máy chủ, bởi vì URL không chứa phần máy chủ lưu trữ hoặc có lỗi DNS hoặc SecurityException, sau đó nó chỉ trả về null. Việc triển khai mặc định của điều này phương pháp này là đủ cho bất kỳ trường hợp hợp lý nào. Không cần thiết phải ghi đè nó.

### 1.2.2.6 Protected boolean hostsEqual(URL u1, URL u2) // Java 1.3

Java 1.3's hostsEqual () xác định xem hai URL có tham chiếu đến cùng một máy chủ hay không. Phương pháp này cố gắng sử dụng DNS để thực sự tra cứu các máy chủ. Nếu điều đó thành công cho cả hai máy chủ, sau đó nó có thể cho biết điều đó, ví dụ: http://metalab.unc.edu/Dave/Dr Fun / new.jpg và ftp://sunsite.unc.edu/pub/linux/distributions/redhat/ hiện tại / là cùng một máy chủ. Tuy nhiên, nếu tra cứu DNS không thành công vì bất kỳ lý do gì, thì hostsEqual ()

rơi trở lại so sánh chuỗi đơn giản không phân biệt chữ hoa chữ thường, trong trường hợp đó, nó sẽ nghĩ đó là hai vật chủ khác nhau.

Việc triển khai mặc định của phương pháp này là đủ cho hầu hết các trường hợp. Bạn có thể sẽ không cần phải ghi đè nó. Trường hợp duy nhất tôi có thể tưởng tượng nơi bạn có thể muốn là nếu bạn đang cố gắng làm cho các trang nhân bản trên các máy chủ khác nhau trông giống nhau.

### 1.2.2.7 Protected boolean sameFile(URL u1, URL u2) // Java 1.3

Java 1.3's sameFile () xác định xem hai URL có trỏ đến cùng một tệp hay không. Nó có thể làm điều này bằng cách so sánh giao thức, máy chủ, cổng và đường dẫn. Các tệp được coi là giống nhau chỉ khi mỗi phần trong số bốn phần đó giống nhau. Tuy nhiên, nó không xem xét chuỗi truy vấn hoặc tham chiếu Hơn nữa, các máy chủ được so sánh bởi hostsEqual () để có thể nhận dạng metalab.unc.edu và sunite.unc.edu là giống nhau nếu DNS có thể giải quyết chúng. Điều này tương tự với phương thức sameFile () của lớp URL. Thật vậy, trong Java 1.3, phương thức sameFile () đó chỉ gọi phương thức sameFile () này.

Việc triển khai mặc định của phương pháp này là đủ cho hầu hết các trường hợp. Bạn có thể sẽ không cần phải ghi đè nó. Có lẽ bạn có thể muốn làm như vậy nếu bạn cần một thử nghiệm phức tạp hơn đã chuyển đổi đường dẫn thành đường dẫn chuẩn hoặc chuyển hướng theo sau trước khi xác định xem hai URL có cùng một phần tệp hay không.

### 1.2.2.8 Protected boolean equals(URL u1, URL u2) // Java 1.3

Phương thức bình đẳng cuối cùng được Java 1.3 thêm vào kiểm tra gần như toàn bộ URL, bao gồm giao thức, máy chủ, tệp, đường dẫn và ref. Chỉ chuỗi truy vấn bị bỏ qua. Tất cả năm trong số này phải bằng nhau để hai URL được coi là bằng nhau. Tất cả mọi thứ ngoại trừ giới thiệu là được so sánh bởi phương thức sameFile (), vì vậy việc ghi đè phương thức đó sẽ thay đổi hành vi của cái này. Các tham chiếu được so sánh bằng đẳng thức chuỗi đơn giản. Kể từ khi phương thức sameFile () sử dụng hostsEqual () để so sánh các máy chủ, phương thức này cũng vậy. Do đó, nó thực hiện tra cứu DNS nếu có thể và có thể chặn. Trong Java 1.3, equal() phương thức của lớp URL gọi phương thức này để so sánh hai đối tượng URL cho bằng nhau. Một lần nữa, bạn có thể sẽ không cần ghi đè phương thức này. Việc triển khai mặc định đủ cho hầu hết các mục đích.

### 1.2.2.9 Protected int hashCode(URL u) // Java 1.3

Java 1.3 cũng cho URLStreamHandlers cơ hội thay đổi hàm băm mặc định tính toán mã bằng cách ghi đè phương pháp này. Bạn nên làm điều này nếu bạn ghi đè equals (), sameFile () hoặc hostsEqual () để đảm bảo rằng hai URL bằng nhau

các đối tượng sẽ có cùng mã băm và hai đối tượng URL không bằng nhau sẽ không có cùng một mã băm, ít nhất là ở một mức độ xác suất rất cao.

### 1.2.3 Phương pháp kết nối

Trách nhiệm thứ hai của URLStreamHandler là tạo URLConnectionđối tượng thích hợp với URL. Điều này được thực hiện bởi phương pháp trừu tượng openConnection ().

### 1.2.3.1 Protected abstract URLConnection openConnection(URL u) throws IOException

Phương thức này phải được ghi đè trong mỗi lớp con của URLConnection. Nó có một đối số u, đó là URL để kết nối. Nó trả về một URLConnection chưa mở, trực tiếp vào tài nguyên mà u trỏ đến. Mỗi lớp con của URLStreamHandler nên biết cách tìm đúng lớp con của URLConnection cho giao thức mà nó xử lý.

Phương thức openConnection () được bảo vệ, vì vậy bạn thường không gọi nó trực tiếp; nó được gọi bởi phương thức openConnection () của một lớp URL. URL u được chuyển như một đối số là URL cần kết nối. Bạn ghi đè phương thức này trong các lớp con để xử lý một giao thức cụ thể. Phương thức openConnection () của lớp con của bạn thường cực kỳ đơn giản; trong hầu hết các trường hợp, nó chỉ gọi hàm tạo cho lớp con thích hợp của URLConnection. Ví dụ: URLStreamHandler cho giao thức mailto có thể có phương thức openConnection () giống như sau:

protected URLConnection openConnection(URL u) throws IOException {

return new com.macfaq.net.www.protocol.MailtoURLConnection(u);

}

Ví dụ 16.1 minh họa một URLStreamHandler hoàn chỉnh cho các URL mailto. Các

tên của lớp là Handler, tuân theo quy ước đặt tên của Sun. Nó giả định sự tồn tại của một lớp MailtoURLConnection.

Ví dụ 16.1. Một mailto URLStreamHandler

package com.macfaq.net.www.protocol.mailto;

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class Handler extends URLStreamHandler {

protected URLConnection openConnection(URL u) throws IOException {

return new MailtoURLConnection(u);

}

public void parseURL(URL u, String spec, int start, int limit) {

String protocol = u.getProtocol( );

String host = "";

int port = u.getPort( );

String file = ""; // really username

String userInfo = null;

String authority = null;

String query = null;

String ref = null;

if( start < limit) {

String address = spec.substring(start, limit);

int atSign = address.indexOf('@');

if (atSign >= 0) {

host = address.substring(atSign+1);

file = address.substring(0, atSign);

}

}

// For Java 1.2 comment out this next line

this.setURL(u, protocol, host, port, authority, userInfo, file, query, ref);

// In Java 1.2 and earlier uncomment the following line:

// this.setURL(u, protocol, host, port, file, ref);

}

protected String toExternalForm(URL u) {

return "mailto:" + u.getFile() + "@" + u.getHost( );;

}

}

### 1.3 Viết một trình xử lý giao thức

Để chứng minh một trình xử lý giao thức hoàn chỉnh, hãy viết một trình xử lý giao thức finger,được định nghĩa trong RFC 1288. Giao thức finger tương đối đơn giản so với các giao thức được hỗ trợ bởi JDK như HTTP và FTP. Các máy khách kết nối với cổng 79 trên máy chủ và gửi danh sách tên người dùng theo sau là cặp ký tự xuống dòng / dòng cấp dữ liệu. Máy chủ phản hồi bằng văn bản ASCII chứa thông tin về từng người dùng có tên hoặc, nếu không có tên nào được liệt kê, danh sách

người dùng hiện đã đăng nhập. Ví dụ:

% **telnet rama.poly.edu 79**

Trying 128.238.10.212...

Connected to rama.poly.edu.

Escape character is '^]'.

Login Name TTY Idle When Where

jacola Jane Colaginae \*pts/7 Tue 08:01 208.34.37.104

marcus Marcus Tullius pts/15 13d Tue 17:33 farm-dialup11.poly.e

matewan Sepin Matewan \*pts/17 17: Thu 15:32 128.238.10.177

hengpi Heng Pin \*pts/10 Tue 10:36 128.238.18.119

nadats Nabeel Datsun pts/12 56 Mon 10:38 128.238.213.227

matewan Sepin Matewan \*pts/8 4 Sun 18:39 128.238.10.177

Connection closed by foreign host.

Hoặc, yêu cầu thông tin về một người dùng cụ thể:

% **telnet rama.poly.edu 79**

Trying 128.238.10.212...

Connected to rama.poly.edu.

Escape character is '^]'.

**Marcus**

Login Name TTY Idle When Where

marcus Marcus Tullius pts/15 13d Tue 17:33 farm-dialup11.poly.e

Vì không có tiêu chuẩn nào cho định dạng của một finger URL, chúng tôi sẽ bắt đầu bằng cách tạo một URL. Lý tưởng nhất là URL này càng giống URL http càng tốt. Do đó, chúng tôi sẽ triển khai một URL ngón tay như thế này:

finger://hostname:port/username

Thứ hai, chúng ta cần xác định loại nội dung được trả về bởi giao thức finger

phương thức getContentType (). Các giao thức mới như HTTP sử dụng tiêu đề MIME để chỉ ra loại nội dung; trong những trường hợp này, bạn không cần ghi đè lên giá trị mặc định phương thức getContentType () được cung cấp bởi lớp URLConnection. Tuy nhiên, kể từ hầu hết các giao thức trước MIME, bạn thường cần chỉ định kiểu MIME một cách rõ ràng hoặc sử dụng các phương thức tĩnh URLConnection.guess Content TypeFromName (String name) và URLConnection.guessContent TypeFromStream (InputStream in) thành thực hiện một phỏng đoán có học. Ví dụ này không cần bất cứ điều gì quá phức tạp,

tuy vậy. Máy chủ finger trả về văn bản ASCII, vì vậy phương thức getContentType () sẽ trả về chuỗi text/plain. Kiểu MIME text/plain có lợi thế mà Java đã hiểu nó. Trong chương tiếp theo, bạn sẽ học cách viết nội dung trình xử lý cho phép Java hiểu các kiểu MIME bổ sung.

Ví dụ 16.2 là một lớp FingerURLConnection phân lớp URLConnection. Lớp này ghi đè các phương thức getContentType () và getInputStream () của URLConnection và thực hiện connect(). Nó cũng có một phương thức khởi tạo để xây dựng URL mới kết nối từ một URL.

Ví dụ 16.2. Lớp kết nối FingerURLC

package com.macfaq.net.www.protocol.finger;

import java.net.\*;

import java.io.\*;

public class FingerURLConnection extends URLConnection {

private Socket connection = null;

public final static int DEFAULT\_PORT = 79;

public FingerURLConnection(URL u) {

super(u);

}

public synchronized InputStream getInputStream( ) throws IOException {

if (!connected) this.connect( );

InputStream in = this.connection.getInputStream( );

return in;

}

public String getContentType( ) {

return "text/plain";

}

public synchronized void connect( ) throws IOException {

if (!connected) {

int port = url.getPort( );

if ( port < 1 || port > 65535) {

port = DEFAULT\_PORT;

}

this.connection = new Socket(url.getHost( ), port);

OutputStream out = this.connection.getOutputStream( );

String names = url.getFile( );

if (names != null && !names.equals("")) {

// delete initial /

names = names.substring(1);

names = URLDecoder.decode(names);

byte[] result;

try {

result = names.getBytes("ASCII");

}

catch (UnsupportedEncodingException e) {

result = names.getBytes( );

}

out.write(result);

}

out.write('\r');

out.write('\n');

out.flush( );

this.connected = true;

}

}

}

Lớp này có hai trường. kết nối là một Socket giữa máy khách và máy chủ. Cả phương thức getInputStream () và phương thức connect () đều cần quyền truy cập vào trường này, vì vậy nó không thể là một biến cục bộ. Trường thứ hai là DEFAULT\_PORT, một trường final static int, chứa cổng mặc định của giao thức finger; cổng này được sử dụng nếu URL không chỉ định cổng một cách rõ ràng.

Hàm tạo của lớp không có gì đáng ngạc nhiên. Nó chỉ gọi hàm tạo của lớp cha với cùng một đối số, URL u. Phương thức connect () mở ra một kết nối với máy chủ được chỉ định trên cổng được chỉ định hoặc, nếu không có cổng nào được chỉ định, thì chuyển sang mặc định cổng finger, 79. Nó gửi yêu cầu cần thiết đến máy chủ finger. Nếu bất kỳ tên người dùng nào đã được chỉ định trong phần tệp của URL, chúng được gửi đi. Nếu không, một dòng trống sẽ được gửi. Giả sử kết nối được mở thành công (không có ngoại lệ nào được đưa ra), nó sẽ đặt trường boolean được kết nối với true. Nhớ lại chương trước rằng kết nối là một trường được bảo vệ trong java.net.URLConnection, được kế thừa bởi lớp con này. Ổ cắm connect () được lưu trữ trong kết nối trường để sử dụng sau này bởi getInputStream (). Phương thức connect () và getInputStream () được đồng bộ hóa để tránh tình trạng chạy đua có thể xảy ra trên biến được kết nối.

Phương thức getContentType () trả về một Chuỗi chứa kiểu MIME cho

dữ liệu. Điều này được sử dụng bởi phương thức getContent () của java.net.URLConnection để chọn trình xử lý nội dung thích hợp. Dữ liệu được trả về bởi một máy chủ finger gần như luôn là văn bản ASCII hoặc một số ước lượng hợp lý của chúng, vì điều này phương thức getContentType () luôn trả về text/plain. GetInputStream () phương thức trả về một InputStream, mà nó nhận được từ Socket kết nối đã tạo. Nếu kết nối chưa được thiết lập khi getInputStream () được gọi, chính phương thức gọi connect ().

Khi bạn có URLConnection, bạn cần một lớp con của URLStreamHandler biết cách xử lý một máy chủ finger. Lớp này cần một openConnection ()

phương pháp tạo Kết nối FingerURLC mới từ một URL. Vì chúng tôi đã xác định

finger URL để nó giống với URL http, chúng tôi không cần triển khai phương thức parseURL (). Ví dụ 16.3 là một trình xử lý luồng cho giao thức finger. Vì thời điểm này, chúng tôi sẽ sử dụng quy ước của Sun để đặt tên cho các trình xử lý giao thức, vì vậy chúng tôi gọi trình xử lý lớp này và đặt nó vào gói com.macfaq.net.www.protocol.finger.

Ví dụ 16.3. Lớp xử lý finger

package com.macfaq.net.www.protocol.finger;

import java.net.\*;

import java.io.\*;

public class Handler extends URLStreamHandler {

public int getDefaultPort( ) {

return 79;

}

protected URLConnection openConnection(URL u) throws IOException {

return new FingerURLConnection(u);

}

}

Bạn có thể sử dụng HotJava để kiểm tra trình xử lý giao thức này. Thêm dòng sau vào tệp .hotjava / properties của bạn hoặc một số nơi khác mà từ đó HotJava sẽ tải nó:

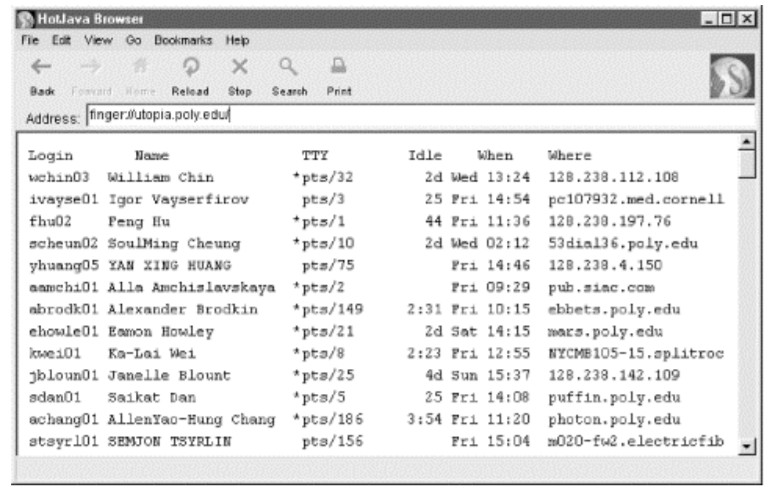
java.protocol.handler.pkgs=com.macfaq.net.www.protocol

Một số (nhưng không phải tất cả) phiên bản HotJava cũng có thể cho phép bạn đặt thuộc tính từ dòng lệnh:

% **hotjava -Djava.protocol.handler.pkgs=com.macfaq.net.www.protocol**

Bạn cũng cần đảm bảo rằng các lớp của bạn ở đâu đó trong đường dẫn lớp của HotJava. Lưu ý rằng HotJava thường không sử dụng biến môi trường CLASSPATH để tìm kiếm các lớp, vì vậy chỉ cần đặt chúng ở một nơi nào đó mà JDK hoặc JRE có thể tìm thấy chúng có thể không đủ. Sử dụng HotJava 3.0 trên Windows với JDK 1.3b1, tôi đã có thể để đưa các lớp của tôi vào thư mục jdk1.3 / jre / lib / class. Số dặm của bạn có thể thay đổi tùy thuộc vào phiên bản HotJava bạn đang sử dụng với phiên bản JDK nào trên nền tảng nào.

