**Tiện ích mở rộng Java: tổng quan chuyên sâu**

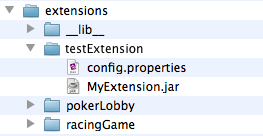
Trong bài viết này, chúng ta sẽ tìm hiểu sâu hơn về các chi tiết và tùy chọn khả dụng khi phát triển Tiện ích mở rộng phía máy chủ. Nếu bạn đã bỏ qua hai chương đầu tiên trong phần này, chúng tôi khuyên bạn nên bắt đầu từ đó trước khi tiếp tục.

Nếu bạn cần chuyển đổi giữa các phần thì đây là bảng phân tích các chủ đề chính:

* [**Tổng quan về triển khai**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview#deployment)
* [**Giải phẫu của một phần mở rộng**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview#anatomy)
* [**Khái niệm mở rộng nâng cao**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview#advanced)

**» Tổng quan triển khai**

Như đã thấy trong hướng dẫn bắt đầu nhanh, Tiện ích mở rộng Java được triển khai dưới **dạng (các) tệp .jar** vào một thư mục đại diện cho tên của chính Tiện ích mở rộng đó. Hình ảnh bên dưới hiển thị thư mục **/phần mở rộng chính** đóng vai trò là gốc cho tất cả các Phần mở rộng trong máy chủ. Thư mục *testExtension* của chúng tôi chứa một tệp jar chứa tất cả mã của chúng tôi.



Có một vài điều nữa cần chú ý:

* Bạn có thể triển khai nhiều tệp jar trong cùng một thư mục Tiện ích mở rộng, ví dụ: phụ thuộc hoặc thư viện khác. Tất cả chúng sẽ được tải trong cùng một Trình tải lớp.
* Có một thư mục đặc biệt gọi là **\_\_lib\_\_** nơi bạn cũng có thể đặt các phụ thuộc. Bằng cách này, bạn có thể chọn thư viện nào được chia sẻ trên nhiều Tiện ích mở rộng và thư viện nào là cục bộ của một Tiện ích mở rộng cụ thể.

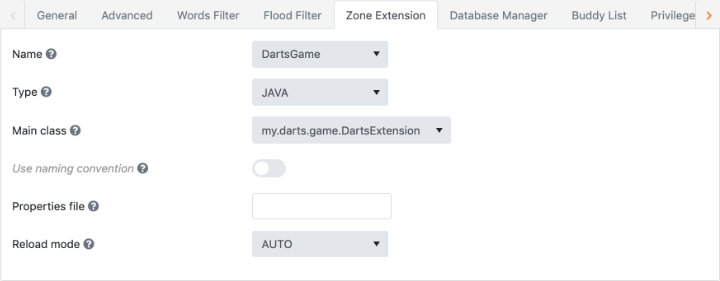
Cả hai cách tiếp cận đều có thể hữu ích tùy thuộc vào những gì bạn cần làm:

* Các thư viện được triển khai trong thư mục Tiện ích mở rộng của bạn sẽ được tải trong Trình tải lớp mở rộng. Điều này có nghĩa là bạn có thể thay đổi thư viện cụ thể đó mà không ảnh hưởng đến các Tiện ích mở rộng khác.
* thư viện được chia sẻ trong thư mục **\_\_lib\_\_** được tải trong Trình tải lớp trên cùng. Nếu bạn thay đổi bất kỳ phần phụ thuộc nào trong số này, nó sẽ ảnh hưởng đến tất cả các Tiện ích mở rộng sử dụng chúng, nhưng bạn sẽ tiết kiệm tài nguyên bộ nhớ vì thư viện chỉ được tải một lần.

**» Cấu hình tùy chỉnh**

Tiện ích mở rộng có thể tự động tải **tệp .properties** với cài đặt tùy chỉnh có sẵn ngay lập tức trong mã của bạn. Theo mặc định, máy chủ cố gắng tải một tệp có tên *config.properties* . Bạn cũng có thể chỉ định một tên tệp khác, nếu cần. Điều này đặc biệt hữu ích khi đính kèm cùng một Tiện ích mở rộng cho nhiều Vùng (hoặc Phòng) và bạn cần chuyển các cài đặt khác nhau cho từng vùng.

