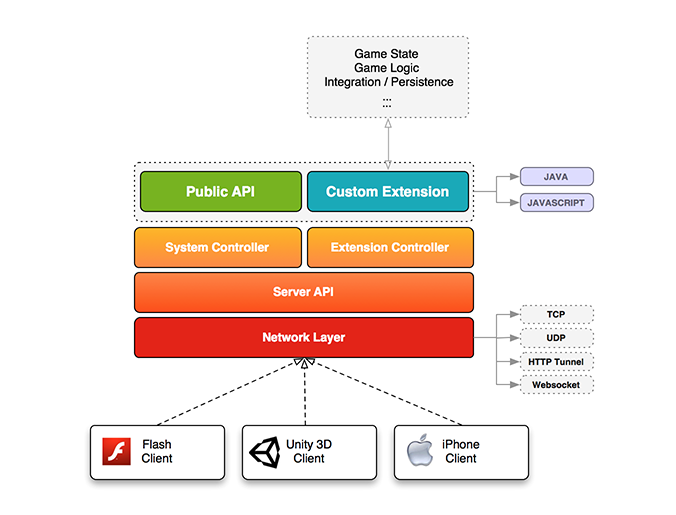
**Tiện ích mở rộng Java: tổng quan**

SmartFoxServer 2X cho phép các nhà phát triển viết mã phía máy chủ của họ để mở rộng khả năng của máy chủ, tích hợp nó với các công nghệ khác và quản lý trạng thái trò chơi ở một nơi tập trung.



Sơ đồ trên cho thấy một phiên bản đơn giản hóa của ngăn xếp máy chủ, trong đó các máy khách ở dưới cùng kết nối thông qua một số giao thức và có thể gọi cả API máy chủ công cộng cũng như các chức năng tùy chỉnh do nhà phát triển trò chơi đưa ra thông qua Tiện ích mở rộng.

Nói một cách đơn giản , Tiện ích mở rộng SmartFoxServer dành cho các trò chơi giống như các trang PHP/JSP dành cho các trang web. Chúng cho phép mã tùy chỉnh chạy ở phía máy chủ và thêm logic ứng dụng hoàn toàn mới.

Trong SmartFoxServer, Tiện ích mở rộng 2X có thể được gắn vào Vùng hoặc Phòng, tùy thuộc vào phạm vi của chính Tiện ích mở rộng. Tiện ích mở rộng phòng có phạm vi nhỏ hơn, xử lý các sự kiện, cuộc gọi và người dùng của Phòng đó, trong khi Tiện ích mở rộng vùng có thể lắng nghe số lượng sự kiện lớn hơn nhiều và kiểm soát tất cả các Phòng và người dùng.

**» Hãy viết một Extension**

Java

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/quick-start)

|  |  |
| --- | --- |
| 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 | **public** **class** MyExtension **extends** SFSExtension  {      @Override  **public** **void** init()      {          trace("Hello, this is my first SFS2X Extension!");            // Add a new Request Handler          addRequestHandler("sum", SumReqHandler.**class**);      }    **public** **static** **class** SumReqHandler **extends** BaseClientRequestHandler      {          @Override  **public** **void** handleClientRequest(User sender, ISFSObject params)          {              // Get the client parameters  **int** n1 = params.getInt("n1");  **int** n2 = params.getInt("n2");                // Create a response object              ISFSObject resObj = **new** SFSObject();              resObj.putInt("res", n1 + n2);                // Send it back              send("sum", resObj, sender);          }      }  } |

Mỗi Tiện ích mở rộng kế thừa từ lớp cơ sở **SFFSextension** và triển khai phương thức **init ( )** , phương thức này được gọi bởi máy chủ khi khởi động. Ở đây chúng ta có thể khởi tạo các đối tượng toàn cầu hoặc cấu trúc dữ liệu và đăng ký các trình xử lý sự kiện. Trong ví dụ này, chúng tôi khai báo một lệnh gọi là " *sum* " được xử lý bởi*SumReqHandler* .

