Link : “ <http://docs2x.smartfoxserver.com/AdvancedTopics/system-controller-filters> ”

# » Bộ lọc SystemController

Kể từ **phiên bản SFS2X 2.3.0** , chúng tôi đã giới thiệu một bộ bộ lọc tùy chỉnh mới có tên là **Bộ lọc SystemController** mà các nhà phát triển có thể sử dụng để kiểm soát bổ sung các yêu cầu của máy khách.

## » Bộ điều khiển chính xác là gì?

Trước khi mô tả cách hoạt động của các bộ lọc, hãy sao lưu một chút và cung cấp định nghĩa chính xác về Bộ điều khiển. Theo kiến trúc SFS2X, Bộ điều khiển chịu trách nhiệm xử lý mọi yêu cầu của khách hàng, cụ thể:

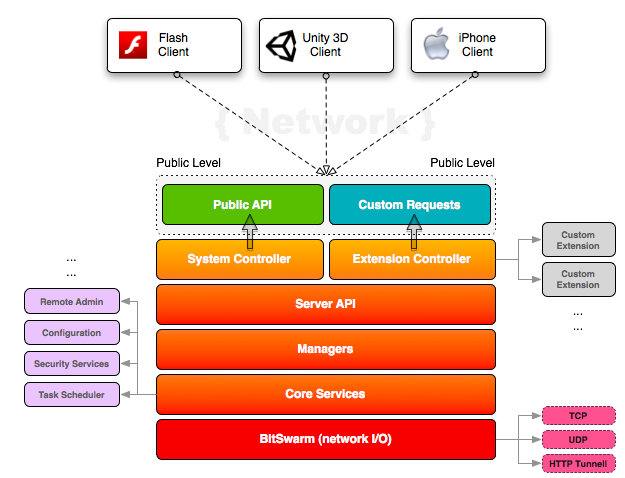
* **SystemController** : xử lý tất cả các lệnh gọi hệ thống như Đăng nhập, Tham gia, TạoRoom, PublicMessage, SetUserVariables, v.v. Nói cách khác, nó quản lý tất cả các lệnh gọi API của máy khách.
* **ExtensionController** : xử lý riêng các cuộc gọi đến Tiện ích mở rộng tùy chỉnh, gửi từng yêu cầu đến trình xử lý thích hợp.

Trong các chương trước của phần này, chúng tôi đã giải thích cách thức hoạt động của ExtensionController và cách các bộ lọc có thể được áp dụng cho Mã mở rộng. Trong bài viết này, chúng tôi sẽ tập trung vào các bộ lọc có thể được áp dụng cho SystemController.

## » Lọc các yêu cầu SystemController

Hình ảnh bên dưới hiển thị chi tiết đường dẫn mà một yêu cầu TCP/UDP đi từ lớp I/O của mạng dưới cùng đến một trong các Bộ điều khiển mà chúng tôi vừa mô tả (nhân tiện, SFS2X không chỉ giới hạn ở 2 bộ điều khiển, nó thực sự có thể sử dụng đồng thời tới 256).

Mặc dù nhà phát triển có toàn quyền kiểm soát yêu cầu Tiện ích mở rộng, nhưng mức độ kiểm soát đối với SystemController bị giới hạn ở các sự kiện được kích hoạt tại Tiện ích mở rộng: ví dụ: sự kiện ROOM\_JOIN hoặc PUBLIC\_MESSAGE mà anh ta có thể phản ứng nhưng không thể sửa đổi hoặc tương tác.