Dưới đây là ví dụ về cách một Tiện ích mở rộng được định cấu hình trong Công cụ quản trị (xem tab **[Tiện ích mở rộng vùng](http://docs2x.smartfoxserver.com/GettingStarted/admintool-ZoneConfigurator-zone" \l "zoneExtension)** hoặc tab **[Tiện ích mở rộng phòng](http://docs2x.smartfoxserver.com/GettingStarted/admintool-ZoneConfigurator-room" \l "roomExtension)** trong tài liệu mô-đun Bộ **cấu hình vùng để biết thêm thông tin và ghi chú quan trọng):**



Ở đâu:

* *Tên* đề cập đến tên của thư mục chứa mã;
* *Loại* cho biết loại Tiện ích mở rộng đang sử dụng (khuyên dùng Java);
* *Lớp chính* là tên đủ điều kiện của lớp Tiện ích mở rộng chính (hoặc tên tệp tập lệnh Python), lớp triển khai phương thức init().
* *Sử dụng quy ước đặt tên* không phải là cài đặt Tiện ích mở rộng: nó chỉ kích hoạt/hủy kích hoạt bộ lọc trên trường *Lớp chính* để chỉ hiển thị những lớp có tên kết thúc bằng từ "Tiện ích mở rộng";
* *Tệp thuộc tính* là tên tùy chọn của tệp **.properties** được triển khai trong thư mục Tiện ích mở rộng chứa cài đặt Tiện ích mở rộng tùy chỉnh (mặc định là **config.properties** );
* *Chế độ tải lại* cho biết nếu Tiện ích mở rộng được theo dõi và tự động tải lại ( **AUTO** ), được tải lại thủ công ( **MANUAL** ) hay không thể tải lại ( **NONE** ).

**»Tải lại tiện ích mở rộng**

SmartFoxServer 2X cung cấp Tiện ích triển khai lại nóng có thể rất hữu ích trong các giai đoạn phát triển. Khi tính năng này được bật (xem đoạn **Cấu hình tùy chỉnh** ở trên), máy chủ sẽ theo dõi các thư mục Tiện ích mở rộng của bạn và tải lại mã của bạn khi tệp jar được triển khai lại.

Tất cả những gì bạn cần làm là định cấu hình IDE Java của mình để xây dựng hoặc sao chép tệp jar trực tiếp trong thư mục Tiện ích mở rộng trong đường dẫn SmartFoxServer của bạn và bạn sẽ có một hệ thống triển khai chỉ bằng một cú nhấp chuột.

**» Ghi nhật ký và phụ thuộc thêm**

Bạn có thể ghi các thông báo trực tiếp vào các tệp nhật ký SFS2X chính (nằm trong SFS2X/logs/smartfox.log) bằng cách sử dụng phương pháp **theo dõi()** , như trong ví dụ này:

Java

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview)

|  |  |
| --- | --- |
| 1 | trace("Hello world, I am testing the logging feature"); |

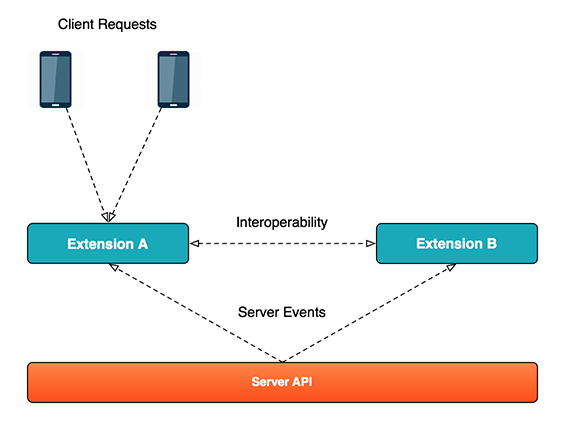
Điều này là đủ cho hầu hết các yêu cầu ghi nhật ký và gỡ lỗi nhưng nếu bạn cần tùy chỉnh đầy đủ định dạng và hành vi ghi nhật ký, bạn có thể làm việc với API ghi nhật ký SFS2X ( **[SLF4J](https://www.slf4j.org/" \t "_blank)** + **[Log4J](https://logging.apache.org/log4j/2.x/" \t "_blank)** ) bằng cách nhập một số thư viện bổ sung được cung cấp trong quá trình cài đặt.

Đây là bài viết chi tiết về **[cách tùy chỉnh các bản ghi Tiện ích mở rộng](http://smartfoxserver.com/blog/custom-logging-for-sfs2x-extensions/" \t "_blank)** trong SmartFoxServer 2X.