Trình xử lý trích xuất các tham số từ **SFSObject** do máy khách truyền (được sử dụng để vận chuyển dữ liệu giữa máy khách và máy chủ) và gửi phản hồi trở lại. Bạn có thể coi SFSObject như một đối tượng giống như từ điển đơn giản để lưu trữ các cặp khóa/giá trị, với lợi ích bổ sung là khai báo chính xác loại dữ liệu là gì. Điều này lần lượt cho phép tối ưu hóa việc sử dụng mạng.   
Thông tin thêm về SFSObject có sẵn trong **[tài liệu này](http://docs2x.smartfoxserver.com/DevelopmentBasics/sfsobject-sfsarray)** ; cũng kiểm tra **[JSDoc](http://docs2x.smartfoxserver.com/api-docs/javadoc/server/com/smartfoxserver/v2/entities/data/SFSObject.html" \t "_blank)** .

Tiện ích mở rộng cũng có thể khai báo phương thức **hủy ( )** nhưng không bắt buộc. Nó chỉ được yêu cầu để giải phóng tài nguyên yêu cầu giải phóng thủ công, chẳng hạn như kết nối cơ sở dữ liệu đang hoạt động, tệp đang mở hoặc tác vụ theo lịch trình.

Lưu ý thêm, phương thức **theo dõi ( )** cho phép bạn xuất thông tin nhật ký ở phía máy chủ.

Bây giờ, hãy xây dựng một ứng dụng khách đơn giản để kết nối với máy chủ và kiểm tra mã máy chủ tùy chỉnh của chúng tôi:

C# (Unity) JS AS3

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/quick-start)

|  |  |
| --- | --- |
| 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  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75 | **public** **class** ExtensionTester : MonoBehaviour  {  **private** SmartFox sfs;    **void** Start()      {          // Create SmartFox client instance          sfs = **new** SmartFox();            // Add event listeners          sfs.AddEventListener(SFSEvent.CONNECTION, OnConnection);          sfs.AddEventListener(SFSEvent.CONNECTION\_LOST, OnConnectionLost);          sfs.AddEventListener(SFSEvent.LOGIN, OnLogin);          sfs.AddEventListener(SFSEvent.LOGIN\_ERROR, OnLoginError);          sfs.AddEventListener(SFSEvent.EXTENSION\_RESPONSE, OnExtensionResponse);            // Set connection parameters          ConfigData cfg = **new** ConfigData();          cfg.Host = "127.0.0.1";          cfg.Port = 9933;          cfg.Zone = "BasicExamples";            // Connect to SFS2X          sfs.Connect(cfg);      }    **void** Update()      {  **if** (sfs != **null**)              sfs.ProcessEvents();      }    **private** **void** OnConnection(BaseEvent evt)      {  **if** ((**bool**)evt.Params["success"])          {              Debug.Log("Connected");                // Send login request              sfs.Send(**new** Sfs2X.Requests.LoginRequest(""));          }  **else**              Debug.LogError("Connection failed");      }    **private** **void** OnConnectionLost(BaseEvent evt)      {          Debug.Log("Disconnected");      }    **private** **void** OnLogin(BaseEvent evt)      {          Debug.Log("Logged in as: " + sfs.MySelf.name);            // Send test request to Extension          ISFSObject **params** = SFSObject.NewInstance();  **params**.PutInt("n1", 25);  **params**.PutInt("n2", 17);            sfs.Send(**new** Sfs2x.Requests.ExtensionRequest("sum", **params**));      }    **private** **void** OnLoginError(BaseEvent evt)      {          Debug.LogError("Login error: " + (**string**)evt.Params["errorMessage"]);      }    **private** **void** OnExtensionResponse(BaseEvent evt)      {          // Retrieve response object          ISFSObject responseParams = (SFSObject)evt.Params["params"];            Debug.Log("Result: " + responseParams.GetInt("res"));      }  } |

Trong hàm **onLogin ( )** , chúng tôi bọc các tham số của mình và gửi lệnh thông qua đối tượng ExtensionRequest . Để xử lý phản hồi của máy chủ, chúng tôi đảm bảo đăng ký sự kiện **EXTENSION\_RESPONSE** và đọc kết quả bằng (các) khóa tương tự được sử dụng ở phía máy chủ.

**» Sự kiện phía máy chủ**

Ngoài việc xử lý các yêu cầu của máy khách, Tiện ích mở rộng phía máy chủ cũng có thể lắng nghe một số sự kiện của Máy chủ, chẳng hạn như sự kiện đăng nhập, đăng xuất, tham gia phòng và nhiều sự kiện khác.