Chạy nó và yêu cầu URL của một trang web đang chạy, chẳng hạn như utopia.poly.edu. Ảnh 16.1 hiển thị kết quả.



### 1.4 Thêm các ví dụ và kỹ thuật về xử lý giao thức

Bây giờ bạn đã thấy cách viết một trình xử lý giao thức, nó không khó để viết . Hãy nhớ năm bước cơ bản để tạo một trình xử lý giao thức mới:

1. Thiết kế một URL cho giao thức, nếu một URL chuẩn cho giao thức đó chưa tồn tại. Kể từ tháng 7 năm 2000, danh sách chính thức của các lược đồ URL tại IANA (http://www.isi.edu/in-notes/iana/assignments/url-schemes) chỉ bao gồm 29 chương trình khác nhau và dự trữ ba nữa. Đối với bất kỳ điều gì khác, bạn cần phải xác định của riêng bạn. Làm cho URL mới của bạn giống với URL http nhất có thể.
2. Quyết định loại MIME nào sẽ được trả về bởi trình xử lý giao thức phương thức getContentType (). Loại nội dung text/plain thường thích hợp cho các giao thức kế thừa. Một tùy chọn khác là chuyển đổi dữ liệu đến sang HTML bên trong getInputStream () và trả về text / html. Dữ liệu nhị phân thường sử dụng một trong nhiều loại ứng dụng. Trong một số trường hợp, bạn có thể sử dụng URLConnection.guessContentTypeFromName () hoặc phương thức URLConnection.guessContentTypeFromStream () để xác định đúng loại MIME.
3. Viết một lớp con của URLConnection hiểu giao thức này. Nó nên triển khai phương thức connect () và có thể ghi đè getContent Type( ), getOutputStream( ) và getInputStream () của URL Connection. Nó cũng cần một phương thức khởi tạo để xây dựng một URLConnection mới từ một URL.
4. Viết một lớp con của URLStreamHandler bằng phương thức openConnection () biết cách trả về một phiên bản mới của lớp con URLConnection của bạn. Đồng thời cung cấp phương thức getDefaultPort () trả về cổng nổi tiếng cho giao thức. Nếu URL của bạn không giống URL http, hãy ghi đè parseURL () và toExternalForm ().
5. Triển khai giao diện URLStreamHandlerFactory và createStream Handler( )trong một lớp thuận tiện.

Hãy xem xét các trình xử lý để biết thêm hai giao thức daytime và chargen, sẽ hiển thị những thứ khác nhau.

### 1.4.1 Trình xử lý giao thức daytime

Đối với trình xử lý giao thức daytime, giả sử rằng URL sẽ giống như

daytime: //metalab.unc.edu. Chúng tôi sẽ cho phép chuyển nhượng cổng không chuẩn trong cùng một như với HTTP: theo sau tên máy chủ bằng dấu hai chấm và cổng (daytime: //metalab.unc.edu: 2082). Cuối cùng, cho phép một dấu gạch chéo kết thúc và bỏ qua mọi thứ sau dấu gạch chéo. Ví dụ:

daytime: //metalab.unc.edu/index.html là

tương đương với daytime: //metalab.unc.edu. Điều này đủ tương tự với một URL http bạn sẽ có thể sử dụng các phương thức toExternalForm () và parseURL () mặc định.

Mặc dù nội dung được trả về bởi giao thức daytime thực sự là text / plain, điều này trình xử lý giao thức sẽ định dạng lại dữ liệu thành một trang HTML. Sau đó, nó có thể trở lại một loại nội dung của text / html và để trình duyệt web hiển thị nó một cách rõ ràng hơn. HTML kết quả sẽ giống như sau:

<html><head><title>The Time at metalab.unc.edu</title></head><body>

<h1>Fri Oct 29 14:32:07 1999</h1>

</body></html>

Bí quyết là trang có thể được chia thành ba chuỗi khác nhau:

• Everything before the time

• The time

• Everything after the time

Chuỗi đầu tiên và chuỗi thứ ba có thể được tính toán trước khi kết nối được mở. Chúng tôi sẽ định dạng chúng dưới dạng mảng byte của văn bản ASCII và sử dụng chúng để tạo hai ByteArrayInputStreams. Sau đó, chúng tôi sẽ sử dụng SequenceInputStream để kết hợp những hai luồng với dữ liệu thực sự được trả về từ máy chủ. Ví dụ 16.4. Đây là một mẹo nhỏ cho các giao thức chẳng hạn như daytime trả lại số lượng dữ liệu rất hạn chế; nó có thể được chèn vào một nơi duy nhất trong tài liệu HTML. Các giao thức như finger trả về văn bản phức tạp hơn và ít dự đoán hơn có thể cần để sử dụng FilterInputStream chèn HTML nhanh chóng thay thế. Và dĩ nhiên khả năng thứ ba chỉ đơn giản là trả về một loại nội dung tùy chỉnh và sử dụng một nội dung tùy chỉnh xử lý để hiển thị nó. Tùy chọn thứ ba này sẽ được khám phá trong chương tiếp theo.

### Example 16.4. The DaytimeURLConnection Class

package com.macfaq.net.www.protocol.daytime;

import java.net.\*;

import java.io.\*;

public class DaytimeURLConnection extends URLConnection {

private Socket connection = null;

public final static int DEFAULT\_PORT = 13;

public DaytimeURLConnection (URL u) {

super(u);

}

public synchronized InputStream getInputStream( ) throws IOException {

if (!connected) connect( );

String header = "<html><head><title>The Time at " + url.getHost( ) + "</title></head><body><h1>";

String footer = "</h1></body></html>";

InputStream in1 = new

ByteArrayInputStream(header.getBytes("8859\_1"));

InputStream in2 = this.connection.getInputStream( );

InputStream in3 = new

ByteArrayInputStream(footer.getBytes("8859\_1"));

SequenceInputStream result = new SequenceInputStream(in1, in2);

result = new SequenceInputStream(result, in3);

return result;

}

public String getContentType( ) {

return "text/html";

}

public synchronized void connect( ) throws IOException {

if (!connected) {

int port = url.getPort( );

if ( port <= 0 || port > 65535) {

port = DEFAULT\_PORT;

}

this.connection = new Socket(url.getHost( ), port);

this.connected = true;

}

}

}

Lớp này khai báo hai trường. Đầu tiên là connection, là một Socket giữa

máy khách và máy chủ. Trường thứ hai là DEFAULT\_PORT, một biến final static int, giữ cổng mặc định cho giao thức daytime (cổng 13) và được sử dụng nếu URL không chỉ định cổng một cách rõ ràng.

Người xây dựng không có bất ngờ. Nó chỉ gọi hàm tạo của lớp cha với cùng một đối số, URL u. Phương thức connect () mở một kết nối đến máy chủ được chỉ định trên cổng được chỉ định (hoặc, nếu không có cổng nào được chỉ định, đến cổng mặc định); nếu kết nối được mở thành công, connect () đặt biến boolean được kết nối thành true. Nhớ lại chương trước rằng kết nối là một trường được bảo vệ trong URLConnection được kế thừa bởi lớp con này. Socket được mở bằng phương pháp này được lưu trữ trong trường kết nối để getInputStream () sử dụng sau này.

Phương thức getContentType () trả về một Chuỗi chứa kiểu MIME cho dữ liệu. Phương thức này được gọi bởi phương thức getContent () của URLConnection để chọn trình xử lý nội dung thích hợp. Phương thức getInputStream () định dạng lại văn bản thành HTML, vì vậy phương thức getContentType () trả về text / html.

Phương thức getInputStream () xây dựng một SequenceInputStream từ một số ký tự chuỗi, thuộc tính máy chủ của url và luồng thực được cung cấp bởi Socket kết nối máy khách với máy chủ. Nếu ổ cắm không được kết nối khi phương thức này được gọi, thì phương thức gọi connect () để thiết lập kết nối.

Tiếp theo, bạn cần một lớp con của URLStreamHandler biết cách xử lý một máy chủ daytime. Lớp này cần một phương thức openConnection () để tạo một DaytimeURLConnection mới từ một URL và một phương thức getDefaultPort () trả về daytime cổng 13. Vì daytime URL đã được tạo tương tự như một URL http, chúng tôi không cần để ghi đè parseURL (); khi chúng ta đã viết openConnection (), chúng ta đã hoàn tất. Ví dụ 16.5 cho thấy URLStreamHandler của giao thức daytime.

### Ví dụ 16.5. Lớp DaytimeURLStreamHandler

package com.macfaq.net.www.protocol.daytime;

import java.net.\*;

import java.io.\*;

public class Handler extends URLStreamHandler {

public int getDefaultPort( ) {

return 13;

}

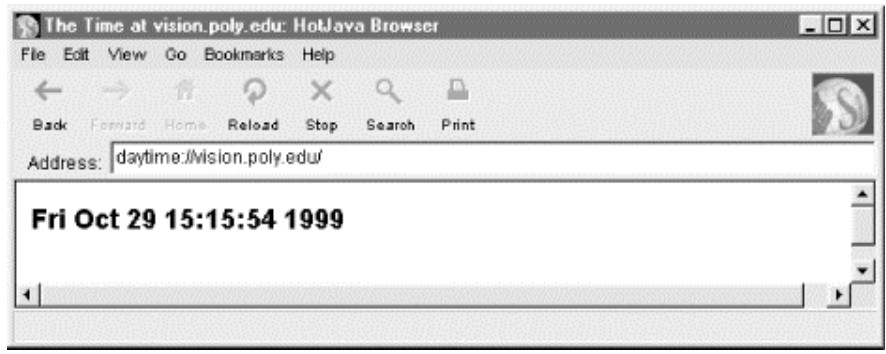
protected URLConnection openConnection(URL u) throws IOException {

return new DaytimeURLConnection(u);

}

}

Vì chúng tôi đã sử dụng quy ước đặt tên gói ở đây giống như đối với trình xử lý giao thức finger trước đó, nên không cần thực hiện thêm thay đổi nào đối với các thuộc tính của HotJava để cho phép HotJava tìm thấy điều này. Chỉ cần biên dịch các tệp, đặt các lớp ở đâu đó trong đường dẫn lớp của HotJava và tải một URL trỏ đến một máy chủ daytime. Hình 16.2 minh họa.



### 16.4.2 Trình xử lý giao thức hàng loạt

Giao thức chargen, được định nghĩa trong RFC 864, là một giao thức rất đơn giản được thiết kế để kiểm tra các máy khách. Máy chủ lắng nghe các kết nối trên cổng 19. Khi một máy khách kết nối, máy chủ sẽ gửi một luồng ký tự vô tận cho đến khi máy khách ngắt kết nối. Bất kỳ đầu vào nào từ máy khách đều bị bỏ qua. RFC không chỉ định chuỗi ký tự nào sẽ gửi, nhưng khuyến nghị rằng máy chủ sử dụng một mẫu dễ nhận biết. Một mẫu phổ biến là xoay, các dòng phân cách 72 ký tự xuống dòng / dòng cấp dữ liệu của 95 ký tự in ASCII như thế này:

!"#$%&'( )\*+,-

./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^\_`abcdefgh

"#$%&'( )\*+,-

./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^\_`abcdefghi

#$%&'( )\*+,-

./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^\_`abcdefghij

$%&'( )\*+,-

./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^\_`abcdefghijk

%&'( )\*+,-

./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^\_`abcdefghijkl

&'( )\*+,-

./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^\_`abcdefghijklm

'( )\*+,-

./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^\_`abcdefghijklmn

( )\*+,-

./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^\_`abcdefghijklmno

Bí quyết lớn với giao thức này là quyết định khi nào nên dừng lại. Máy chủ ký tự TCP gửi một lượng dữ liệu không giới hạn. Hầu hết các trình duyệt web không giải quyết tốt vấn đề này. HotJava thậm chí sẽ không cố gắng hiển thị một tệp cho đến khi nhìn thấy phần cuối của luồng. Do đó, điều đầu tiên chúng ta cần là một lớp con FilterInputStream cắt máy chủ (hoặc ít nhất là bắt đầu bỏ qua nó) sau khi một lượng dữ liệu nhất định đã được gửi đi. Ví dụ 16.6 là một lớp như vậy.

### Ví dụ 16.6. FiniteInputStream

package com.macfaq.io;

import java.io.\*;

public class FiniteInputStream extends FilterInputStream {

private int limit = 8192;

private int bytesRead = 0;

public FiniteInputStream(InputStream in) {

this(in, 8192);

}

public FiniteInputStream(InputStream in, int limit) {

super(in);

this.limit = limit;

}

public int read( ) throws IOException {

if (bytesRead >= limit) return -1;

int c = in.read( );

bytesRead++;

return c;

}

public int read(byte[] data) throws IOException {

return this.read(data, 0, data.length);

}

public int read(byte[] data, int offset, int length) throws IOException {

if (data == null) throw new NullPointerException( );

else if ((offset < 0) || (offset > data.length) || (length < 0) || ((offset + length) > data.length) || ((offset + length) < 0)) {

throw new IndexOutOfBoundsException( );

}

else if (length == 0) {

return 0;

}

if (bytesRead >= limit) return -1;

else if (bytesRead + length > limit) {

int numToRead = bytesRead + length - limit;

int numRead = in.read(data, offset, numToRead);

if (numRead == -1) return -1;

bytesRead += numRead;

return numRead;

}

else { // will not exceed limit

int numRead = in.read(data, offset, length);

if (numRead == -1) return -1;

bytesRead += numRead; return numRead;

}

}

public int available( ) throws IOException {

if (bytesRead >= limit) return 1;

else return in.available( );

}

}

Tiếp theo, vì không có tiêu chuẩn nào cho định dạng của một URL ký tự, nên chúng tôi phải tạo một định dạng. Lý tưởng nhất là URL này càng giống URL http càng tốt. Do đó, chúng tôi sẽ triển khai một URL ký tự như thế này:

chargen://hostname:port

Thứ hai, chúng ta cần chọn kiểu nội dung được trả về bởi phương thức getContentType () của giao thức chargen. Máy chủ chargen trả về văn bản ASCII, vì vậy phương thức getContentType () sẽ trả về chuỗi text/plain. Kiểu MIME text/plain có ưu điểm là Java đã hiểu nó.

Ví dụ 16.7 là một lớp ChargenURLConnection phân lớp con URLConnection. Lớp này ghi đè các phương thức getContentType () và getInputStream () của URLConnection và thực hiện connect (). Nó cũng có một hàm tạo xây dựng một URLConnection mới từ một URL.

package com.macfaq.net.www.protocol.chargen;

import java.net.\*;

import java.io.\*;

import com.macfaq.io.\*;

public class ChargenURLConnection extends URLConnection {

private Socket connection = null;

public final static int DEFAULT\_PORT = 19;

public ChargenURLConnection(URL u) {

super(u);

}

public synchronized InputStream getInputStream( ) throws IOException {

if (!connected) this.connect( );

return new FiniteInputStream(this.connection.getInputStream( ));

}

public String getContentType( ) {

return "text/plain";

}

public synchronized void connect( ) throws IOException {

if (!connected) {

int port = url.getPort( );

if ( port < 1 || port > 65535) {

port = DEFAULT\_PORT;

}

this.connection = new Socket(url.getHost( ), port);

this.connected = true;

}

}

}

Lớp này có hai trường. Connectionlà một Socket giữa máy khách và máy chủ. Trường thứ hai là DEFAULT\_PORT, một final static int chứa cổng mặc định của giao thức chargen; cổng này được sử dụng nếu URL không chỉ định cổng rõ ràng.