[**^menu trên cùng**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview#items)

**» Cấu tạo của tiện ích mở rộng**

Sơ đồ sau đây cho thấy cách Tiện ích mở rộng giao tiếp với máy chủ và thế giới bên ngoài:



Có ba cách giao tiếp:

* **Khách hàng được kết nối** : bằng cách trả lời yêu cầu hoặc gửi thông báo
* **Sự kiện máy chủ** : bằng cách phản ứng với các sự kiện cụ thể do máy chủ tạo ra
* **Khả năng tương tác của tiện ích mở rộng** : bằng cách gửi hoặc nhận tin nhắn từ các Tiện ích mở rộng khác đang chạy trong hệ thống.

**» Khách hàng được kết nối**

Trong hướng dẫn "Bắt đầu nhanh" của chúng tôi, chúng tôi đã thấy một ví dụ Tiện ích mở rộng nhận hai số từ khách hàng và gửi lại tổng của chúng. Toàn bộ quá trình dựa trên việc xác định một loạt các **lệnh tùy chỉnh** mà máy chủ sẽ nhận ra và chuyển đến một trình xử lý cụ thể đã đăng ký trong Tiện ích mở rộng.

Về phía khách hàng, chúng tôi có một cái gì đó giống như thế này ( sfs là thể hiện của lớp *SmartFox* ):

C# JS AS3

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | ISFSObject **params** = SFSObject.NewInstance();  **params**.PutInt("number", 100);  **params**.PutBool("bool", **true**);  **params**.PutUtfString("str", "Hello");    sfs.Send(**new** Sfs2X.Requests.ExtensionRequest("test", **params**)); |

" **test** " là tên của lệnh tùy chỉnh của chúng tôi và **params** luôn là một **[SFSObject](http://docs2x.smartfoxserver.com/DevelopmentBasics/sfsobject-sfsarray)** chứa bất kỳ tham số nào chúng tôi cần gửi đến máy chủ (ví dụ: tọa độ của người chơi, thông tin xác thực của người dùng, v.v.). Mặt khác, trong mã Tiện ích mở rộng của chúng tôi, chúng tôi có thể khai báo bao nhiêu trình xử lý cần thiết với mã này:

Java

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview)

|  |  |
| --- | --- |
| 1  2  3  4  5 | @Override  **public** **void** init()  {      addRequestHandler("test", TestHandler.**class**);  } |

Lưu ý cách mỗi trình xử lý được cung cấp dưới dạng đối tượng lớp: trong trường hợp này (mà chúng tôi khuyên dùng) trình xử lý được khởi tạo và quản lý bởi SmartFoxServer. Bạn cũng có thể sử dụng chữ ký thay thế của cùng một phương thức lấy phiên bản **IClientRequestHandler** , mặc dù theo cách này, bạn sẽ quản lý vòng đời của tất cả các trình xử lý theo cách thủ công, có thể hoạt động nhiều hơn và dễ bị lỗi.

Như đã thấy trước đó, các trình xử lý yêu cầu mở rộng lớp **BaseClientRequestHandler** và triển khai logic tương đối trong phương thức *handleClientRequest* .

Java

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | **public** **class** MyReqHandler **extends** BaseClientRequestHandler  {      @Override  **public** **void** handleClientRequest(User sender, ISFSObject params)      {  **int** number = params.getInt("number");  **boolean** bool = params.getBool("bool");          string str = params.getUtfString("str);            // ... more logic here ...      }  } |

**» Sự kiện phía máy chủ**

Các sự kiện máy chủ có thể hữu ích để phản ứng với các thay đổi trạng thái cụ thể trong SmartFoxServer, chẳng hạn như ngắt kết nối người dùng, tạo Phòng, v.v. Các tiện ích mở rộng có thể đăng ký một số sự kiện thông qua phương thức *addEventHandler* như đã thấy trong ví dụ bắt đầu nhanh:

Java

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | **public** **class** MyExtension **extends** SFSExtension  {      @Override  **public** **void** init()      {          addEventHandler(SFSEventType.USER\_JOIN\_ZONE, ZoneEventHandler.**class**);      }    **public** **class** ZoneEventHandler **extends** BaseServerEventHandler      {          @Override  **public** **void** handleServerEvent(ISFSEvent event) **throws** SFSException          {              User user = (User) event.getParameter(SFSEventParam.USER);              trace("Welcome new user: " + user.getName());          }      }  } |

Mỗi sự kiện sẽ gửi Tiện ích mở rộng một bộ tham số khác nhau thông qua đối tượng *sự kiện , tất cả đều được ghi lại trong* **[javadoc tương đối](http://docs2x.smartfoxserver.com/api-docs/javadoc/server/com/smartfoxserver/v2/core/SFSEventType.html" \t "_blank)** .