Lắng nghe các sự kiện của máy chủ cũng đơn giản như việc xử lý các yêu cầu của máy khách: chúng ta chỉ cần tạo một hàm và đăng ký nó làm trình xử lý sự kiện. Đây là một ví dụ cơ bản không có mục đích nào khác ngoài việc thể hiện chức năng này:

Java

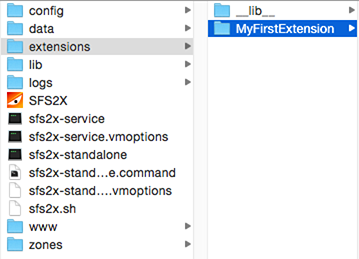
[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/quick-start)

|  |  |
| --- | --- |
| 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** MyExtension **extends** SFSExtension  {      @Override  **public** **void** init()      {          trace("Hello, this is my first SFS2X Extension!");            // Add a new Event Handler          addEventHandler(SFSEventType.USER\_JOIN\_ZONE, ZoneEventHandler.**class**);      }        @Override  **public** **void** destroy()      {          // Always make sure to invoke the parent class first  **super**.destroy();            trace("Destroy is called!");      }    **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());          }      }  } |

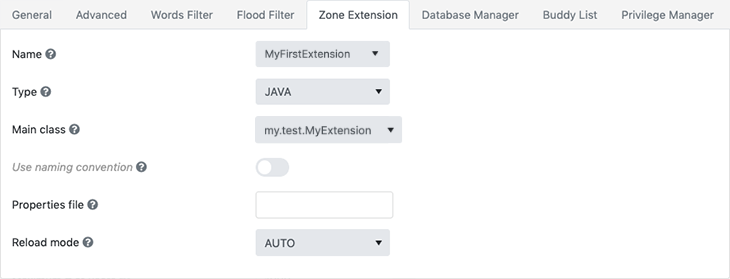
Mỗi sự kiện cung cấp một số tham số có thể được truy cập như được hiển thị trong mã. Để biết danh sách đầy đủ các sự kiện, hãy kiểm tra [**tài liệu tại đây**](http://docs2x.smartfoxserver.com/api-docs/javadoc/server/com/smartfoxserver/v2/core/SFSEventParam.html) .

**» Triển khai mở rộng**

Cuối cùng , chúng ta có thể xem cách chúng ta có thể triển khai mã của mình đến máy chủ và chạy nó. Để tạo tiện ích mở rộng , bạn cần tạo thư mục mới trong **SFS2X/extensions/**



Sau đó, chúng tôi cần xuất các lớp Java đã biên dịch của mình sang tệp jar. Bước này thường được thực hiện với sự trợ giúp của IDE bạn chọn, chẳng hạn như [**Eclipse**](https://www.eclipse.org/downloads/?) , [**Netbeans**](https://netbeans.org/) hoặc [**Intellij IDEA**](https://www.jetbrains.com/idea/) . Tiếp theo , chúng tôi mở **AdminTool** , chọn Bộ **cấu hình vùng** và nhấp vào tab **Mở rộng vùng** .



Trước tiên, hãy đảm bảo rằng menu thả xuống **Loại** được đặt thành **Java** , sau đó từ danh sách thả xuống trên cùng, chọn thư mục Tiện ích mở rộng và *"Lớp chính"* là thư mục có phương thức init ( ). Bây giờ chúng ta có thể lưu và khởi động lại máy chủ để kích hoạt các thay đổi.

Chế **độ tải lại** được đặt thành *AUTO* cho phép tự động tải lại Tiện ích mở rộng mỗi khi có thay đổi đối với tệp chính, do đó giúp tải lại mã của chúng tôi một cách nhanh chóng. Điều này thuận tiện cho việc phát triển và thử nghiệm nhưng trong quá trình sản xuất, tốt nhất nên đặt chế độ này thành *MANUAL* để tránh tải lại ngẫu nhiên.

**LƯU Ý:** nếu bạn muốn đính kèm Tiện ích mở rộng vào một Phòng cụ thể thay vì một Vùng, bạn có thể lặp lại quy trình chính xác tương tự, lần này chọn Phòng mục tiêu từ Bộ cấu hình Vùng.

**" Bước tiếp theo**

Có rất nhiều nền tảng để phát triển phần mở rộng. Khi bạn đã hiểu phần giới thiệu này, chúng tôi khuyên bạn nên xem [**video hướng dẫn này**](https://www.youtube.com/watch?v=nKGxhwJ0Ccc&list=PLC16B8E94B9D7C3E5) và theo dõi các bài viết tiếp theo trong phần này.