Phương thức khởi tạo của lớp chỉ chuyển URL u tới phương thức khởi tạo của lớp cha. Phương thức connect () mở kết nối đến máy chủ được chỉ định trên cổng được chỉ định (hoặc, nếu không có cổng nào được chỉ định, thì đến cổng chargen mặc định, 19) và, giả sử kết nối được mở thành công, hãy đặt trường boolean được kết nối thành true . Socket connect () mở ra được lưu trữ trong kết nối trường để getInputStream () sử dụng sau này. Phương thức connect () được đồng bộ hóa để tránh điều kiện chạy đua có thể xảy ra trên biến được kết nối.

Phương thức getContentType () trả về một Chuỗi chứa kiểu MIME cho dữ liệu. Dữ liệu do máy chủ chargen trả về luôn là văn bản ASCII, vì vậy phương thức getContentType () này luôn trả về text / plain.

GetInputStream () kết nối nếu cần, sau đó lấy InputStream từ this.connection. Thay vì trả lại nó ngay lập tức, getInputStream () đầu tiên chuỗi nó thành FiniteInputStream.

Bây giờ chúng ta đã có URLConnection, chúng ta cần một lớp con của URLStreamHandler biết cách xử lý một máy chủ chargen. Lớp này cần một phương thức openConnection () để xây dựng một ChargenURLConnection mới từ một URL và một phương thức getDefaultPort () trả về cổng chargen nổi tiếng. Vì chúng tôi đã xác định URL chargen để nó giống với URL http, nên chúng tôi không cần triển khai phương thức parseURL (). Ví dụ 16.8 là một trình xử lý luồng cho giao thức chargen.

### Ví dụ 16.8. Lớp xử lý chargen

package com.macfaq.net.www.protocol.chargen;

import java.net.\*;

import java.io.\*;

public class Handler extends URLStreamHandler {

public int getDefaultPort( ) {

return 19;

}

protected URLConnection openConnection(URL u) throws IOException {

return new ChargenURLConnection(u);

}

}

Bạn có thể sử dụng HotJava để kiểm tra trình xử lý giao thức này. Chạy nó và yêu cầu URL của một trang web đang chạy một máy chủ chargen, chẳng hạn như vision.poly.edu. Hình 16.3 cho thấy kết quả.

### 16.5 Giao diện URLStreamHandlerFactory

Phần cuối cùng hướng dẫn bạn cách cài đặt các trình xử lý giao thức mới mà bạn đã viết vào HotJava, một ứng dụng mà người khác đã viết. Tuy nhiên, nếu bạn viết ứng dụng của riêng mình, bạn có thể triển khai lược đồ của riêng mình để tìm và tải các trình xử lý giao thức nhập. Cách dễ nhất để làm điều này là cài đặt URLStream HandlerFactory trong ứng dụng của bạn:

Giao diện URLStreamHandlerFactory khai báo một phương thức duy nhất, createURLStreamHandler ():

public abstract URLStreamHandler createURLStreamHandler(String protocol)

Phương thức này tải trình xử lý giao thức thích hợp cho giao thức được chỉ định. Để sử dụng phương pháp này, hãy viết một lớp triển khai giao diện URLStreamHandlerFactory và bao gồm một phương thức createURLStreamHandler () trong lớp đó. Phương pháp này cần biết cách tìm trình xử lý giao thức cho một giao thức nhất định. Điều này không phức tạp hơn việc biết tên và gói của các giao thức tùy chỉnh mà bạn đã triển khai.

Phương thức createURLStreamHandler () không cần biết tên của tất cả các trình xử lý giao thức đã cài đặt. Nếu nó không nhận ra một giao thức, thì nó sẽ chỉ trả về null. Điều này yêu cầu Java tun theo quy trình mặc định để định vị trình xử lý luồng; nghĩa là, để tìm kiếm một lớp có tên là protocol.Handler trong một trong các gói được liệt kê trong thuộc tính hệ thống java.protocol.handler.pkgs hoặc trong sun.net.www.protocol.

Để cài đặt nhà máy xử lý luồng, hãy chuyển một phiên bản của lớp triển khai giao diện URLStreamHandlerFactory tới phương thức tĩnh URL.setURLStreamHandlerFactory () khi bắt đầu chương trình của bạn. Ví dụ 16.9 là một URLStreamHandlerFactory () có phương thức createURLStreamHandler () nhận dạng các giao thức finger, daytime và chargen và trả về trình xử lý thích hợp từ một số ví dụ trước. Vì tất cả các lớp này đều được đặt tên là Trình xử lý, nên các tên đủ điều kiện gói hoàn toàn được sử dụng.

### Ví dụ 16.9. URLStreamHandlerFactory cho finger, daytime và chargen

package com.macfaq.net.www.protocol;

import java.net.\*;

public class NewFactory implements URLStreamHandlerFactory {

public URLStreamHandler createURLStreamHandler(String protocol) {

if (protocol.equalsIgnoreCase("finger")) {

return new com.macfaq.net.www.protocol.finger.Handler( );

}

else if (protocol.equalsIgnoreCase("chargen")) {

return new com.macfaq.net.www.protocol.chargen.Handler( );

}

else if (protocol.equalsIgnoreCase("daytime")) {

return new com.macfaq.net.www.protocol.daytime.Handler( );

}

else {

return null;

}

}

}

Chúng tôi sử dụng phương thức equalsIgnoreCase () từ java.lang.String để kiểm tra danh tính của giao thức vì nó sẽ không tạo ra sự khác biệt cho dù bạn yêu cầu finger: //rama.poly.edu hay FINGER: //RAMA.POLY. EDU. Nếu giao thức được nhận dạng, thì createURLStreamHandler () tạo một thể hiện của lớp Xử lý thích hợp và trả về nó; nếu không, phương thức trả về giá trị null, điều này cho lớp URL biết để tìm kiếm URLStreamHandler ở các vị trí tiêu chuẩn.

Vì các trình duyệt, bao gồm HotJava, thường không cho phép bạn cài đặt URLStreamHandlerFactory của riêng mình, điều này sẽ chỉ được sử dụng trong các ứng dụng. Ví dụ 16.10 là một chương trình chế độ ký tự đơn giản sử dụng nhà máy này và các trình xử lý giao thức liên quan của nó để in dữ liệu máy chủ trên System.out. Lưu ý rằng nó không nhập com.macfaq.net.www.protocol.chargen, com.macfaq.net.www.protocol.finger hoặc com.macfaq.net.www.protocol.daytime. Tất cả những gì chương trình này biết là nó có một URL. Nó không cần biết giao thức đó được xử lý như thế nào hoặc thậm chí cách khởi tạo đối tượng URLConnection phù hợp.

### Ví dụ 16.10. Chương trình SourceViewer đặt URLStreamHandlerFactory

import java.net.\*;

import java.io.\*;

import com.macfaq.net.www.protocol.\*;

public class SourceViewer3 {

public static void main (String[] args) {

URL.setURLStreamHandlerFactory(new NewFactory( ));

if (args.length > 0) {

try {

//Open the URL for reading

URL u = new URL(args[0]);

InputStream in = new BufferedInputStream(u.openStream( ));

// chain the InputStream to a Reader

Reader r = new InputStreamReader(in);

int c;

while ((c = r.read( )) != -1) {

System.out.print((char) c);

}

}

catch (MalformedURLException e) {

System.err.println(args[0] + " is not a parseable URL");

}

catch (IOException e) {

System.err.println(e);

}

}

// end if

}

// end main

}

// end SourceViewer3

Ví dụ: ở đây chương trình đọc từ một finger URL:

D:\JAVA\JNP2\examples\15>java SourceViewer3 finger://rama.poly.edu/

Login Name TTY Idle When Where

nadats Nabeel Datsun pts/0 55 Fri 16:54 128.238.213.227

marcus Marcus Tullius \*pts/1 20 Thu 12:12 128.238.10.177

marcus Marcus Tullius \*pts/5 2:24 Thu 16:42 128.238.10.177

wri Weber Research Insti pts/10 55 Fri 13:26 rama.poly.edu

jbjovi John B. Jovien pts/9 25d Mon 14:54 128.238.213.229

Ở đây nó đọc từ một daytime URL:

% java SourceViewer3 daytime://tock.usno.navy.mil/

<html><head><title>The Time at

tock.usno.navy.mil</title></head><body><h1>Fri Oc

t 29 21:22:49 1999

</h1></body></html>

Tuy nhiên, nó vẫn hoạt động với tất cả các trình xử lý giao thức thông thường đi kèm với JDK. Ví dụ, đây là một vài dòng đầu tiên của đầu ra khi nó đọc từ một URL http:

% java SourceViewer3 <http://www.oreilly.com/oreilly/about.html>

<HTML>

<HEAD>

<TITLE>About O'Reilly &amp; Associates</TITLE>

</HEAD>

<BODY LINK="#770000" VLINK="#0000AA" BGCOLOR="#ffffff">

<table border=0 cellspacing=0 cellpadding=0 width=515>

<tr>

<td>

<img

src="http://www.oreilly.com/graphics\_new/generic\_ora\_header\_wide.gif"

width="515" height="37" ALT="O'Reilly and Associates">

# Chương 2.

## Chapter17. Trình xử lí nội dung

Trình xử lý nội dung là một trong những ý tưởng khiến các nhà phát triển hào hứng với Java trong thời điểm đầu tiên. Tại thời điểm HotJava được tạo, Netscape, NCSA, Spyglass và một số đối thủ khác đang chiến đấu trong một trận chiến xem ai sẽ kiểm soát các tiêu chuẩn cho duyệt web. Một trong những chiến trường là khả năng của các trình duyệt khác nhau xử lý nhiều loại tệp khác nhau. Các trình duyệt đầu tiên chỉ hiểu HTML. Tiếp theo thế hệ hiểu HTML và GIF. Hỗ trợ JPEG sớm được thêm vào. Cường độ của cuộc chiến này có nghĩa là các phiên bản trình duyệt mới đã được phát hành sau mỗi hàng tuần. Netscape đã thực hiện nỗ lực đầu tiên để phá vỡ vòng lặp vô hạn này bằng cách giới thiệu các plug-in trong Navigator 2.0. Plugin là các trình mở rộng trình duyệt phụ thuộc vào nền tảng được viết bằng C bổ sung khả năng xem các loại nội dung mới như Adobe PDF và VRML.

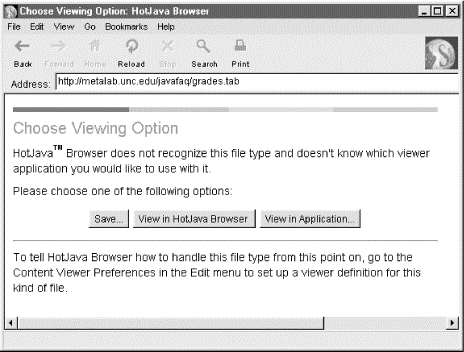
Tuy nhiên, plug-in có những hạn chế của chúng. Mỗi loại nội dung mới yêu cầu người dùng tải xuống và cài đặt một trình cắm mới, nếu thực sự là trình cắm phù hợp thậm chí có sẵn cho nền tảng của người dùng. Để theo kịp, người dùng đã phải sử dụng một lượng lớn băng thông chỉ để tải xuống các trình duyệt và trình cắm mới, mỗi trình cắm sẽ sửa một số lỗi và thêm một số tính năng mới.

Nhóm Java đã tìm ra cách giải quyết vấn đề này. Ý tưởng của họ là chỉ sử dụng Java để tải xuống các phần của chương trình phải được cập nhật thay vì toàn bộ trình duyệt. Hơn nữa, khi người dùng gặp một trang web sử dụng loại nội dung mới, trình duyệt có thể tự động tải xuống mã cần thiết để xem nội dung đó kiểu. Người dùng sẽ không phải dừng, chuyển sang một trình cắm, thoát khỏi trình duyệt, cài đặt trình cắm, khởi động lại trình duyệt và tải lại trang. Cơ chế mà nhóm Java đã hình dung là trình xử lý nội dung. Mỗi kiểu dữ liệu mới mà một trang web muốn cung cấp sẽ được liên kết với một trình xử lý nội dung được viết bằng Java. Trình xử lý nội dung sẽ chịu trách nhiệm phân tích cú pháp nội dung và hiển thị nó cho người dùng trong trình duyệt cửa sổ web. Lớp trừu tượng mà nội dung xử lý cho các kiểu dữ liệu cụ thể như PNG hoặc RTF sẽ mở rộng là java.net.ContentHandler. James Gosling và Henry McGilton đã mô tả kịch bản này vào năm 1996:

Hành vi động của HotJava cũng được sử dụng để hiểu các các loại đối tượng. Ví dụ: hầu hết các trình duyệt Web có thể hiểu một tập hợp nhỏ các định dạng hình ảnh (thường là GIF, pixmap X11 và X11 bitmap). Nếu họ thấy một số loại khác, họ không có cách nào để đối phó với nó. Mặt khác, HotJava có thể liên kết động mã từ máy chủ có hình ảnh cho phép nó hiển thị định dạng mới. Vì vậy nếu ai đó phát minh ra một thuật toán nén mới, nhà phát minh chỉ cần đảm bảo rằng một bản sao mã Java của nó được cài đặt trên máy chủ chứa những hình ảnh mà họ muốn xuất bản; họ không phải nâng cấp tất cả các trình duyệt trên thế giới. Về cơ bản, HotJava tự nâng cấp khi đang vận hành nó nhìn thấy loại mới này.

Thật không may, những người xử lý nội dung không bao giờ thực sự đưa nó ra khỏi sách trắng của Sun thành phần mềm vận chuyển. Lớp ContentHandler vẫn tồn tại trong thư viện chuẩn và nó có một số công dụng trong các ứng dụng tùy chỉnh. Tuy nhiên, không phải HotJava hay bất kỳ web nào khác trình duyệt thực sự sử dụng nó để hiển thị nội dung. Khi HotJava tải xuống một trang HTML hoặc một hình ảnh được ánh xạ bit, nó được xử lý bằng các quy trình được mã hóa cứng để xử lý loại dữ liệu. Khi HotJava gặp một loại nội dung không xác định, nó chỉ cần hỏi người dùng để tìm một ứng dụng trợ giúp có thể hiển thị tệp, gần như chính xác như một trình duyệt web truyền thống như Netscape Navigator hoặc Internet Explorer sẽ làm được. (Hình 17.1 minh họa.) Lời hứa về các trình duyệt web có thể mở rộng động tự động tải xuống trình xử lý nội dung cho các loại dữ liệu mới khi chúng gặp phải chúng không bao giờ được nhận ra. Có lẽ vấn đề lớn nhất là ContentHandler lớp quá chung chung, cung cấp quá ít thông tin về loại đối tượng đang được tải xuống và nó sẽ được hiển thị như thế nào.

Hình 17.1. Phản ứng của HotJava đối với một loại nội dung không mong muốn, ngay cả khi một nội dung trình xử lý cho loại này đã được cài đặt



 Một trình xử lý nội dung có tư duy tốt hơn và mạnh mẽ hơn nhiều cơ chế hiện có sẵn dưới tên JavaBeans Khung kích hoạt. Đây là một phần mở rộng tiêu chuẩn cho Java 1.1 và sau đó cung cấp API cần thiết để quyết định những việc cần làm với các kiểu dữ liệu tùy ý trong thời gian chạy. Tuy nhiên, JAF vẫn chưa được sử dụng bên trong các trình duyệt web hoặc thậm chí được chấp nhận rộng rãi, mặc dù chắc chắn điều đó sẽ không ngăn bạn sử dụng nó bên trong ứng dụng của riêng bạn nếu bạn thấy nó hữu ích. Truy cập http://java.sun.com/beans/glasgow/jaf.html để biết thêm chi tiết.

### 17. 1 Trình xử lý nội dung là gi?

Trình xử lý nội dung là một ví dụ của lớp con của java.net.ContentHandler: public abstract class ContentHandler extends Object

Giao diện SAX2 để phân tích cú pháp XML xác định một giao diện riêng biệt có tên ContentHandler. Điều này không có gì để làm với các trình xử lý nội dung mà chúng ta đang thảo luận trong chương này.