Đó chính xác là những gì **Bộ lọc SystemController** cung cấp. Có thể dễ dàng cắm các bộ lọc (thông qua mã) vào SystemController và chúng có thể tương tác với từng yêu cầu của máy khách, cho phép bạn lọc các tham số, thêm logic bổ sung và xác thực, v.v. Dưới đây là một vài trường hợp sử dụng ví dụ:

* bỏ qua tính năng lọc Tin nhắn công khai/riêng tư/bạn bè, thêm bộ lọc tùy chỉnh của riêng bạn hoặc giải pháp của bên thứ ba
* bỏ qua Bộ lọc chống tràn bằng triển khai tùy chỉnh hoặc bên thứ ba
* thêm logic tùy chỉnh vào đầu các yêu cầu như JoinRoom và CreateRoom
* thêm tính năng lọc hoặc logic tùy chỉnh vào các yêu cầu như SetUserVariables và SetRoomVariables

## " Làm thế nào nó hoạt động

Hãy lấy trường hợp sử dụng đầu tiên và xem cách chúng ta có thể tạo bộ lọc tương tác với yêu cầu **PublicMessage** , chẳng hạn như bằng cách xóa tất cả các trường hợp ký tự số trong mọi thư.

Chúng tôi bắt đầu bằng cách tạo một Lớp mới có tên PublicMessageNumberFilter mở rộng lớp cơ sở **SysControllerFilter** , từ gói sfs2x.extension.filters . Đây là một lớp rất đơn giản chỉ yêu cầu một phương thức được ghi đè:

Java

[**?**](http://docs2x.smartfoxserver.com/AdvancedTopics/system-controller-filters)

|  |  |
| --- | --- |
| 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 | **public** **class** PublicMessageNumberFilter **extends** SysControllerFilter  {      @Override  **public** FilterAction handleClientRequest(User sender, ISFSObject params) **throws** SFSException      {          StringBuilder message = **new** StringBuilder(params.getUtfString(GenericMessage.KEY\_MESSAGE));            // Character pointer  **int** p = 0;    **while** (p < message.length())          {  **char** ch = message.charAt(p);                // Remove numeric characters  **if** (ch >= 0x30 && ch <= 0x39)                  message.deleteCharAt(p);  **else**                  p++;          }            // Store new message in parameters list          params.putUtfString(GenericMessage.KEY\_MESSAGE, message.toString());  **return** FilterAction.CONTINUE;      }  } |

Thật thú vị khi lưu ý rằng phương thức **handleClientRequest()** luôn trả về một đối tượng FilterAction. Khi chúng tôi trả lại **FilterAction.CONTINUE,** chúng tôi cho phép bộ lọc tiếp theo trong chuỗi tiếp tục luồng thực thi của nó xuống cho đến SystemController. Tuy nhiên, nếu bất kỳ bộ lọc nào trả về **FilterAction.HALT** , quá trình thực thi sẽ dừng ở đó mà không tuyên truyền thêm nữa.

Bây giờ chúng tôi đã chuẩn bị bộ lọc, chúng tôi có thể cắm nó vào SystemController thông qua Tiện ích mở rộng của chúng tôi. Thông thường, chúng tôi khuyên bạn nên thực hiện việc này trong phương thức init() của Tiện ích mở rộng cấp vùng của bạn:

Java

[**?**](http://docs2x.smartfoxserver.com/AdvancedTopics/system-controller-filters)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | **public** **class** FilteringExtension **extends** SFSExtension  {      @Override  **public** **void** init()      {          // Reset filter chain to clean previous filters          getParentZone().resetSystemFilterChain();            ISystemFilterChain filterChain = **new** SysControllerFilterChain();          filterChain.addFilter("numberFilter", **new** PublicMessageNumberFilter());            // Plug the filter chain          getParentZone().setFilterChain(SystemRequest.PublicMessage, filterChain);      }        @Override  **public** **void** destroy()      {  **super**.destroy();      }  } |

Dòng mã đầu tiên trong phương thức init() sẽ xóa tất cả các bộ lọc trước đó mà chúng tôi có thể đã áp dụng. Đây là một thao tác quan trọng vì chúng tôi có thể tải lại Tiện ích mở rộng nhiều lần trong quá trình thử nghiệm và ngay cả trong quá trình sản xuất và chúng tôi không muốn các bộ lọc trùng lặp. Sau đó, chúng tôi tiến hành tạo một chuỗi bộ lọc mới, chúng tôi thêm lớp **PublicMessageNumberFilter** và cuối cùng chúng tôi đăng ký nó với yêu cầu **PublicMessage** . Đây là tất cả những gì chúng ta cần để lọc tin nhắn trò chuyện công khai hoặc bất kỳ yêu cầu API nào khác theo cách tương tự.