**» Khả năng tương tác tiện ích mở rộng**

Khi các Tiện ích mở rộng cần nói chuyện với nhau, chúng có thể làm như vậy bằng cách triển khai và gọi phương thức **handleInternalMessage** , có sẵn từ lớp cha. Đây là những gì phương pháp trông giống như:

Java

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview)

|  |  |
| --- | --- |
| 1 | Object handleInternalMessage(String cmdName, Object params) |

Điều này khá linh hoạt vì chúng ta có thể chuyển bất kỳ tên lệnh và đối tượng tham số nào và nhận lại bất kỳ đối tượng nào. Đây là một ví dụ triển khai ở cấp Vùng:

Java

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | @Override  **public** Object handleInternalMessage(String cmdName, Object params)  {  **if** (cmdName.equals("test"))      {          Vec3D vec = (Vec3D) params;            // Do something with parameters  **return** **new** Vec3D(x, y, z);      }  } |

Đây là cách Phần mở rộng phòng có thể gọi lệnh kiểm tra trên Phần mở rộng vùng:

Java

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview)

|  |  |
| --- | --- |
| 1  2  3 | MyZoneExtension zoneExt = (MyZoneExtension) getParentZone().getExtension();    Vec3D vec = zoneExt.handleInternalMessage("test", someVec3D); |

**LƯU Ý Vì các đối tượng được truyền từ loại**   
*java.lang.Object* cấp cao nhất sang bất kỳ thứ gì được sử dụng trong các cuộc gọi này, điều quan trọng là phải cẩn thận hơn để đảm bảo rằng các tham số được truyền chính xác ở cả hai đầu của giao tiếp. Không thể phát hiện lỗi truyền tại thời điểm biên dịch nhưng sẽ tạo ra phần đầu xấu xí của chúng khi chạy, với ClassCastException.

**» Phương thức hủy ()**

Cuối cùng, chúng ta cần xem xét phương thức **destroy()** được gọi bởi máy chủ khi một Tiện ích mở rộng bị tắt. Điều này xảy ra mỗi khi một Tiện ích mở rộng được triển khai lại ở Cấp vùng hoặc khi một Phòng bị xóa khỏi hệ thống và Tiện ích mở rộng tương ứng cần được tắt.

Về mặt kỹ thuật, không bắt buộc phải triển khai phương thức hủy (), trên thực tế, bạn có thể triển khai Tiện ích mở rộng mà không cần triển khai phương thức đó và mọi thứ sẽ hoạt động tốt. Tuy nhiên, có một số trường hợp ngoại lệ khi cần phải ghi đè phương thức: nếu bạn đang làm việc với các tệp, cơ sở dữ liệu hoặc bất kỳ tài nguyên nào khác có vòng đời phức tạp, bạn sẽ cần theo dõi những thứ đó và tắt chúng trong hủy () bị ghi đè .

**LƯU Ý**   
Bạn không cần phải xóa trình xử lý sự kiện được tạo theo cách thủ công tại thời điểm khởi tạo trong phương thức **hủy () của mình** . Trình nghe này đã được API máy chủ xử lý.

Một ví dụ điển hình về các ngoại lệ nói trên đang chạy một tác vụ theo lịch trình lặp lại:

Java

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | **public** **class** SchedulerTestExtension **extends** SFSExtension  {  **private** **class** TaskRunner **implements** Runnable      {  **private** **int** runningCycles = 0;    **public** **void** run()          {  **try**              {                  runningCycles++;                  trace("Inside the running task. Cycle:  " + runningCycles);              }  **catch** (Exception e)              {                  // Handle exceptions here              }          }      }        // Keeps a reference to the task execution      ScheduledFuture<?> taskHandle;        @Override  **public** **void** init()      {          SmartFoxServer sfs = SmartFoxServer.getInstance();            // Schedule the task to run every second, with no initial delay          taskHandle = sfs.getTaskScheduler().scheduleAtFixedRate(**new** TaskRunner(), 0, 1, TimeUnit.SECONDS);      }        @Override  **public** **void** destroy()      {  **super**.destroy();    **if** (taskHandle != **null**);              taskHandle.cancel();      }  } |

Tại thời điểm khởi tạo, chúng tôi tạo một tác vụ lặp lại chạy mỗi giây. Nếu chúng tôi không tắt tác vụ theo cách thủ công, tác vụ sẽ tiếp tục ngay cả sau khi Tiện ích mở rộng đã bị hủy, gây rò rỉ bộ nhớ. Để tránh điều này, chúng tôi đảm bảo hủy nó trong hàm hủy().