Lớp này biết cách sử dụng URLConnection, kiểu MIME và chuyển dữ liệu từ URLConnection vào một đối tượng Java của một kiểu thích hợp. Do đó, một trình xử lý nội dung cho phép một applet hiểu các loại dữ liệu mới. Kể từ khi Java hạ thấp thanh để viết mã bên dưới những gì cần thiết để viết một trình duyệt hoặc một trình cắm thêm Netscape, lý thuyết là nhiều trang web khác nhau có thể viết các trình xử lý tùy chỉnh, thay vì phải dựa vào các nhà sản xuất trình duyệt làm việc quá sức.

Java đã có thể tái các lớp từ Initemet. Do đó, không có nhiều điều kỳ diệu để khiến nó tải xuống một lớp có thể hiểu một loại nội dung mới. Một nội dung trình xử lý chỉ là một tập lớp giống như bất kỳ tệp nào khác. Điều kỳ diệu là tất cả bên trong trình duyệt web, trình duyệt này biết khi nào và ở đâu yêu cầu tập class để xem một loại nội dung mới. Tất nhiên, một số trình duyệt kỳ diệu hơn những trình duyệt khác. Hiện tại, cách duy nhất để làm cho điều này hoạt động trong trình duyệt là kết hợp với một applet biết cách yêu cầu trình xử lý nội dung một cách rõ ràng. Nó cũng có thể được sử dụng trên thực tế, nó có thể được sử dụng dễ dàng hơn đáng kể trong một ứng dụng độc lập hoàn toàn bỏ qua các trình duyệt.

Cụ thể, một trình xử lý nội dung đọc dữ liệu từ một Kết nối URL và tạo một đối tượng thích hợp cho loại nội dung từ dữ liệu. Mỗi lớp con của ContentHandler xử lý một kiểu MIME và kiểu con cụ thể, chẳng hạn như text/plain hoặc image/gif. Do đó, trình xử lý nội dung image/gif trả về đối tượng URL ImageSource (một lớp triển khai giao diện Image Producer), trong khi trình xử lý text/plain trả về một Chuỗi. Một trình xử lý nội dung cơ sở dữ liệu có thể trả về một đối tượng java sql ResultSet trình xử lý nội dung application/x-macbinhex40 có thể trả về một đối tượng BinhexDecoder được viết bởi cùng một lập trình viên đã viết trình xử lý nội dung application/x-macbinhex40, Java đã có thể tải các lớp từ Internet.

Vì vậy, không có nhiều thuật toán để tải xuống một lớp có thể hiểu một loại nội dung mới. Một nội dung trình xử lý chỉ là một tệp .class giống như bất kỳ tệp nào khác. Điều kỳ diệu là tất cả bên trong trình duyệt web, biết khi nào và ở đâu yêu cầu tệp .class để xem kiểu nội dung mới. Tất nhiên, một số trình duyệt kỳ diệu hơn những trình duyệt khác. Hiện tại, cách duy nhất để thực hiện hoạt động này trong một trình duyệt kết hợp với một applet biết cách yêu cầu xử lý nội dung một cách rõ ràng. Nó cũng có thể được sử dụng — trên thực tế, nó có thể được sử dụng dễ dàng hơn đáng kể — trong một ứng dụng độc lập hoàn toàn bỏ qua các trình duyệt.

Trình xử lý nội dung được gắn chặt với trình xử lý giao thức. Trong chương trước, Phương thức getContent () của lớp URLConnection trả về một Dòng đầu vào mà đã đưa dữ liệu từ máy chủ đến máy khách. Điều này hoạt động đối với các giao thức đơn giản trả về chỉ văn bản ASCII, chẳng hạn như finger, whois và daytime. Tuy nhiên, trả về một đầu vào luồng không hoạt động tốt đối với các giao thức như FTP, gopher và HTTP có thể trả về nhiều loại nội dung khác nhau, nhiều loại trong số đó không thể được hiểu là một dòng ASCII chữ. Đối với các giao thức như thế này, getContent () cần kiểm tra kiểu MIME và sau đó sử dụng phương thức createContentHandler () của ứng dụng ContentHandlerFactory để tạo ra một trình xử lý nội dung phù hợp. Một lần ContentHandler tồn tại, phương thức getContent () của URLConnection gọi Phương thức getContent () của ContentHandler, tạo đối tượng Java để trả lại. Bên ngoài phương thức getContent () của URLConnection, bạn hiếm khi bao giờ, hãy gọi bất kỳ phương thức ContentHandler nào. Các ứng dụng không bao giờ được gọi các phương thức của một ContentHandler trực tiếp. Thay vào đó, họ nên sử dụng phương thức getContent () của URL hoặc URLConnection. Một đối tượng triển khai giao diện ContentHandlerFactory có thể đáp ứngchọn ContentHandler phù hợp với kiểu MIME. MỘT ContentHandlerFactory được cài đặt trong một chương trình bằng cách tĩnh Phương thức URLConnection.setContentHandlerFactory (). Chỉ một ContentHandlerFactory có thể được chọn trong suốt thời gian tồn tại của ứng dụng. Khi nào một chương trình bắt đầu chạy, không có ContentHandlerFactory; đó là, ContentHandlerFactory rỗng.

Khi không có factory, Java tìm kiếm các lớp xử lý nội dung với tên type.subtype, trong đó type là kiểu MIME của nội dung và subtype là Loại phụ MIME. Nó tìm kiếm các lớp này trước tiên trong bất kỳ gói nào được đặt tên bởi thuộc tính java.content.handler.pkgs, sau đó trong gói sun.net.www.content. Thuộc tính java.content.handler.pkgs phải chứa danh sách các tiền tố gói cách nhau một thanh thẳng đứng (|). Điều này tương tự như cách Java tìm thấy các trình xử lý giao thức. Ví dụ: nếu thuộc tính java.content.handler.pkgs có value com.macfaq.net.www.content | org.cafeaulait.content và chương trình của bạn đang tìm kiếm một trình xử lý nội dung cho các tệp text / xml, thì trước tiên nó sẽ cố gắng khởi tạo com.macfaq.net.www.content.text.xml. Nếu điều đó không thành công, nó sẽ cố gắng tiếp theo Instantiate, org.cafeaulait.content.text.xml. Nếu điều đó không thành công, như một phương sách cuối cùng, nó sẽ cố gắng khởi tạo sun.net.www.content.text.xml. Những quy ước này là cũng được sử dụng để tìm kiếm trình xử lý nội dung nếu ContentHandlerFactory được cài đặt nhưng phương thức createContentHandler () trả về null.

Tóm lại, đây là chuỗi sự kiện:

1. Một đối tượng URL được tạo ra trỏ đến một số tài nguyên Internet.
2. Phương thức getContent () của URL được gọi để trả về một đối tượng đại diện nội dung của tài nguyên.
3. Phương thức getContent () của URL gọi phương thức getContent () của URLConnection bên dưới của nó.
4. Phương thức URLConnectiongetContent () gọi phương thức không công khai getContentHandler () để tìm một trình xử lý nội dung cho kiểu MIME và kiểu con.
5. getContentHandler () kiểm tra xem liệu nó đã có một trình xử lý cho loại này trong bộ nhớ cache của nó hay chưa. Nếu có, trình xử lý đó được trả về getContent () .Do đó, các trình duyệt sẽ không tải xuống các trình xử lý nội dung cho các loại thông thường, chẳng hạn như text / html, mỗi khi người dùng truy cập một trang web mới.
6. Nếu không có ContentHandler thích hợp trong bộ đệm và ContentHandlerFactory không rỗng, getContentHandler () sẽ gọi phương thức createContentHandler () của ContentHandlerFactory để khởi tạo một ContentHandler mới. Nếu điều này thành công, đối tượng ContentHandler được trả về getContent ().
7. Nếu ContentHandlerFactory là null hoặc createContentHandler () không thể khởi tạo ContentHandler mới, thì Java sẽ tìm kiếm một lớp xử lý nội dung được đặt tên là type.subtype, trong đó type là kiểu MIME của nội dung và kiểu con là kiểu con MIME trong một trong các gói có tên trong thuộc tính hệ thống java.content.handler.pkgs. Nếu một trình xử lý nội dung được tìm thấy, nó sẽ được trả về. Nếu không thì . . .
8. Java tìm kiếm một lớp xử lý nội dung có tên sun.net.www.content.type.subtype. Nếu nó được tìm thấy, nó sẽ được trả lại. Nếu không, createContentHandler () trả về null.
9. Nếu đối tượng ContentHandler không phải là null, thì phương thức getContent () của ContentHandler này được gọi. Phương thức này trả về một đối tượng phù hợp với kiểu nội dung. Nếu ContentHandler là null, thì một IOException sẽ được ném ra.
10. Đối tượng trả về hoặc ngoại lệ được chuyển vào chuỗi cuộc gọi, cuối cùng đến phương thức đã gọi getContent ().

Bạn có thể ảnh hưởng đến chuỗi sự kiện này theo ba cách: thứ nhất, bằng cách xây dựng một URL và gọi phương thức getContent () của nó; thứ hai, bằng cách tạo ContentHandler mới lớp con mà getContent () có thể sử dụng; thứ ba, bằng cách cài đặt ContentHandlerFactory với URLConnection.setContentHandlerFactory (), thay đổi cách ứng dụng tìm kiếm trình xử lý nội dung.

### 17.2 Lớp ContentHandler

Một lớp con của ContentHandler ghi đè phương thức getContent () để trả về một đối tượng tương đương với nội dung của Java. Phương pháp này có thể khá đơn giản hoặc khá phức tạp, phụ thuộc gần như hoàn toàn vào mức độ phức tạp của loại nội dung mà bạn đang cố gắng phân tích cú pháp. Một trình xử lý text / plain khá đơn giản; một trình xử lý nội dung text / rtf sẽ rất phức tạp.

Lớp ContentHandler chỉ có một hàm tạo noargs đơn giản:

public ContentHandler ()

Vì ContentHandler là một lớp trừu tượng, bạn không bao giờ gọi hàm tạo của nó một cách trực tiếp, chỉ từ bên trong các hàm tạo của các lớp con.

Phương thức chính của lớp, mặc dù là phương thức trừu tượng, là getContent ():

public abstract Object getContent(URLConnection uc) throws IOException

Phương thức này thường chỉ được gọi từ bên trong phương thức getContent () của một đối tượng URLConnection. Nó được ghi đè trong một lớp con cụ thể cho loại nội dung đang được xử lý. getContent () nên sử dụng InputStream của URLConnection để tạo một đối tượng. Không có quy tắc nào về loại đối tượng mà trình xử lý nội dung phải trả về. Nói chung, điều này phụ thuộc vào những gì ứng dụng yêu cầu nội dung mong đợi. Các trình xử lý nội dung cho nội dung dạng văn bản đi kèm với JDK trả về một số lớp con của InputStream. Trình xử lý nội dung cho hình ảnh trả về đối tượng ImageProductioner.

Phương thức getContent () của trình xử lý nội dung không nhận được InputStream đầy đủ mà URLConnection có quyền truy cập. Dòng đầu vào mà trình xử lý nội dung nhìn thấy chỉ nên bao gồm dữ liệu thô của nội dung. Mọi tiêu đề MIME hoặc thông tin cụ thể về giao thức khác đến từ máy chủ sẽ bị URLConnection loại bỏ trước khi nó chuyển luồng đến ContentHandler. ContentHandler chỉ chịu trách nhiệm về nội dung, không chịu trách nhiệm về bất kỳ chi phí giao thức nào có thể có. URLConnection lẽ ra đã thực hiện bất kỳ thao tác bắt tay cần thiết nào với máy chủ và diễn giải bất kỳ tiêu đề nào mà nó gửi.

### 17.2.1 Trình xử lý nội dung cho các giá trị được phân tách bằng tab

Để xem cách thức hoạt động của trình xử lý nội dung, hãy tạo Contenitilandler để xử lý loại nội dung text/tab-separated-values. Chúng tôi không quan tâm đến việc các giá trị được phân tách bằng tab đến với chúng tôi như thế nào. Đó là để một trình xử lý giao thức giải quyết. Tất cả những gì ContentHandler cần biết là kiểu MIME và định dạng của dữ liệu, Các giá trị được phân tách bảng tạo được tạo ra bởi nhiều chương trình cơ sở dữ liệu và bảng tỉnh. Tệp được phân tách bằng tab có thể trông giống như thế này. Các tab được biểu thị bằng các mũi tên.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| JPE Associates |  | 341 Lafayette Street, Suite 1025 |  | New York **Æ** NY **Æ** 10012 |
| O'Reilly & Associates |  | 103 Morris Street, Suite A |  | Sebastopol **Æ** CA **Æ** 95472 |

Theo cách nói của cơ sở dữ liệu, mỗi dòng là một bản ghi và dữ liệu trước mỗi tab là một trường. Thông thường (mặc dù không nhất thiết) đúng là mỗi trường có cùng ý nghĩa trong mỗi bản ghi. Trong ví dụ trước, trường đầu tiên là tên công ty.

Câu hỏi đầu tiên cần đặt ra là: loại đối tượng Java nào chúng ta nên chuyển đổi các giá trị được phân tách bằng tab thành? Cách đơn giản và tổng quát nhất để lưu trữ mỗi bản ghi là một mảng các chuỗi. Các bản ghi kế tiếp có thể được thu thập trong một Vector. Tuy nhiên, trong nhiều ứng dụng, bạn có nhiều kiến thức hơn về định dạng chính xác và ý nghĩa của dữ liệu hơn chúng tôi ở đây. Bạn càng biết nhiều về dữ liệu bạn đang xử lý, bạn càng có thể viết một ContentHandler tốt hơn. Ví dụ: nếu bạn biết rằng dữ liệu bạn đang tải xuống đại diện cho các địa chỉ ở Hoa Kỳ, thi bạn có thể xác định một lớp như sau

public class Address {

private String name;

private String street;

private String city;

private String state;

private String zip;

}

Lớp này cũng sẽ có các hàm tạo thích hợp và các phương thức khác để đại diện cho mỗi bản ghi. Trong ví dụ này, chúng tôi không biết trước bất cứ điều gì về dữ liệu hoặc chúng tôi sẽ phải lưu trữ bao nhiêu bản ghi. Do đó, chúng tôi sẽ thực hiện cách tiếp cận tổng quát nhất và chuyển đổi mỗi bản ghi thành một mảng chuỗi, sử dụng một Vector để lưu trữ mỗi mảng cho đến khi không còn bản ghi nào nữa, Phương thức getContent () có thể trả về Vector mảng chuỗi.

Ví dụ 17.1 hiển thị mã cho Contenthlandler như vậy. Goi đầy đủ đủ điều kiện tên là commacfaq net www cofiterit.text tub\_separated\_values. Tên lớp bất thường này tuân theo quy ước đặt tên cho trình xử lý nội dung cho giá trị text/tab-separated-values kiểu MIME. Vì kiểu MIME thường chứa dấu gạch nối, như trong ví dụ này, tồn tại một quy ước để thay thế những dấu gạch ngang này bằng dấu gạch dưới (). Do đó, text/tab-separated-values trở thành text.tab\_separated values. Để cài đặt trình xử lý nội dung này, tất cả những gì cần thiết là đặt tập class đã biên dịch ở nơi nào đó mã trình tái lớp có thể tìm thấy nó và đặt thuộc tỉnh java, content handler.pkgs thành com.macfag.net.www.content.

Ví dụ 17.1.A. Trinh xử lý nội dung cho text/tab-separated-values

package com.macfaq.net.www.content.text;

import java.net.\*;

import java.io.\*;

import java.util.\*;

import com.macfaq.io.SafeBufferedReader // From Chapter 4

public class tab\_separated\_values extends ContentHandler {

public Object getContent(URLConnection uc) throws IOException {

String theLine;

Vector v = new Vector( );

InputStreamReader isr = new

InputStreamReader(uc.getInputStream( ));

SafeBufferedReader in = new SafeBufferedReader(isr); while ((theLine = in.readLine( )) != null) {

String[] linearray = lineToArray(theLine); v.addElement(linearray);

}

return v;

}

private String[] lineToArray(String line) {

int numFields = 1;

for (int i = 0; i < line.length( ); i++) { if (line.charAt(i) == '\t') numFields++;

}

String[] fields = new String[numFields]; int position = 0;

for (int i = 0; i < numFields; i++) { StringBuffer buffer = new StringBuffer( );

while (position < line.length( ) && line.charAt(position) != '\t') {

buffer.append(line.charAt(position)); position++;

}

fields[i] = buffer.toString( ); position++;

}

return fields;

}

}

Ví dụ 17,1 có hai phương pháp. Phương thức tiện ích riêng lineToArray () chuyển đổi một chuỗi được phân tách bang tạo thành một mảng chuỗi. Phương thức này dành cho việc sử dụng riêng tư của lớp con này và không được giao diện Contentharidler yêu cầu. Nội dung bạn đang cố gắng phân tích càng phức tạp thì lớp của bạn càng cần nhiều phương thức như vậy. Phương thức lineToArray () bắt đầu bằng cách đếm số tab trong chuỗi. Điều này đặt biển numFieids thành một nhiều hơn số tab. Một mảng được tạo cho các trường có độ dài là numFields. Sau đó, một vòng lặp for lấp đầy mảng với các chuỗi giữa các tab. Sau đó, mảng này được trả về.