Cuối cùng, xin lưu ý rằng bằng cách sử dụng chuỗi bộ lọc, bạn có thể thêm bất kỳ số lượng bộ lọc nào theo thứ tự thực hiện mà bạn muốn, lưu ý rằng bộ lọc đầu tiên được thêm vào chuỗi là bộ lọc đầu tiên được thực thi, v.v.

## » Triển khai bộ lọc

Cách dễ nhất để triển khai các bộ lọc của riêng bạn là gộp chúng trong tệp .jar Tiện ích mở rộng vùng chính của bạn. Điều này sẽ đảm bảo rằng các bộ lọc sẽ được tải lại nóng với mã Tiện ích mở rộng của bạn khi bạn triển khai lại.

Tuy nhiên, nếu bạn dự định cung cấp các bộ lọc dưới dạng giải pháp của bên thứ ba (ví dụ: Bộ lọc từ xấu tùy chỉnh), bạn có thể muốn cung cấp một tệp .jar riêng có thể được triển khai vào thư mục **tiện ích mở rộng/\_\_lib\_\_/** để bất kỳ Tiện ích mở rộng nào cũng có thể truy cập nó. Trong trường hợp này, việc tải lại nóng Tiện ích mở rộng sẽ không cập nhật các bộ lọc, do đó, trong trường hợp cập nhật bộ lọc, việc khởi động lại máy chủ là cần thiết. Nếu không muốn điều này, các bộ lọc có thể được triển khai trong cùng thư mục với Tiện ích mở rộng mà nó đang sử dụng.

Cuối cùng, đối với nhà phát triển bộ lọc bên thứ ba, sẽ thuận tiện hơn nếu cung cấp cho lớp một phương thức tĩnh đảm nhiệm việc thiết lập tất cả các bộ lọc cần thiết, ẩn nó khỏi mã Tiện ích mở rộng của nhà phát triển. Đây là một mẫu ví dụ:

Java

[**?**](http://docs2x.smartfoxserver.com/AdvancedTopics/system-controller-filters)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | **public** **class** CustomFilterSetup  {  **public** **static** **void** initialize(Zone targetZone)      {          // Reset filter chain to clean previous filters          targetZone.resetSystemFilterChain();            ISystemFilterChain filterChain = **new** SysControllerFilterChain();          filterChain.addFilter("numberFilter", **new** PublicMessageNumberFilter());            // Plug the filter chain          targetZone.setFilterChain(SystemRequest.PublicMessage, filterChain);      }  } |

Ví dụ trên dựa trên ví dụ **PublicMessageNumberFilter trước đây của chúng tôi** . Nhà phát triển tiện ích mở rộng sẽ chỉ cần gọi **CustomFilterSetup .initialize(...)** trong phương thức **init() của mình** .

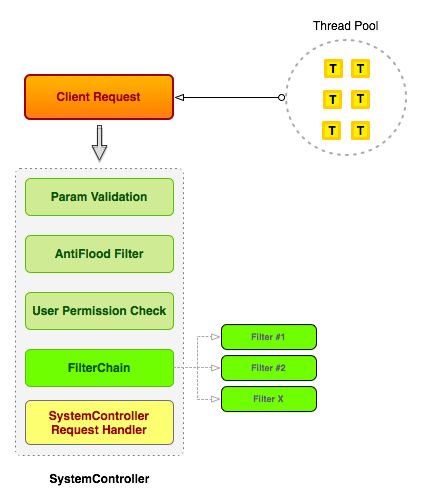
## » Đề xuất luồng

Tất cả các bộ lọc được thực thi trong (các) chuỗi **SystemController** . Thông thường, tất cả các trình xử lý Yêu cầu hệ thống được thực thi rất nhanh mà không bị chặn hoặc hoạt động lâu dài, do đó Bộ điều khiển có thể hoạt động chỉ với một luồng, điều này được khuyến nghị.