**LƯU Ý**   
Luôn đảm bảo gọi **super.destroy() trước** trong triển khai tùy chỉnh của bạn.

**» API máy chủ**

Bên cạnh việc triển khai phương thức gốc của Tiện ích mở rộng, bạn cũng có quyền truy cập vào nhiều loại API để thực hiện tất cả các tác vụ chính trong máy chủ, tương tác với Vùng, Phòng, Người dùng, v.v.

Bạn có thể tham khảo **[Javadoc API phía máy chủ](http://docs2x.smartfoxserver.com/api-docs/javadoc/server" \t "_blank)** để biết tất cả các chi tiết nhưng chúng tôi cũng khuyên bạn nên theo dõi tất cả các bài viết trong phần này để chúng ta sẽ quay lại chủ đề API phía máy chủ một cách chi tiết hơn.

[**^menu trên cùng**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview#items)

**» Các khái niệm mở rộng nâng cao**

Bây giờ chúng ta đã trình bày những kiến thức cơ bản về kiến trúc Tiện ích mở rộng 2.0 mới, chúng ta có thể đi sâu vào các tính năng nâng cao hơn cho phép kiểm soát mã của bạn phức tạp hơn.

**» Chú thích khởi tạo**

Chúng tôi cung cấp một số chú thích hữu ích có thể được sử dụng để chỉ định cách khởi tạo các lớp trình xử lý của bạn. Theo mặc định, khi bạn khai báo một lớp xử lý yêu cầu/sự kiện, lớp này sẽ được khởi tạo như mới trên mỗi cuộc gọi. Bạn có thể thay đổi hành vi này bằng cách sử dụng chú thích **@Instantiation** trên các lớp của mình:

* **@Instantiation(NEW\_INSTANCE)** : tạo một phiên bản mới trên mỗi cuộc gọi
* **@Instantiation(SINGLE\_INSTANCE)** : sử dụng cùng một phiên bản cho tất cả các cuộc gọi

**» Đa xử lý và cú pháp dấu chấm yêu cầu**

Để sắp xếp hợp lý các tên yêu cầu trong ứng dụng phức tạp, chúng tôi đã thiết lập một quy ước tương tự như cách đặt tên gói Java sử dụng "cú pháp dấu chấm". Giả sử Tiện ích mở rộng của bạn có thể xử lý một số trò chơi và các hoạt động khác như đăng ký người dùng và chỉnh sửa hồ sơ. Bạn có thể tổ chức tất cả các yêu cầu như thế này:

register.submitForm   
register.passwordLostregister.changeEmailregister. ...Đăng ký. ...   
  
profile.changeAvatarTypeprofile.changeNickprofile.getAvatarprofile. ...Hồ sơ. ...   
  
cờ đam.sendMovecheckers.getMyScorecheckers.leaveGamecheckers. ... quân cờ. ...

API tiện ích mở rộng cung cấp chú thích **@MultiHandler** có thể được thêm vào định nghĩa lớp trình xử lý của bạn. Điều này sẽ đăng ký lớp cho tất cả các yêu cầu bắt đầu bằng một tiền tố nhất định. Hãy triển khai một trình xử lý đa năng cho tập hợp các yêu cầu *đăng ký* :

Java

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | @MultiHandler  **public** **class** RegisterMultiHandler **extends** BaseClientRequestHandler  {      @Override  **public** **void** handleClientRequest(User sender, ISFSObject params)      {          // Obtain the request custom name          String command = params.getUtfString(SFSExtension.MULTIHANDLER\_REQUEST\_ID);    **if** (command.equals("submitForm"))              handleSubmitForm(sender, params);  **else** **if** (command.equals("changeEmail"))              handleChangeEmail(sender, params);            // ... etc ...      }    **private** **void** handleSubmitForm(User sender, ISFSObject params)      {          // ... app logic here      }    **private** **void** handleChangeEmail(User sender, ISFSObject params)      {          // ... app logic here      }  } |

Trong trình xử lý nhiều trình xử lý, id yêu cầu cụ thể trong số các "đăng ký" của ví dụ trên được lấy từ đối tượng tham số được truyền cho trình xử lý. Sau đó, câu lệnh **if** hoặc **switch** có thể được sử dụng để thực thi mã thích hợp tùy thuộc vào id yêu cầu.