Bạn có thể đã mong đợi một StringTokenizer chia dòng thành nhiều phần. Tuy nhiên, lớp đó có những ý tưởng bất thường về những gi tạo nên mã thông – bảo nhân đồi. Đặc biệt, nó sẽ diễn giải nhiều tạo liên tiếp dưới dạng một đầu phân cách duy nhất. Nghĩa là, nó không bao giờ trả về một chuỗi rỗng dưới dạng mà thông báo phương thức getContent () bắt đầu bằng cách khởi tạo một Vector. Sau đó, nó lấy InputStream tử uc URL Connection và chuỗi này thành một InputStreamReader, lần lượt được liên kết với SafeBullered Reader được giới thiệu Chương 4, vì vậy nó có thể đọc từng dòng một trong một vòng lập. Mỗi dòng được cung cấp cho phương thức lineToArray (), phương thức này sẽ chia nó thành một mảng Chuỗi. Máng này sau đó được thêm vào Vector. Khi không còn dòng nào nữa, vòng lập sẽ thoát ra và Vecto được trả về

### 17.2 2 Sử dụng trình xử lý nội dung

Bây giờ bạn đã viết Contenthandie đầu tiên của mình, hãy xem cách sử dụng nó trong một chương trình. Các tập có văn bản kiểu MIME text/tab-separated-values có thể được cung cấp bởi máy chủ gopher, máy chủ HTTP máy chủ FTP v.v. Giả sử bạn đang truy xuất tệp giá trị được phân tách bang tab từ máy chủ HTTP. Tên tệp phải kết thúc băng đuôi tsv hoặc tab để máy chủ biết đó là tập text/tab-separated-values.

Không phải tất cả các máy chủ đều được định cấu hình để hỗ trợ loại này Tham khảo tài liệu máy chủ của bạn để xem cách thiết lập

Không phải tất cả các máy chủ đều được định cấu hình để hỗ trợ loại này. Tham khảo tài liệu máy chủ của bạn để xem cách thiết lập ảnh xa kiểu MIME cho máy chủ của bạn. Ví dụ để định cấu hình máy chủ Apache của tôi, tôi đã thêm các dòng này vào tệp .htaccess của mình:

AddType text/tab-separated-values tab

AddType text/tab-separated-values tsv

Bạn có thể kiểm tra cấu hình máy chủ web bằng cách kết nối với cổng 80 máy chủ web có Telnet và yêu cầu lập thủ công:

% **telnet metalab.unc.edu 80**

Trying 127.0.0.1...

Connected to metalab.unc.edu. Escape character is `^]'.

**GET /javafaq/addresses.tab HTTP 1.0**

HTTP 1.0 200 OK

Date: Mon, 15 Nov 1999 18:36:51 GMT

Server: Apache/1.3.4 (Unix) PHP/3.0.6 mod\_perl/1.17 Last-Modified: Thu, 04 Nov 1999 18:22:51 GMT Content-type: text/tab-separated-values

Content-length: 163

JPE Associates 341 Lafayette Street, Suite 1025 New York NY 10012

O'Reilly & Associates 103 Morris Street, Suite A Sebastopol CA 95472

Kết nối bị đóng bởi máy chủ nước ngoài.

Bạn đang tìm kiếm một dòng có nội dung Loại nội dung text/tab-separated-values tsv. Nếu bạn thấy loại Nội dung của text/plain, application / octet-stream, hoặc một số giá trị khác hoặc bạn không thay bất kỳ loại Nội dung nào, máy chủ được định cấu hình sai và phải được sửa trước khi bạn tiếp tục.

Ứng dụng sử dụng trình xử lý nội dung giá trị được phân tách bằng tab không cần biết rõ ràng về nó. Nó chỉ cần gọi phương thức getContent () của URL hoặc URL Connection trên URL có kiểu MIME phù hợp. Hơn nữa, gửi má trình xử lý nội dung có thể được tìm thấy phải được liệt kê trong thuốc tinh java.content.handlers.pkg.

Ví dụ 17,2 là một lớp tái xuống và in text/tab-separated-values sử dụng ContentHandler của Ví dụ 17.1. Tuy nhiên, lưu ý rằng nó không nhập commacfaq net www content văn bản và nó không bao giờ tham chiếu đến lớp tab\_separated\_values. Nó thêm rõ ràng commacfag.net.www.content vào thuộc tỉnh java content handlers pkgs vì đó là cách đơn giản nhất để đảm bảo chương trình độc lập này hoạt động. Tuy nhiên, các dòng làm điều đó có thể bị xoa nếu thuộc tỉnh được đặt trong tếp thuộc tỉnh hoặc từ dòng lệnh và thực sự điều này là bắt buộc trong Java 1.

Ví dụ 17 2. Lớp ContentTester được phân tách bang các gia trị bảng tạo

import java.io.\*; import java.net.\*; import java.util.\*;

public class TSVContentTester {

private static void test(URL u) throws IOException { Object content = u.getContent( );

Vector v = (Vector) content;

for (Enumeration e = v.elements() ; e.hasMoreElements( ) ;) { String[] sa = (String[]) e.nextElement( );

for (int i = 0; i < sa.length; i++) { System.out.print(sa[i] + "\t");

}

System.out.println( );

}

}

public static void main (String[] args) {

// If you uncomment these lines in Java 1.2, then you don't

// have to set the java.content.handler.pkgs property from the

// command line or your properties files.

/\* String pkgs = System.getProperty("java.content.handler.pkgs", "");

if (!pkgs.equals("")) { pkgs = pkgs + "|";

}

pkgs += "com.macfaq.net.www.content"; System.setProperty("java.content.handler.pkgs", pkgs); \*/

for (int i = 0; i < args.length; i++) { try {

URL u = new URL(args[i]); test(u);

}

catch (MalformedURLException e)

System.err.println(args[i] + " is not a good URL");

}

catch (Exception e) { e.printStackTrace( );

}

}

}

Đây là cách bạn chạy chương trình này trong Java 1.1 và 1.3. Các mũi tên chỉ ra các tab:

% **java -Djava.content.handler.pkgs=com.macfaq.net.www.content\ TSVContentTester** [**http://metalab.unc.edu/javafaq/addresses.tab**](http://metalab.unc.edu/javafaq/addresses.tab)

JPE Associates

->

341 Lafayette Street, Suite 1025

->

New York

->

NY

->

10012

O'Reilly & Associates

->

103 Morris Street, Suite A

->

Sebastopol

->

CA

->

95472

Java 1.2 phức tạp hơn vì chính sách tải lớp mới được sử dụng trong Java 1.2 ngăn trình xử lý nội dung được tải từ đường dẫn lớp cục bộ. Lỗi này đã được sửa trong Java 1.3. Tuy nhiên, trong Java 1.2, chỉ cần chạy Ví dụ 17.2 sẽ dẫn đến một ClassCastException. Vì không tìm thấy trình xử lý nội dung tùy chỉnh, getContent () trả về InputStream (cụ thể là sun.net.www.MeteredStream) thay thế:

& java TSVContentTester http://metalab.unc.edu/javafaq/addresses.tab

java.lang.ClassCastException: sun.net.www.MeteredStream

at TSVContentTester.test (TSVContentTester.java:10)

at TSVContentTester.main (TSVContentTester.java:35)

Có một số cách để giải quyết vấn đề này. Thay vào đó, bạn có thể sử dụng trình thông dịch oldjava. Thao tác này tải các trình xử lý nội dung từ đường dẫn lớp cục bộ. Ví dụ:

% **oldjava TSVContentTester** [**http://metalab.unc.edu/javafaq/addresses.tab**](http://metalab.unc.edu/javafaq/addresses.tab)

JPE Associates

->

341 Lafayette Street, Suite 1025

->

New York

->

NY

->

10012

O'Reilly & Associates

->

103 Morris Street, Suite A

->

Sebastopol

->

CA

->

95472

Ngoài ra, bạn có thể sử dụng công tắc dòng lệnh -Xbootclasspath không chuẩn để cho Java biết nơi tìm các trình xử lý nội dung. Ví dụ: dòng này yêu cầu Java tìm trong thư mục hiện tại để tìm trình xử lý nội dung và các tệp khác trước khi tìm trong tệp rt jar (vị trí chính xác của tệp rt.jar sẽ phải được điều chỉnh để phù hợp với hệ thống của bạn):

% **java -Xbootclasspath:.;/usr/local/jdk1.3/jre/lib/rt.jar\ TSVContentTester** [**http://metalab.unc.edu/javafaq/addresses.tab**](http://metalab.unc.edu/javafaq/addresses.tab)JPE

JPE Associates

->

341 Lafayette Street, Suite 1025

->

New York

->

NY

->

10012

O'Reilly & Associates

->

103 Morris Street, Suite A

->

Sebastopol

->

CA

->

95472

Lỗi yêu cầu các giải pháp thay thế này có trong tất cả các phiên bản của Java 12, mặc dù nó không tự biểu hiện trong các ứng dụng do bản chất khác nhau của ClassLoader mà trình duyệt web sử dụng. Tất nhiên, giải pháp cuối cùng là sử dụng ContentHanderFactory, một tùy chọn mà tôi sẽ thảo luận sau.

### 17.2.3 Chọn loại trả lại

Java 1.3 thêm một biến thể được nạp chồng của phương thực getContent () vào lớp ContentHandier public Object get Content (URL Connection uc, lop Class II) // Java 1.3 throws IOException

Sự khác biệt là mảng các đối tượng java.lang.Class được truyền làm đối số thứ hai. Điều này cho phép người gọi yêu cầu trả về nội dung dưới dạng một trong các kiểu trong mảng và cho phép trình xử lý nội dung hỗ trợ nhiều kiểu. Ví dụ: trình xử lý nội dung text/tab-separated-values có thể trả về dữ liệu dưới dạng Vectơ, mảng, chuỗi hoặc Dòng đầu vào. Một sẽ là mặc định được sử dụng bởi phương thức getContent () đối số duy nhất, trong khi những cái khác sẽ là các tùy chọn mà khách hàng có thể yêu cầu.

Nếu máy khách không yêu cầu bất kỳ lớp nào mà ContentHandler biết cách cung cấp, thì nó sẽ trả về null.

Để gọi phương thức này, ứng dụng khách gọi phương thức với các đối số giống nhau trong đối tượng URL hoặc URLConnection. Nó chuyển một mảng các đối tượng Class theo thứ tự nó muốn nhận dữ liệu. Do đó, nếu nó thích nhận một Chuỗi nhưng sẵn sàng chấp nhận một InputStream và sẽ coi một Vectơ làm phương sách cuối cùng, thì nó sẽ đặt String.classin là thành phần thứ 0 của mảng, InputStream.class trong thành phần đầu tiên của mảng và Vector.class trong thành phần cuối cùng của mảng. Sau đó, nó sẽ sử dụng instanceof để kiểm tra những gì thực sự được trả về và xử lý nó hoặc chuyển đổi nó thành kiểu ưa thích. Ví dụ:

Class[] requestedTypes = {String.class, InputStream.class, Vector.class};

Object content = url.getContent(requestedTypes); if (content instanceof String) {

String s = (String) content; System.out.println(s);

}

else if (content instanceof InputStream) { InputStream in = (InputStream) content; int c;

while ((c = in.read( )) != -1) System.out.write(c);

}

else if (content instanceof Vector) { Vector v = (Vector) content;

for (Enumeration e = v.elements() ; e.hasMoreElements( ) ;) { String[] sa = (String[]) e.nextElement( );

for (int i = 0; i < sa.length; i++) { System.out.print(sa[i] + "\t");

}

System.out.println( );

}

}

else {

System.out.println("Unrecognized content type " + content.getClass( ));

}

Để chứng minh điều này, hãy viết một trình xử lý nội dung có thể được sử dụng cùng với giao thức thời gian. Nhớ lại rằng giao thức thời gian trả về thời gian hiện tại tại máy chủ như một số nguyên không dấu 4 byte, big-endian cho số giây kể từ nửa đêm,ngày 1 tháng 1 năm 1900, giờ chuẩn Greenwich. Có một số ứng cử viên rõ ràng cho lưu trữ dữ liệu này trong trình xử lý nội dung Java, bao gồm java.lang.Long (java.lang.Integer sẽ không hoạt động vì giá trị không dấu có thể vượt quá giới hạn của một int), java.util.Date, java.util.Calendar, java.lang.String và java.io.InputStream, thường hoạt động như một phương sách cuối cùng. Ví dụ 17.3 cung cấp tất cả năm lựa chọn. Vì không có loại MIME tiêu chuẩn cho định dạng thời gian, chúng tôi sẽ sử dụng ứng dụng cho loại để chỉ ra rằng đây là dữ liệu nhị phân và x-time cho subtype để chỉ ra rằng đây là một loại tiện ích mở rộng không chuẩn. Nó sẽ được tùy thuộc vào thời gian trình xử lý giao thức để trả về loại nội dung phù hợp.

VD. 17.3. Trình xử lý nội dung một thời gian

package com.macfaq.net.www.content.application;

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class x\_time extends ContentHandler {

public Object getContent(URLConnection uc) throws IOException {

Class[] classes = new Class[1];

classes[0] = Date.class;

return this.getContent(uc, classes);

}

public Object getContent(URLConnection uc, Class[] classes) throws IOException {

InputStream in = uc.getInputStream( );

for (int i = 0; i < classes.length; i++) {

if (classes[i] == InputStream.class) {

return in;

}

else if (classes[i] == Long.class) {

long secondsSince1900 = readSecondsSince1900(in);

return new Long(secondsSince1900);

}

else if (classes[i] == Date.class) {

long secondsSince1900 = readSecondsSince1900(in);

Date time = shiftEpochs(secondsSince1900);

return time;

}

else if (classes[i] == Calendar.class) {

long secondsSince1900 = readSecondsSince1900(in);

Date time = shiftEpochs(secondsSince1900);

Calendar c = Calendar.getInstance( );

c.setTime(time);

return c;

}

else if (classes[i] == Date.class) {

long secondsSince1900 = readSecondsSince1900(in);

Date time = shiftEpochs(secondsSince1900);

return time.toString( );

}

}

return null; // no requested type available

}

private long readSecondsSince1900(InputStream in) throws IOException

{

long secondsSince1900 = 0;

for (int j = 0; j < 4; j++) {

secondsSince1900 = (secondsSince1900 << 8) | in.read( );

}

return secondsSince1900;

}

private Date shiftEpochs(long secondsSince1900) {

// The time protocol sets the epoch at 1900, the Java Date class

// at 1970. This number converts between them. long differenceBetweenEpochs = 2208988800L;

long secondsSince1970 = secondsSince1900 - differenceBetweenEpochs; long msSince1970 = secondsSince1970 \* 1000;

Date time = new Date(msSince1970);

return time;

}

}

Hầu hết công việc được thực hiện bởi phương thức getContent () thứ hai. Điều này kiểm tra xem liệu nó có nhận ra bất kỳ lớp nào trong mảng các lớp hay không. Nếu vậy, nó cố gắng chuyển đổi nội dung thành một đối tượng của loại đó. Vòng lặp for được sắp xếp để các lớp trước đó trong mảng được ưu tiên; nghĩa là, trước tiên, chúng tôi cố gắng so khớp lớp đầu tiên trong mảng; tiếp theo, chúng tôi cố gắng so khớp lớp thứ hai trong mảng; sau đó là lớp thứ ba trong mảng; và như thế. Ngay sau khi một lớp được so khớp, phương thức này sẽ trả về các lớp sau sẽ không được so khớp ngay cả khi chúng là một lựa chọn được phép. Khi một loại được so khớp, một thuật toán đơn giản sẽ chuyển đổi bốn byte mà thời gian máy chủ gửi vào đúng loại đối tượng, InputStream, Long, Date, Calendar, hoặc một chuỗi. Việc chuyển đổi InputStream là không đáng kể. Chuyển đổi dài là một trong những lần hiếm hoi khi có vẻ hơi bất tiện khi các kiểu dữ liệu nguyên thủy không phải là đối tượng. Mặc dù bạn có thể chuyển đổi thành và trả về bất kỳ loại đối tượng nào, nhưng bạn không thể chuyển đổi sang và trả về một kiểu dữ liệu nguyên thủy như long, vì vậy chúng tôi trả về trình bao bọc kiểu lớp Long thay. Chuyển đổi Ngày và Lịch yêu cầu thay đổi nguồn gốc của thời gian từ ngày 1 tháng 1 năm 1900 đến ngày 1 tháng 1 năm 1970 và đổi đơn vị tính từ giây sang mili giây như đã thảo luận trong Chương 10. Cuối cùng, việc chuyển đổi thành Chuỗi đơn giản chuyển đổi thành Ngày trước, sau đó gọi phương thức toString () của đối tượng Ngày.