Nếu bạn dự định chạy các hoạt động chặn hoặc kéo dài chẳng hạn như lệnh gọi ổ cắm, truy vấn DB, yêu cầu HTTP, v.v., bạn nên lưu ý rằng bạn có thể làm chậm đáng kể Bộ điều khiển, do đó có thể dẫn đến phản hồi chậm cho máy khách. Trong trường hợp này, chúng tôi khuyên bạn nên thay đổi kích thước nhóm luồng SystemController hoặc, nếu cần, hãy chạy nhóm luồng chuyên dụng cho các hoạt động "chậm" (nhóm luồng SystemController có thể được định cấu hình từ [AdminTool](http://docs2x.smartfoxserver.com/GettingStarted/admintool-ServerConfigurator) ).

## » Bộ lọc chính xác được áp dụng ở đâu?

Để cung cấp cho bạn ý tưởng tốt hơn về cách thức hoạt động của SystemController và cách thức hoạt động của Bộ lọc, hãy xem sơ đồ sau:



Trước khi yêu cầu của khách hàng đến mã bộ lọc, có một số kiểm tra được áp dụng để đảm bảo rằng yêu cầu hợp lệ:

* **Xác thực tham số chính thức** : bất kỳ yêu cầu nào thiếu các tham số thiết yếu mà API máy chủ mong đợi đều bị loại bỏ với thông báo lỗi trong tệp nhật ký.
* **Bộ lọc chống ngập** : nếu bộ lọc này được bật, mỗi yêu cầu sẽ được kiểm tra theo các quy tắc đã định cấu hình và bị loại bỏ trong trường hợp phát hiện thấy nỗ lực làm ngập.
* **Kiểm tra** quyền của người dùng : yêu cầu cũng được xác thực đối với hồ sơ quyền của người gửi và bị loại bỏ trong trường hợp người dùng không được phép thực hiện lệnh đó.

Nếu bất kỳ xác thực nào trong số này không thành công, thông báo sẽ không bao giờ đến được mã bộ lọc, vì vậy nếu bạn không nhận được thông báo cụ thể, bạn nên kiểm tra kỹ cài đặt cho Cấu hình cấp phép và Bộ lọc chống tràn.

**XIN LƯU Ý**   
Trong trường hợp các tin nhắn như tin nhắn Công khai/Riêng tư/Bạn bè, Bộ lọc Từ ngữ Xấu được áp dụng **sau** Bộ lọc SystemController. Nếu bạn đang thay thế tính năng này bằng một giải pháp tùy chỉnh, vui lòng đảm bảo **tắt Bộ lọc từ xấu** trong cấu hình Vùng.

## » Các yêu cầu SystemController được hỗ trợ

Đây là danh sách các yêu cầu hiện được hỗ trợ để lọc và các tham số tương đối cho từng yêu cầu:

 JoinRoom

 CreateRoom

 ObjectMessage

 SetRoomVariables

 SetUserVariables

 LeaveRoom

 SubscribeRoomGroup

 UnsubscribeRoomGroup

 PublicMessage

 PrivateMessage

 ModeratorMessage

 AdminMessage

 KickUser

 BanUser

 SetUserPosition (\*)

 AddBuddy (\*)

 BlockBuddy (\*)

 RemoveBuddy (\*)

 SetBuddyVariables (\*)

 BuddyMessage (\*)

 InviteUser (\*)

 InvitationReply (\*)

 QuickJoinGame (\*)