Bây giờ chúng ta đăng ký lớp trong phương thức **init** của Extension như chúng ta đã học trước đây.

Java

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | @Override  **public** **void** init()  {      trace("Hello, this is my first multi handler test!");        // Add a new Request Handler      addRequestHandler("register", RegisterMultiHandler.**class**)  } |

Sự khác biệt thực sự duy nhất trong ví dụ này là lớp trình xử lý được đánh dấu là **@MultiHandler** . Khi điều này được thực hiện, bộ điều phối Tiện ích mở rộng sẽ gọi trình xử lý trên bất kỳ id yêu cầu nào bắt đầu bằng tiền tố "đăng ký". Nói cách khác, nó sẽ xử lý **register.\*** .

**LƯU Ý**   
Bạn cũng có thể kết hợp chú thích **@Instantiation** với chú thích **@MultiHandler** .

Tóm lại, cũng đáng lưu ý rằng bạn không bị giới hạn ở một "dấu chấm" duy nhất trong tên yêu cầu. Bạn có thể có nhiều cấp độ lồng nhau, chẳng hạn như: *games.spacewars.fireBullet* hoặc *user.profile.avatar.getHairColor* , v.v. Khuyến nghị duy nhất mà chúng tôi có là giữ cho các tên yêu cầu này ngắn hợp lý vì chúng sẽ được truyền cùng với các đối tượng yêu cầu/phản hồi.

**» Bộ lọc mở rộng**

Tính năng nâng cao cuối cùng của chuyến tham quan này là **Bộ lọc tiện ích mở rộng** . Nếu bạn đã quen thuộc với API Java Servlet, điều này có thể sẽ rung chuông. Bộ lọc Tiện ích mở rộng trong SmartFoxServer được lấy cảm hứng từ các bộ lọc servlet và chúng phục vụ một mục đích tương tự: chúng được thực thi theo chuỗi và chúng có thể được sử dụng để ghi nhật ký, lọc hoặc xử lý các yêu cầu hoặc sự kiện cụ thể trước khi chúng đến chính Tiện ích mở rộng.

Ưu điểm của Bộ lọc có thể cắm là chúng không cản trở mã Tiện ích mở rộng của bạn, thứ tự thực thi của chúng có thể bị thay đổi và thậm chí chúng có thể dừng luồng thực thi nếu cần. Một ví dụ về điều này có thể là bộ lọc cấm tùy chỉnh trong đó thông tin đăng nhập của người dùng được kiểm tra dựa trên danh sách đen trước khi yêu cầu được chuyển đến Trình xử lý đăng nhập của bạn.

Đây là một ví dụ về bộ lọc Tiện ích mở rộng đơn giản:

Java

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | **public** **class** CustomFilter **extends** SFSExtensionFilter  {      @Override  **public** **void** init(SFSExtension ext)      {  **super**.init(ext);          trace("Filter inited!");      }        @Override  **public** **void** destroy()      {          trace("Filter destroyed!");      }        @Override  **public** FilterAction handleClientRequest(String cmd, User sender, ISFSObject params)      {          // If something goes wrong you can stop the execution chain here!  **if** (cmd.equals("BadRequest"))  **return** FilterAction.HALT;  **else**  **return** FilterAction.CONTINUE;      }        @Override  **public** FilterAction handleServerEvent(ISFSEvent event)      {  **return** FilterAction.CONTINUE;      }  } |

Có thể dễ dàng thêm các bộ lọc vào bất kỳ Tiện ích mở rộng nào tại thời điểm cấu hình hoặc động, tại thời điểm chạy:

Java

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/overview)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | @Override  **public** **void** init()  {      /\*      \* This is your Extension main class init()      \*/        // Add filters      addFilter("customLoginFilter", **new** CustomLoginFilter());      addFilter("pubMessageFilter", **new** PubMessageFilter());      addFilter("privMessageFilter", **new** PrivMessageFilter());  } |

Khi một yêu cầu hoặc sự kiện mới được gửi đến Tiện ích mở rộng của bạn, trước tiên, nó sẽ đi qua **Chuỗi bộ lọc** theo thứ tự các Bộ lọc đã được thêm vào. Trong ví dụ trên, nó sẽ là: **customLoginFilter** » **pubMessageFilter** » **privMessageFilter** » **Extension** .