Mặc dù có thể định cấu hình máy chủ web để gửi dữ liệu kiểu MIME application / x-time, lớp này thực sự được thiết kế để sử dụng bởi một giao thức tùy chỉnh người xử lý. Trình xử lý này không chỉ biết cách nói giao thức thời gian mà còn biết cách trả về ứng dụng / x-time từ phương thức getContentType (). Thí dụ 17.4 và Ví dụ 17.5 chứng minh một trình xử lý giao thức như vậy. Nó giả định rằng thời gian

URL trông giống như thời gian: //vision.poly.edu: 3737 /.

VD: 17.4. Kết nối URL cho Trình xử lý Giao thức Thời gian

package com.macfaq.net.www.protocol.time;

import java.net.\*;

import java.io.\*;

import com.macfaq.net.www.content.application.\*;

public class TimeURLConnection extends URLConnection {

private Socket connection = null;

public final static int DEFAULT\_PORT = 37;

public TimeURLConnection (URL u) {

super(u);

}

public String getContentType( ) {

return "application/x-time";

}

public Object getContent( ) throws IOException {

ContentHandler ch = new x\_time( ); return ch.getContent(this); } public Object getContent(Class[] classes) throws IOException { ContentHandler ch = new x\_time( );

return ch.getContent(this, classes);

}

public InputStream getInputStream( ) throws IOException {

if (!connected) this.connect( );

return this.connection.getInputStream( );

}

public synchronized void connect( ) throws IOException {

if (!connected) {

int port = url.getPort( );

if ( port < 0) {

port = DEFAULT\_PORT;

}

this.connection = new Socket(url.getHost( ), port);

this.connected = true;

}

}

}

Nói chung, cần đủ để trình xử lý giao thức chỉ cần biết hoặc có thể suy ra loại nội dung MIME chính xác. Tuy nhiên, trong trường hợp như thế này, cả hai các trình xử lý nội dung và giao thức phải được cung cấp, bạn có thể ràng buộc chúng chặt chẽ hơn một chút cùng nhau bằng cách ghi đè getContent (). Điều này cho phép bạn tránh làm lộn xộn với thuộc tính java.content.handler.pkgs hoặc cài đặt ContentHandlerFactory. Bạn vẫn sẽ cần thiết lập thuộc tính java.protocolhandler.pkgs để trỏ đến gói của bạn hoặc cài đặt một URLStreamHandlerFactory, tuy nhiên. Ví dụ 17.5 là một URLStreamHandler đơn giản cho trình xử lý giao thức thời gian.

VD:17,5. URLStreamHandler cho Trình xử lý giao thức thời gian

package com.macfaq.net.www.protocol.time;

import java.net.\*;

import java.io.\*;

public class Handler extends URLStreamHandler {

protected URLConnection openConnection(URL u) throws IOException {

return new TimeURLConnection(u);

}

}

Chúng ta có thể cài đặt trình xử lý giao thức thời gian vào HotJava, như trong chương trước. Tuy nhiên, ngay cả khi chúng ta đặt trình xử lý nội dung thời gian trong đường dẫn lớp của HotJava, HotJava sẽ không sử dụng nó. Do đó, tôi đã viết một ứng dụng độc lập đơn giản, được hiển thị trong

Ví dụ 17.6, sử dụng các giao thức và trình xử lý nội dung này để cho biết thời gian. Lưu ý rằng nó không cần nhập hoặc tham chiếu trực tiếp đến bất kỳ lớp nào có liên quan. Nó chỉ cần cho phép URL tìm thấy trình xử lý nội dung phù hợp.

VD: 17.6. URLTimeClient

import java.net.\*;

import java.util.\*;

import java.io.\*;

public class URLTimeClient {

public static void main(String[] args) {

System.setProperty("java.protocol.handler.pkgs", "com.macfaq.net.www.protocol");

try {

// You can replace this with your own time server URL u = new URL("time://tock.usno.navy.mil/");

Class[] types = {String.class, Date.class, Calendar.class, Long.class};

Object o = u.getContent(types);

System.out.println(o);

}

catch (IOException e) {

// Let's see what went wrong e.printStackTrace( );

}

}

}

Đây là bài chạy mẫu:

D:\JAVA\JNP2\examples\16>java URLTimeClient

Thu Nov 18 08:27:31 PST 1999

Trong trường hợp này, một đối tượng Chuỗi đã được trả về. Đây là sự lựa chọn đầu tiên của URLTimeClient nhưng là lựa chọn cuối cùng của trình xử lý nội dung. Sự lựa chọn của khách hàng luôn được ưu tiên.

### 17.3 Giao diện ContentHandlerFactory

Một ContentHandlerFactory xác định các quy tắc cho vị trí của các lớp ContentHandler được lưu trữ. Tạo một lớp triển khai ContentHandlerFactory và cung cấp cho lớp này

Phương thức createContentHandler () biết cách khởi tạo Trình xử lý nội dung. Phương thức createContentHandler () sẽ trả về null nếu nó không thể tìm thấy ContentHandler thích hợp cho kiểu MIME; null báo hiệu Java để xem xét cho các lớp ContentHandler ở các vị trí mặc định. Khi ứng dụng của bạn bắt đầu, gọi phương thức setContentHandlerFactory () của URLConnection để đặt ContentHandlerFactory. Phương thức này chỉ có thể được gọi một lần trong thời gian tồn tại của ứng dụng.

### 17.3.1 Phương thức createContentHandler ()

Cũng giống như phương thức createURLStreamHandler () của URLStreamHandlerFactory giao diện chịu trách nhiệm tìm và tải trình xử lý giao thức thích hợp, vì vậy cũng như phương thức createContentHandler () của ContentHandlerFactory giao diện chịu trách nhiệm tìm và tải ContentHandler thích hợp đã cho một kiểu MIME:

public abstract ContentHandler createContentHandler(String mimeType)

Phương thức này chỉ nên được gọi bởi phương thức getContent () của một

Đối tượng URLConnection. Ví dụ: Ví dụ 17.7 là một ContentHandlerFactory

biết cách tìm trình xử lý phù hợp cho nội dung text/tab-separated-values xử lý của Ví dụ 17.7:

Ví dụ 17.7. TabFactory

package com.macfaq.net.www.content;

import java.net.\*;

public class TabFactory implements ContentHandlerFactory {

public ContentHandler createContentHandler(String mimeType)) {

if (mimeType.equals("text/tab-separated-values") {

return new com.macfaq.net.www.content.text.tab\_separated\_values( );

}

else {

return null;

// look for the handler in the default locations

}

}

}

Nhà máy này chỉ biết cách tìm một loại trình xử lý nội dung, nhưng không có giới hạn đến bao nhiêu một nhà máy có thể biết về. Ví dụ, this createContentHandler () phương pháp cũng đề xuất các trình xử lý cho application/x-time, text/plain, video/mpeg, và model/vrml. Lưu ý rằng khi bạn đang sử dụng ContentHandlerFactory, bạn không nhất thiết phải tuân theo các quy ước đặt tên tiêu chuẩn cho ContentHandler các lớp con:

public ContentHandler createContentHandler(String mimeType)) {

if (mimeType.equals("text/tab-separated-values") {

return new com.macfaq.net.www.content.text.tab\_separated\_values( );

}

else if (mimeType.equals("application/x-time") {

return new com.macfaq.net.www.content.application.x\_time( );

}

else if (mimeType.equals("text/plain") {

return new sun.net.www.content.text.plain( ); }

if (mimeType.equals("video/mpeg") {

return new com.macfaq.video.MPEGHandler( );

}

if (mimeType.equals("model/vrml") {

return new com.macfaq.threed.VRMLModel( );

}

else {

return null;

// look for the handler in the default locations

}

}

### 17.3.2 Cài đặt các nhà máy xử lý nội dung

ContentHandlerFactory được cài đặt trong một ứng dụng bằng cách sử dụng tĩnh Phương thức URLConnection.setContentHandlerFactory ():

public static void setContentHandlerFactory(ContentHandlerFactory fac) throws SecurityException, Error

Lưu ý rằng phương thức này nằm trong lớp URLConnection, không phải lớp ContentHandler. Nó có thể được gọi nhiều nhất một lần trong bất kỳ lần chạy ứng dụng nào. Nó ném ra một Lỗi nếu nó được gọi là lần thứ hai.

Sử dụng ContentHandlerFactory chẳng hạn như TabFactory trong Ví dụ 17.5, nó có thể viết một ứng dụng độc lập có thể tự động tải trình xử lý nội dung giá trị được phân tách bằng tab của chúng tôi và chạy trong Java 1.1 đến 1.3 mà không có bất kỳ phức tạp lớn với đường dẫn lớp. Ví dụ 17.8 là một chương trình như vậy. Tuy nhiên, như với hầu hết các phương thức setFactory () khác, các applet không đáng tin cậy nói chung sẽ không được phép để thiết lập nhà máy xử lý nội dung. Cố gắng làm như vậy sẽ tạo ra một Ngoại lệ an ninh. Do đó, việc cài đặt các trình xử lý nội dung mới trong các applet khá nhiều yêu cầu truy cập trực tiếp vào phương thức getContent () của ContentHandler lớp con của chính nó. Tốt nhất, điều này không cần thiết, nhưng cho đến khi Sun cung cấp tốt hơn hỗ trợ cho các trình xử lý nội dung có thể tải xuống trong các trình duyệt, đây là điều chúng tôi gặp khó khăn.

Ví dụ 17.8. TabLoader sử dụng ContentHandlerFactory

import java.io.\*;

import java.net.\*;

import java.util.\*;

import com.macfaq.net.www.content.\*;

public class TabLoader {

public static void main (String[] args) {

URLConnection.setContentHandlerFactory(new TabFactory( ));

for (int i = 0; i < args.length; i++) {

try {

URL u = new URL(args[i]);

Object content = u.getContent( );

Vector v = (Vector) content;

for (Enumeration e = v.elements() ;e.hasMoreElements( ) ;)

{

String[] sa = (String[]) e.nextElement( );

for (int j = 0; j < sa.length; j++) {

System.out.print(sa[j] + "\t"); } System.out.println( );

}

}

catch (MalformedURLException e) {

System.err.println(args[i] + " is not a good URL");

}

catch (Exception e) {

e.printStackTrace( );

}

}

}

}

Đây là một lần chạy điển hình. Như thường lệ, các tab được biểu thị bằng các mũi tên:

java TabLoader http://metalab.unc.edu/javafaq/addresses.tab

JPE Associates

-> 341 Lafayette St, Suite 1025

-> New York

-> NY

->10012 O'Reilly & Associates

-> 103 Morris St, Suite A

->Sebastopol

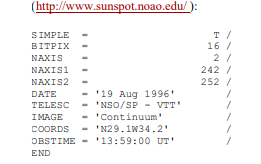
-> CA

-> 95472

### 17.4 Trình xử lý nội dung cho định dạng hình ảnh: image / x-fit

Đó thực sự là tất cả những gì cần có đối với trình xử lý nội dung. Như một ví dụ cuối cùng, tôi sẽ chỉ cho bạn cáchđể viết một trình xử lý nội dung cho các tệp hình ảnh. Những điều này khác với nội dung dựa trên văn bản các trình xử lý mà bạn đã thấy ở đó chúng thường tạo ra một đối tượng triển khai giao diện java.awt.ImageProductioner chứ không phải là một đối tượng Dòng đầu vào. Cácví dụ cụ thể mà chúng tôi sẽ chọn là định dạng Hệ thống truyền tải hình ảnh linh hoạt (FITS) được sử dụng phổ biến giữa các nhà thiên văn học. Các tệp FITS có thang độ xám, hình ảnh được ánh xạ bit với tiêu đề xác định độ sâu bit của hình ảnh, chiều rộng và chiều cao của hình ảnh và số lượng hình ảnh trong tệp. Mặc dù các tệp FITS thường chứa một số hình ảnh (thường là những hình ảnh của cùng một thứ được chụp vào những thời điểm khác nhau), trong ví dụ, chúng tôi chỉ xem hình ảnh đầu tiên trong một tệp.

Có một số điều chính bạn cần biết để xử lý tệp FITS. Đầu tiên, các tệp FITSđược chia thành các khối chính xác 2,880 byte. Nếu không có đủ dữ liệu để điền vào khối, nó được đệm bằng khoảng trắng ở cuối. Mỗi tệp FITS có hai phần, tiêu đề và đơn vị dữ liệu chính. Tiêu đề chiếm một số khối tích hợp, cũng như đơn vị dữ liệu chính. Nếu tệp FITS chứa phần mở rộng, có thể có thêm dữ liệusau đơn vị dữ liệu chính, nhưng chúng tôi bỏ qua điều đó ở đây. Bất kỳ phần mở rộng nào hiện cósẽ không thay đổi hình ảnh có trong đơn vị dữ liệu chính. Tiêu đề bắt đầu trong khối đầu tiên của tệp FITS. Nó có thể chiếm một hoặc nhiều các khối; khối cuối cùng có thể được đệm bằng khoảng trắng ở cuối. Tiêu đề là văn bản ASCII. Mỗi dòng của tiêu đề rộng đúng 80 byte. Tám ký tự đầu tiên của mỗi ký tự dòng tiêu đề chứa một từ khóa, theo sau là một dấu bằng (ký tự 9), tiếp theo là dấu cách (10). Từ khóa được đệm ở bên phải với khoảng trắng để tạo từ khóa dài tám ký tự. Các cột từ 11 đến 30 chứa một giá trị; giá trị có thể được căn bên phải và được đệm ở bên trái bằng khoảng trống nếu cần. Giá trị có thể là một số nguyên, một số dấu phẩy động, T hoặc F biểu thị các giá trị boolean đúng và sai, hoặc chuỗi được phân cách bằng dấu ngoặc kép. Một chú thích có thể xuất hiện trong các cột từ 31 đến 80; nhận xét được phân tách khỏi giá trị của một trường bằng dấu gạch chéo (/). Đây là một tiêu đề đơn giản lấy từ hình ảnh FITS do K. S. Balasubramaniam sản xuất tại Tháp chân không Kính viễn vọng tại Đài quan sát Mặt trời Quốc gia ở Vết đen Mặt trời, New Mexico



Mọi tệp FITS đều bắt đầu bằng từ khóa SIMPLE. Từ khóa này luôn có giá trị T. Nếu không đúng như vậy, tệp không hợp lệ. Dòng thứ hai của tệp FITS luôn có từ khóa BITPIX, cho bạn biết cách dữ liệu được lưu trữ. Có năm giá trị có thể có cho BITPIX, bốn trong số đó tương ứng chính xác với dữ liệu nguyên thủy của Java các loại. Giá trị phổ biến nhất của BITPIX là 16, nghĩa là có 16 bit cho mỗi pixel, tương đương với một đoạn ngắn Java. BITPIX 32 là một int Java. A BITPIX của -32 có nghĩa là mỗi pixel được biểu thị bằng một số dấu phẩy động 32 bit (tương đương với một float Java); BITPIX của -64 tương đương với một đôi Java. Một BITPIX của 8 có nghĩa là 8 bit được sử dụng để đại diện cho mỗi pixel; điều này tương tự như một Javabyte, ngoại trừ FITS sử dụng các byte không dấu từ đến 255; Dữ liệu byte của Javakiểu được ký, nhận các giá trị nằm trong khoảng từ -128 đến 127. Các từ khóa còn lại trong tệp FITS có thể xuất hiện theo bất kỳ thứ tự nào. Họ không phải nhất thiết phải theo thứ tự hiển thị ở đây. Trong trình xử lý nội dung FITS của chúng tôi, trước tiên chúng tôi đọc tất cả từ khóa vào Hashtable và sau đó trích xuất những từ khóa chúng ta muốn theo tên.

Tiêu đề NAXIS chỉ định số lượng trục (nghĩa là kích thước) của mảng dữ liệu chính. Giá trị NAXIS của một trong số đó xác định hình ảnh một chiều. Một giá trị NAXIS của hai cho biết một hình ảnh hình chữ nhật hai chiều bình thường. Một Giá trị NAXIS của ba được gọi là khối dữ liệu và thường có nghĩa là tệp chứa loạt ảnh của cùng một đối tượng được chụp tại các thời điểm khác nhau. Trong khác từ, thời gian là chiều thứ ba. Trong những trường hợp hiếm hoi, chiều thứ ba có thể đại diện cho độ sâu: tức là tệp chứa hình ảnh ba chiều thực. NAXIS của bốn nghĩa là tệp chứa một chuỗi các hình ảnh ba chiều được chụp tạinhững khoảnh khắc khác nhau trong thời gian. Các giá trị cao hơn của NAXIS, trong khi về mặt lý thuyết, là hiếm thấy trong thực tế. Ví dụ của chúng tôi sẽ chỉ xem xét hai chiều đầu tiên hình ảnh trong một tệp. Các tiêu đề NAXISn (trong đó n là một số nguyên nằm trong khoảng từ 1 đến NAXIS) cho chiều dài của hình ảnh tính bằng pixel dọc theo kích thước đó. Trong ví dụ này, NAXIS1 là 242, vì vậy hình ảnh có chiều rộng 242 pixel. NAXIS2 là 252, vì vậy hình ảnh này cao 252 pixel. Vì hình ảnh FITS thường là hình ảnh của các thiên thể thiên văn như mặt trời, nó không thực sự quan trọng nếu bạn đảo ngược chiều rộng và chiều cao. Tất cả các hình ảnh FITS đều chứa Các từ khóa SIMPLE, BITPIX, END và NAXIS, cùng với một loạt các từ khóa NAXISn. Tất cả các từ khóa này đều cung cấp thông tin cần thiết cho việc hiển thị hình ảnh. Năm từ khóa tiếp theo dành riêng cho tệp này và có thể không có trong các FITS khác các tập tin. Chúng mang lại ý nghĩa cho hình ảnh, mặc dù chúng không cần thiết để hiển thị nó. Các Từ khóa DATE cho biết hình ảnh này được chụp vào ngày 19 tháng 8 năm 1996. TELESCtừ khóa cho biết hình ảnh này được chụp bởi Kính viễn vọng Tháp Chân không (VTT) tại Đài quan sát Mặt trời Quốc gia (NSO) trên Đỉnh Sacramento (SP). Từ khóa IMAGE nói rằng đây là hình ảnh của liên tục ánh sáng trắng; hình ảnh được chụp qua máy quang phổ có thể chỉ xem xét một bước sóng cụ thể trong quang phổ. Các Từ khóa COORDS cung cấp vĩ độ và kinh độ của kính thiên văn. cuối cùng Từ khóa OBSTIME cho biết hình ảnh này được chụp lúc 1:59 CH. Giờ quốc tế (về cơ bản, Giờ chuẩn Greenwich). Có nhiều tiêu đề tùy chọn khác không xuất hiện trong ví dụ này. Giống như năm từ khóa được thảo luận ở đây, các từ khóa còn lại có thể giúp ai đó diễn giải hình ảnh, nhưng họ không cung cấp thông tin cần thiết để hiển thị nó.

Từ khóa END kết thúc tiêu đề. Theo sau từ khóa END, tiêu đề là được đệm bằng khoảng trắng để nó lấp đầy một khối 2,880 byte. Một tiêu đề có thể chiếm nhiều hơn một khối 2,880 byte, nhưng nó phải luôn được đệm vào một số khối tích phân. Dữ liệu hình ảnh theo sau tiêu đề. Hình ảnh được lưu trữ như thế nào phụ thuộc vào giá trị của BITPIX, như đã giải thích trước đó. May mắn thay, các kiểu dữ liệu này được lưu trữ trong các định dạng (big endian, hai phần bổ sung) có thể được đọc trực tiếp bằng DataInputStream. Các ý nghĩa chính xác của mỗi số trong dữ liệu hình ảnh hoàn toàn phụ thuộc vào tệp. Hơnthường xuyên hơn không, đó là số lượng electron đã được thu thập trong một thời gian cụ thể khoảng cách bởi một pixel cụ thể trong một thiết bị tích hợp điện tích (CCD); trong các tệp FITS cũ hơn, các con số có thể đại diện cho giá trị đọc được từ phim ảnh bằng máy đo mật độ.

Tuy nhiên, chủ đề thống nhất là những con số lớn hơn thể hiện ánh sáng rực rỡ hơn. Đếdiễn giải những con số này dưới dạng hình ảnh thang độ xám, bạn ánh xạ giá trị nhỏ nhất trong dữ liệu sang màu đen tuyền, giá trị lớn nhất trong dữ liệu thành màu trắng tinh khiết và chia tỷ lệ tất cả trung giancác giá trị một cách thích hợp. Một trình đọc FITS có mục đích chung không thể giải thích các con số là bất cứ thứ gì ngoại trừ mức độ sáng trừu tượng. Nếu không mở rộng quy mô, sự khác biệt có xu hướng nhận được rửa sạch. Ví dụ, một điểm tối trên Mặt trời có xu hướng khoảng 4.000K. Đó là tối so với nhiệt độ bề mặt mặt trời bình thường là 6.000K, nhưng đáng kể sáng hơn bất cứ thứ gì bạn có thể thấy trên bề mặt Trái đất.

Ví dụ 17.9 là một trình xử lý nội dung FITS. Các tệp FITS sẽ được phân phát với MIME gõ image / x-phù hợp. Điều này gần như chắc chắn không được bao gồm trong mặc định của máy chủ của bạn ánh xạ kiểu MIME, vì vậy hãy đảm bảo thêm ánh xạ giữa các tệp kết thúc trong .fit, .fts hoặc .fits và hình ảnh loại MIME / x-fit

Ví dụ 17.9. Trình xử lý nội dung x-fit

package com.macfaq.net.www.content.image; import java.net.\*;

import java.io.\*;

import java.awt.image.\*;

import java.util.\*;

public class x\_fits extends ContentHandler {

public Object getContent(URLConnection uc) throws IOException {

int width = -1;

int height = -1;

int bitpix = 16;

int[] data = null;

int naxis = 2;

Hashtable header = null;

DataInputStream dis = new DataInputStream(uc.getInputStream( ));

header = readHeader(dis);

bitpix = getIntFromHeader("BITPIX ", -1, header);

if (bitpix <= 0) return null;

naxis = getIntFromHeader("NAXIS ", -1, header);

if (naxis < 1) return null;

width = getIntFromHeader("NAXIS1 ", -1, header);

if (width <= 0) return null;

if (naxis == 1) height = 1; else height = getIntFromHeader("NAXIS2 ", -1, header); if (height <= 0) return null;

if (bitpix == 16) {

short[] theInput = new short[height \* width];

for (int i = 0; i < theInput.length; i++) {

theInput[i] = dis.readShort( );

} data = scaleArray(theInput);

}

else if (bitpix == 32) {

int[] theInput = new int[height \* width];

for (int i = 0; i < theInput.length; i++) {

theInput[i] = dis.readInt( );

} data = scaleArray(theInput);

}

else if (bitpix == 64) {

long[] theInput = new long[height \* width];

for (int i = 0; i < theInput.length; i++) {

theInput[i] = dis.readLong( );

}

data = scaleArray(theInput);

}

else if (bitpix == -32) {

float[] theInput = new float[height \* width];

for (int i = 0; i < theInput.length; i++) {

theInput[i] = dis.readFloat( );

}

data = scaleArray(theInput);

}

else if (bitpix == -64) {

double[] theInput = new double[height \* width];

for (int i = 0; i < theInput.length; i++) {

theInput[i] = dis.readDouble( );

}

data = scaleArray(theInput);

}

else { System.err.println("Invalid BITPIX"); return null;

}

// end if-else-if return new MemoryImageSource(width, height, data, 0, width);

} // end getContent

private Hashtable readHeader(DataInputStream dis) throws IOException {

int blocksize = 2880;

int fieldsize = 80;

String key, value;

int linesRead = 0;

byte[] buffer = new byte[fieldsize];

Hashtable header = new Hashtable( );

while (true) {

dis.readFully(buffer);

key = new String(buffer, 0, 8, "ASCII");

linesRead++;

if (key.substring(0, 3).equals("END")) break;

if (buffer[8] != '=' || buffer[9] != ' ') continue;

value = new String(buffer, 10, 20, "ASCII");

header.put(key, value);

}

int linesLeftToRead = (blocksize - ((linesRead \* fieldsize) % blocksize))/fieldsize;

for (int i = 0; i < linesLeftToRead; i++) dis.readFully(buffer);

return header;

}

private int getIntFromHeader(String name, int defaultValue, Hashtable header)

{

String s = "";

int result = defaultValue;

try {

s = (String) header.get(name);

}

catch (NullPointerException e) {

return defaultValue;

}

try {

result = Integer.parseInt(s.trim( ));

}

catch (NumberFormatException e) {

System.err.println(e);

System.err.println(s);

return defaultValue;

}

return result;

}

// parameterized types (templates) would help a lot here private int[] scaleArray(short[] theInput)

{

int data[] = new int[theInput.length];

int max = 0;

int min = 0;

for (int i = 0; i < theInput.length; i++) {

if (theInput[i] > max) max = theInput[i];

if (theInput[i] < min) min = theInput[i];

}

long r = max - min;

double a = 255.0/r;

double b = -a \* min;

int opaque = 255; for (int i = 0; i < data.length; i++) { int temp = (int) (theInput[i] \* a + b);

data[i] = (opaque << 24) | (temp << 16) | (temp << 8) | temp;

}

return data;

}

private int[] scaleArray(int[] theInput) {

int data[] = new int[theInput.length];

int max = 0;

int min = 0;

for (int i = 0; i < theInput.length; i++)

{

if (theInput[i] > max) max = theInput[i];

if (theInput[i] < min) min = theInput[i];

} long r = max - min;

double a = 255.0/r;

double b = -a \* min;

int opaque = 255;

for (int i = 0; i < data.length; i++) {

int temp = (int) (theInput[i] \* a + b);

data[i] = (opaque << 24) | (temp << 16) | (temp << 8) | temp;

}

return data;

}

private int[] scaleArray(long[] theInput) {

int data[] = new int[theInput.length];

long max = 0;

long min = 0;

for (int i = 0; i < theInput.length; i++) {

if (theInput[i] > max) max = theInput[i];

if (theInput[i] < min) min = theInput[i];

}

long r = max - min;

double a = 255.0/r;

double b = -a \* min;

int opaque = 255;

for (int i = 0; i < data.length; i++) {

int temp = (int) (theInput[i] \* a + b);

data[i] = (opaque << 24) | (temp << 16) | (temp << 8) | temp;

}

return data;

}

private int[] scaleArray(double[] theInput) {

int data[] = new int[theInput.length];

double max = 0;

double min = 0;

for (int i = 0; i < theInput.length; i++) {

if (theInput[i] > max) max = theInput[i];

if (theInput[i] < min) min = theInput[i];

}

double r = max - min;

double a = 255.0/r;

double b = -a \* min;

int opaque = 255;

for (int i = 0; i < data.length; i++) {

int temp = (int) (theInput[i] \* a + b);

data[i] = (opaque << 24) | (temp << 16) | (temp << 8) | temp;

}

return data;

}

private int[] scaleArray(float[] theInput) {

int data[] = new int[theInput.length];

float max = 0;

float min = 0;

for (int i = 0; i < theInput.length; i++) {

if (theInput[i] > max) max = theInput[i];

if (theInput[i] < min) min = theInput[i]; } double r = max - min;

double a = 255.0/r;

double b = -a \* min;

int opaque = 255;

for (int i = 0; i < data.length; i++) {

int temp = (int) (theInput[i] \* a + b);

data[i] = (opaque << 24) | (temp << 16) | (temp << 8) | temp;

}

return data;

}

}

Phương thức quan trọng của lớp x\_fits là getContent (); nó là một phương pháp mà Lớp ContentHandler yêu cầu các lớp con thực hiện. Các phương pháp khác trong này lớp chỉ đơn giản là các phương thức tiện ích giúp chia nhỏ chương trình thành các phần dễ hiểu hơn. getContent () được gọi bởi một URLConnection, nó sẽ chuyển một tham chiếu đến chính nó trong đối số uc. Phương thức getContent () đọc dữ liệu từ URLConnection đó và sử dụng nó để xây dựng một đối tượng triển khai

Giao diện ImageProductioner. Để đơn giản hóa nhiệm vụ tạo ImageProductioner, chúng tôitạo một mảng dữ liệu hình ảnh và sử dụng đối tượng MemoryImageSource, triển khai giao diện ImageProductioner, để chuyển đổi mảng đó thành một hình ảnh. getContent () trả về MemoryImageSource này.MemoryImageSource có một số hàm tạo. Cái mà chúng tôi sử dụng ở đây yêu cầu chúng tôi cung cấp chiều rộng và chiều cao của hình ảnh, một mảng các giá trị nguyên có chứa Dữ liệu RGB cho mỗi pixel, độ lệch của thời điểm bắt đầu dữ liệu đó trong mảng và số pixel trên mỗi dòng trong mảng:

public MemoryImageSource(int width, int height, int[] pixels, int offset, int scanlines);

Dữ liệu chiều rộng, chiều cao và pixel có thể được đọc từ tiêu đề của hình ảnh FITS. Vì chúng tôi đang tạo một mảng mới để giữ dữ liệu pixel, độ lệch bằng 0 và đường quét là chiều rộng của hình ảnh.

Trình xử lý nội dung của chúng tôi có một phương thức tiện ích được gọi là readHeader () để đọc hình ảnh tiêu đề từ InputStream của uc. Phương thức này trả về một Hashtable chứa từ khóa và giá trị của chúng dưới dạng đối tượng Chuỗi. Bình luận bị ném đi. readHeader () đọc 80 byte một lúc, vì đó là độ dài của mỗi trường. Người đầu tiên tám byte được chuyển thành khóa Chuỗi. Nếu không có khóa, dòng là bình luận và bị bỏ qua. Nếu có một khóa, thì byte thứ mười một đến thứ ba mươi là được lưu trữ trong một chuỗi được gọi là giá trị. Cặp khóa-giá trị được lưu trữ trong Hashtable. Cái này tiếp tục cho đến khi từ khóa END được phát hiện. Tại thời điểm này, chúng tôi thoát ra khỏi vòng lặp và đọc nhiều dòng nếu cần để kết thúc khối. (Nhớ lại rằng tiêu đề là được đệm bằng dấu cách để tạo thành bội số của 2,880). Cuối cùng, readHeader () trả về tiêu đề Hashtable.

Sau khi tiêu đề đã được đọc vào tiêu đề Hashtable, InputStream bây giờ là trỏ vào byte dữ liệu đầu tiên. Tuy nhiên, trước khi sẵn sàng đọc dữ liệu, chúng tôi phải trích xuất chiều cao, chiều rộng và các bit trên mỗi pixel của đơn vị dữ liệu chính từ đầu trang. Đây là tất cả các giá trị số nguyên, vì vậy để đơn giản hóa mã, chúng tôi sử dụng getIntFromHeader (Tên chuỗi, int defaultValue, tiêu đề Hashtable) phương pháp. Phương thức này lấy các đối số là tên của tiêu đề có giá trị mà chúng ta muốn (ví dụ: BITPIX), một giá trị mặc định cho tiêu đề đó và Hashtable chứa đầu trang. Phương thức này truy xuất giá trị được liên kết với tên chuỗi từ Hashtable và truyền kết quả tới một đối tượng String — chúng tôi biết việc truyền này là an toàn vì chúng tôi chỉ đặt dữ liệu Chuỗi vào Hashtable. Chuỗi này sau đó được chuyển đổi thành int sử dụng Integer.parseInt (s.trim ()); sau đó chúng tôi trả về kết quả int. Nếu một ngoại lệ được ném ra, getIntFromHeader () trả về đối số defaultValue thay thế. Trong trình xử lý nội dung này, chúng tôi sử dụng giá trị cờ không thể (-1) làm giá trị mặc định cho chỉ ra rằng getIntFromHeader () không thành công. getContent () sử dụng getIntFromHeader () để truy xuất bốn giá trị quan trọng từ tiêu đề: NAXIS, NAXIS1, NAXIS2 và BITPIX. NAXIS là số thứ nguyên trong mảng dữ liệu chính; nếu nó lớn hơn hoặc bằng hai, chúng tôi đọc chiều rộng và chiều cao từ NAXIS1 và NAXIS2. Nếu có nhiều hơn hai thứ nguyên, chúng tôi vẫn đọc một khung hai chiều từ dữ liệu. FITS nâng cao hơn trình xử lý nội dung có thể đọc các khung tiếp theo và đưa chúng vào bên dưới bản gốc hình ảnh, hoặc hiển thị chuỗi hình ảnh dưới dạng hoạt ảnh. Nếu NAXIS là một, chiều rộng được đọc từ NAXIS1 và chiều cao được đặt thành một. [3] Nếu NAXIS nhỏ hơn một, có không có dữ liệu hình ảnh nào cả, vì vậy chúng tôi trả về null.

Bây giờ chúng tôi đã sẵn sàng để đọc dữ liệu hình ảnh. Dữ liệu có thể được lưu trữ ở một trong năm các định dạng, tùy thuộc vào giá trị của BITPIX: byte không dấu, short, ints, float hoặc ăng gấp đôi. Đây là nơi mà việc thiếu các kiểu và mẫu được tham số hóa trong Java khiến mã hóa khó khăn: chúng ta cần lặp lại thuật toán đọc dữ liệu năm lần, một lần cho mỗi loại trong số năm kiểu dữ liệu có thể có. Trong mỗi trường hợp, dữ liệu được đọc trước tiên từ luồng vào một mảng của kiểu thích hợp được gọi là Đầu vào. Sau đó, mảng này được chuyển cho Phương thức scaleArray (), trả về một mảng được chia tỷ lệ. scaleArray () là một phương thức nạp chồng để đọc dữ liệu trong Đầu vào và sao chép dữ liệu vào int mảng dữ liệu, trong khi chia tỷ lệ dữ liệu để giảm từ 255; có một phiên bản khác của scaleArray () cho mỗi loại trong số năm kiểu dữ liệu mà chúng ta có thể cần xử lý.

Dữ liệu này bây giờ cần được chuyển đổi thành các giá trị RGB theo thang độ xám. 32-bit tiêu chuẩnMô hình màu RGB cho phép 256 sắc thái xám khác nhau, từ đen tuyền đến tinh khiết trắng; 8 bit được sử dụng để biểu thị độ mờ, thường được gọi là "alpha". Để có được một cụ thể màu xám, các byte màu đỏ, xanh lục và xanh lam của bộ ba RGB đều phải được đặt thành cùng một giá trị và giá trị alpha phải là 255 (hoàn toàn không rõ ràng). Nghĩ về những điều này như bốn giá trị byte, bạn cần các màu như 255.127.127.127 (xám trung bình) hoặc 255.255.255.255 (trắng tinh). Điều này được sản xuất bởi các dây chuyền:

int temp = (int) (theInput[i] \* a + b);

theData[i] = (opaque << 24) | (temp << 16) | (temp << 8) | temp;

Khi nó đã chuyển đổi mọi pixel trongInput [] thành giá trị màu 32 bit và được lưu trữ kết quả trong theData [], scaleArray () trả về theData. Điều duy nhất còn lại cho getContent () cần làm là cung cấp cho mảng này, cùng với các giá trị tiêu đề trước đó được truy xuất, vào phương thức khởi tạo MemoryImageSource và trả về kết quả.

Trình xử lý nội dung FITS này có một vấn đề khó hiểu. Hình ảnh phải hoàn toàn được tải trước khi phương thức trả về. Vì hình ảnh FITS khá giống thiên văn về kích thước, tải hình ảnh có thể mất một lượng thời gian đáng kể. Sẽ tốt hơn nếu

tạo một lớp mới cho các hình ảnh FITS triển khai giao diện ImageProductioner và trong đó dữ liệu có thể được truyền trực tuyến không đồng bộ. ImageConsumer mà cuối cùng hiển thị hình ảnh có thể sử dụng các phương pháp của ImageProductioner để xác định khi chiều cao và chiều rộng khả dụng, khi dòng quét mới đã được đọc, khi hình ảnh được tải hoàn toàn hoặc bị lỗi, v.v. getContent () sẽ sinh ra một luồng riêng biệt để cung cấp dữ liệu vào ImageProductioner và sẽ trả về gần như ngay lập tức. Tuy nhiên, một FITS ImageProductioner sẽ không thể nhận được nhiều lợi thế của việc tải liên tục, vì định dạng tệp không xác định rõ ràng ý nghĩa của mỗi giá trị dữ liệu; trước khi bạn có thể tạo các pixel RGB, bạn phải đọc tất cả dữ liệu và tìm các giá trị nhỏ nhất và lớn nhất.

Ví dụ 17.10 là một ContentHandlerFactory đơn giản nhận dạng các hình ảnh FITS. Đối với tất cả các loại khác với image / x-fit, nó trả về null để các vị trí mặc định sẽ là đã tìm kiếm trình xử lý nội dung

Ví dụ 17.10. FITS ContentHandlerFactory

import java.net.\*;

public class FitsFactory implements ContentHandlerFactory {

public ContentHandler createContentHandler(String mimeType) {

if (mimeType.equalsIgnoreCase("image/x-fits")) {

return new com.macfaq.net.www.content.image.x\_fits( );

}

return null;

}

}

Ví dụ 17.11 là một chương trình đơn giản kiểm tra trình xử lý nội dung này bằng cách tải và hiển thị hình ảnh FITS từ một URL. Trên thực tế, nó có thể hiển thị bất kỳ loại hình ảnh nào một trình xử lý nội dung được cài đặt. Tuy nhiên, nó sử dụng FitsFactory để nhận ra FITS hình ảnh.

Ví dụ 17.11. Trình xem FITS

import java.awt.\*;

import javax.swing.\*;

import java.awt.image.\*;

import java.net.\*;

import java.io.\*;

public class FitsViewer extends JFrame {

private URL url;

private Image theImage;

public FitsViewer(URL u) {

super(u.getFile( ));

this.url = u;

}

public void loadImage( ) throws IOException {

Object content = this.url.getContent( );

ImageProducer producer;

try {

producer = (ImageProducer) content;

}

catch (ClassCastException e) {

throw new IOException("Unexpected type " + content.getClass( ));

}

if (producer == null) theImage = null;

else {

theImage = this.createImage(producer);

int width = theImage.getWidth(this);

int height = theImage.getHeight(this);

if (width > 0 && height > 0) this.setSize(width, height);

}

}

public void paint(Graphics g) {

if (theImage != null) g.drawImage(theImage, 0, 0, this);

}

public static void main(String[] args) {

URLConnection.setContentHandlerFactory(new FitsFactory( ));

for (int i = 0; i < args.length; i++) {

try {

FitsViewer f = new FitsViewer(new URL(args[i]));

f.setSize(252, 252);

f.loadImage( );

f.show( );

}

catch (MalformedURLException e) {

System.err.println(args[i] + " is not a URL I recognize.");

}

catch (IOException e) {

e.printStackTrace( );

}

}

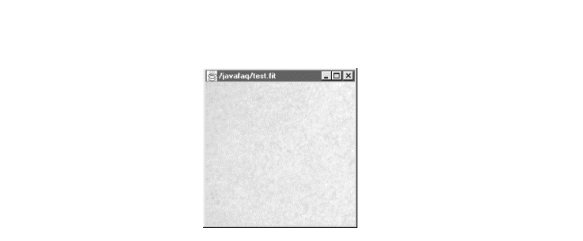
}

}

Chương trình FitsViewer mở rộng JFrame. Phương thức main () lặp qua tất cả các các đối số dòng lệnh tạo một cửa sổ mới cho mỗi đối số. Sau đó, nó tải vào cửa sổ và hiển thị nó. Phương thức loadImage () thực sự tải xuống hình ảnh được yêu cầu bằng cách sử dụng ngầm định trình xử lý nội dung của Ví dụ 17.9 để chuyển đổi dữ liệu FITS thành một đối tượng java.awt.Image được lưu trữ trong trường theImage. Nếu như chiều rộng và chiều cao của hình ảnh có sẵn (vì chúng sẽ dành cho hình ảnh FITS sử dụng trình xử lý nội dung của chúng tôi nhưng có thể không cho một số loại hình ảnh khác tải hình ảnh trong một chuỗi riêng biệt), sau đó cửa sổ được thay đổi kích thước thành kích thước chính xác của hình ảnh.

Phương thức paint () chỉ đơn giản là vẽ hình ảnh này trên màn hình. Hầu hết công việc là được thực hiện bên trong trình xử lý nội dung. Trên thực tế, chương trình này thực sự có thể hiển thị hình ảnh của bất kỳ loại nào mà trình xử lý nội dung được cài đặt và có sẵn. Ví dụ, nó hoạt động tốt như nhau đối với ảnh GIF và JPEG. Hình 17.2 cho thấy chương trình này hiển thị một hình ảnh một phần của quá trình tạo hạt năng lượng mặt trời.

Hình 17.2. Ứng dụng FitsViewer hiển thị hình ảnh FITS về quá trình tạo hạt năng lượng mặt trời



# Chương 3.

## Dịch vụ lập trình công cụ crawl dữ liệu website tự động.

### 1.Crawl dữ liệu là gì.

Crawl dữ liệu hay còn gọi là cào dữ liệu là một thuật ngữ không có gì xa lạ trong ngành marketing. Vì crawl là một kĩ thuật mà các robots của các công cụ tìm kiếm phổ biến hiện nay sử dụng như Google, Yahoo, Bing, Yandex… Crawl có công việc chính là thu thập dữ liệu từ một trang web bất kì, hoặc chỉ định trước rồi phân tích cú pháp mã nguồn HTML để đọc dữ liệu và bóc tách thông tin dữ liệu theo yêu cầu mà người dùng đặt ra các dữ liệu mà Search Engine yêu cầu.

Vậy việc bạn cần Crawl dữ liệu của một hoặc nhiều website khác cũng tương tự như cách mà Google hay làm. Crawl và sao đó Indexing dữ liệu cào được vào dữ liệu của Google sau cùng là phục vụ công việc tìm kiếm của chúng ta.

Vậy tại sao bạn cần công cụ Crawl nhỉ nghe như là một điều to lớn cho những ông lớn sử dụng chứ bạn không cần dùng đúng không? Sai bạn ạ. Thời đại hiện tại là thời đại của công nghệ 4.0 và Big-data cho nên bạn càng làm chủ DATA bao nhiêu thì bạn càng có nhiều co hội trong việc thương mại của doanh nghiệp của bạn hoặc cá nhân bạn vào một ứng dụng nào đó.

### 2.Crawl phù hợp với những doanh nghiệp nào.

- Sàn TMĐT, Website rao vặt.

- Tin tức hàng ngày.

- Pháp luật đời sống.

- Website vệ tinh – PBN.

- Website bán hàng online, nhập hàng nước ngoài.

### 3.Lợi ích của việc crawl data là gì?

Crawl data là giảm tải công việc sáng tạo cho nhân viên của bạn, nhân sự là một bài toán vô cùng quan trọng của một doanh nghiệp đang khởi nghiệp online. Bạn nghĩ sao khi vào một website mà website chỉ có vài sản phẩm hoặc một web đọc tin tức mà chỉ có vài tin ít ỏi?.

Bạn sẽ thoát và tìm một trang web giàu nội dung hơn đúng không? Chắc chắn rồi vì ta chẳng có gì để xem ở một website rỗng cả. Bạn không đủ tài chính để thuê một đội nhập liệu vài trăm nhân sự? Quá cồng kềnh và tốn nhiều chi phí và thủ tục pháp lý đi kèm cho nhân sự không hề đơn giản.

Nhưng ngược lại nếu bạn đầu tư một phần mềm crawler data tự động thì bạn có thể giảm tải gần như 90% nhân sự content hiện tại, chỉ giữ 10% nhân sự để chỉnh sửa, viết lách các nội dung quan trọng cho công ty và quản trị các công cụ crawler data mà thôi.

Ngoài ra bạn có thể sử dụng data vào nhiều mục đích khác nhau, như phân tích thị trường, thiết kế website cổng thông tin, thiết kế website rao vặt,...đều được cả, và phù hợp nhất vẫn là thiết kế website bất động sản hoặc mua giới Bđs vì ngành nghề này rất khát thông tin.

Crawler data sẽ giúp website của bạn có nhiều nội dung hơn, nhiều tin tức hơn,... Và sẽ có nhiều khách hàng hơn.

Bật mí bí mật:

Các công ty chuyên bán hàng bằng Affiliate tìm việc cần một công cụ crawl link, crawl data là vô cùng quan trọng, bạn chỉ cần crawler hết data của các sản phẩm ở website khác, sau đó gắn Link? Ref-Code (Refer) để có thể tăng doanh số của mình một cách chóng mặt.

### 4.Công nghệ sử dụng là gì?

Lptech sử dụng các công cụ mới nhất hiện nay để crawl và bóc tách dữ liệu một cách chính xác và thông minh. Các ngôn ngữ lập trình crawler tốt nhất hiện nay như:

1.Python

2.PHP

3. Node

Proxy trong crawl là điều vô cùng quan trọng chống các website Victim chặn việc thu thập của chúng ta, ngoài ra còn có các kỹ thuật khác sử dụng để phân tích các website cao cấp và có cấu trúc thay đổi liên tục như Zalo shop, Tiki, Sendo, Chotot, Muaban....

### 5.Có bị Google phạt không?

Về việc crawl data có bị phạt không thì cũng là một vấn đề gặp phải của các công ty phần mềm cung cấp dịch vụ này. Theo nguyên tắc thì việc crawl dữ liệu sẽ chia làm hai khía cạnh như sau:

- Đối với Google:

Việc copy hay crawl là sẽ tạo ra một bản sao chép website đó về database của bạn nếu bạn chỉ crawler 100% nội dung thì có thể bạn sẽ vi phạm chính sách nội dung của Google và DMCA sẽ khởi kiện bạn.

Hãy lưu ý việc này nếu bạn đang crawl hay copy bằng tay website, bài viết của một ai đó thì hãy dừng lại ngay vì bạn sẽ bị thuật toán của Google chặn sớm thôi . Hãy sử dụng ảnh công cụ đủ thông minh tái biên soạn lại nội dung của bạn.

- Đối với pháp luật Việt Nam:

Việt Nam có luật bản quyền tác giả được công bố tại nghị định 22/2018/NĐ-CP quy định chi tiết Luật sở hữu trí tuệ, luật sửa đổi luật sở hữu trí tuệ về quyền tác giả, quyền liên quan.

Ngoài ra khi bạn là công ty hay cá nhân có nhu cầu crawl dữ liệu thì trước tiên bạn sẽ được trực kỹ ý một thỏa thuận về bảo mật thông tin để đảm bảo quyền lợi cho đôi bên.

Do đó việc copy dữ liệu của một website hay một báo điện tử là vi phạm pháp luật ở Việt Nam nếu không được đơn vị chủ quyền cho phép. Vì thế việc crawler dữ liệu phải cẩn trọng trong việc này để tránh vi phạm pháp luật ở Việt Nam.

Nếu bạn crawl dữ liệu các website rao vặt thì thường các website này cho phép việc crawler nên bạn không phải lo nhiều về vấn đề trên.

### 6.Kết luận:

Việc crawl dữ liệu để website ứng dụng của bạn có càng nhiều thông tin hữu ích càng quan trọng vì việc đó càng tăng khả năng nhưng tiếp cận của bạn đến mọi người trên internet.

Tất cả các doanh nghiệp lớn làm thương mại điện tử, rao vặt như: Muaban, Chotot, Muabannhanh... Đều có hệ thống crawl data của họ cả. Vì càng có nhiều content thì bạn càng làm chủ vì Content Is King. Có rất nhiều công cụ hữu ích đã nêu trên không phải ai cũng nói cho bạn nghe bí mật như LPTech đã chia sẻ. Hãy liên hệ với LPTech ngày hôm nay để có thể tiếp cận công nghệ đỉnh cao trong thời đại Big data hiện nay.