**SFSObject và SFArray**

SmartFoxServer 2X đã giới thiệu hai lớp cơ bản, **SFSObject** và **SFSArray** , đóng vai trò trung tâm trong thao tác và truyền dữ liệu giữa máy khách và máy chủ. Hai lớp này phổ biến trên tất cả các API ở tất cả các ngôn ngữ (bao gồm cả API phía máy chủ) giúp dễ dàng chuyển mã sang bất kỳ nền tảng nào và trên mọi phía của ứng dụng.

**SFSObject** và **SFSArray** đại diện cho các đối tượng cấp cao, trung lập với nền tảng, trừu tượng hóa việc vận chuyển dữ liệu giữa máy khách và máy chủ. Chúng được sử dụng để biểu diễn dữ liệu tương ứng dưới dạng Bản đồ/Từ điển hoặc Danh sách/Mảng; chúng có thể được lồng vào nhau để tạo cấu trúc dữ liệu phức tạp và chúng hỗ trợ nhiều loại dữ liệu khác nhau (từ byte đến số nguyên, nhân đôi, chuỗi, v.v.).   
Hai lớp này cung cấp khả năng kiểm soát chi tiết đối với từng thành phần dữ liệu được gửi qua mạng và cung cấp khả năng tuần tự hóa tốc độ cao bằng giao thức nhị phân SFS2X mặc định.

Hãy xem xét ví dụ đơn giản này: chúng ta cần gửi dữ liệu liên quan đến phương tiện chiến đấu trong trò chơi nhiều người chơi.

C# JS AS3

[**?**](http://docs2x.smartfoxserver.com/DevelopmentBasics/sfsobject-sfsarray)

|  |  |
| --- | --- |
| 1  2  3  4  5 | ISFSObject sfso = SFSObject.NewInstance();  sfso.PutByte("id", 10);  sfso.PutShort("health", 5000);  sfso.PutIntArray("pos", **new** Int[] {120, 150});  sfso.PutUtfString("name", "Hurricane"); |

Trong mã, chúng tôi sử dụng một Byte đơn (8 bit có dấu) để gửi bất kỳ giá trị số nguyên nhỏ nào, một số Ngắn (16 bit có dấu) cho các giá trị lớn hơn và số nguyên cho bất kỳ số nào sẽ được biểu diễn dưới dạng giá trị 32 bit (có dấu) thông thường . Trong ví dụ này, chúng tôi sử dụng một mảng số nguyên để gửi tọa độ X, Y. Tùy thuộc vào kích thước của khu vực trò chơi, các giá trị tọa độ này có thể được biểu thị bằng các loại khác nhau như float, short.

**» Các loại dữ liệu được hỗ trợ**

Đây là danh sách tất cả các loại được cả hai lớp hỗ trợ:

|  |  |  |
| --- | --- | --- |
| **Type** | **Bytes used** | **Ranges and Limits** |
| **NULL** | 1 | N/A |
| **BOOL** | 1 | true/false |
| **BYTE** | 1 | 8 bit Int, 0 - 2^8 (Java uses a signed byte) |
| **SHORT** | 2 | 16 bit Int, -2^15 to 2^15 |
| **INT** | 4 | 32 bit Int, -2^31 to 2^31 |
| **LONG** | 8 | 64 bit Int, -2^63 to 2^63 |
| **FLOAT** | 4 | [**See single precision floating point format**](http://en.wikipedia.org/wiki/Single_precision_floating-point_format) |
| **DOUBLE** | 8 | [**See double precision floating point format**](http://en.wikipedia.org/wiki/Double_precision_floating-point_format) |
| **UTF-STRING** | variable | [**UTF-8 multibyte encoding**](http://en.wikipedia.org/wiki/UTF-8) max string len: 2^15 chars (32 KB) |
| **BOOL ARRAY** | variable | max array len: 2^15 items (32767) |
| **BYTE ARRAY** | variable | max array len: 2^31 items (2147483648) |
| **SHORT ARRAY** | variable | max array len: 2^15 items (32767) |
| **INT ARRAY** | variable | max array len: 2^15 items (32767) |
| **LONG ARRAY** | variable | max array len: 2^15 items (32767) |
| **FLOAT ARRAY** | variable | max array len: 2^15 items (32767) |
| **DOUBLE ARRAY** | variable | max array len: 2^15 items (32767) |
| **UTF-STRING ARRAY** | variable | max array len: 2^15 items (32767) |
| **SFSOBJECT** | variable | max key-pair values: 2^15 items (32767) |
| **SFSARRAY** | variable | max array len: 2^15 items (32767) |
| **CLASS** | variable | N/A — [**Please check this advanced tutorial**](http://docs2x.smartfoxserver.com/AdvancedTopics/class-serialization) |

Các kiểu mảng đặc biệt hữu ích khi truyền danh sách các giá trị có cùng kiểu, vì kết quả là một cấu trúc dữ liệu rất nhỏ gọn. Mặt khác, nếu bạn cần gửi một danh sách các giá trị với các loại khác nhau, thì SFSObject sẽ là lựa chọn tốt nhất.

**LƯU Ý dành cho nhà phát triển JavaScript và ActionScript 3**   
Có sự khác biệt cơ bản giữa mảng JS/AS3 và các mảng này. Cái sau là **các mảng dày đặc** , nghĩa là chúng phải có một giá trị (hoặc ít nhất là *null* ) trong mỗi chỉ mục trong khi các mảng JS/AS3 không có cùng hạn chế.

**» Ví dụ về cách sử dụng**

Một ví dụ về việc sử dụng SFSObject/SFSArray là trong phát triển Tiện ích mở rộng (xem phần **Phát triển Tiện ích mở rộng** để biết thông tin chuyên sâu), nơi chúng được sử dụng để truyền mọi yêu cầu và phản hồi. Bằng cách mở rộng ví dụ được cung cấp ở đầu bài viết này, chúng ta hãy xem trường hợp sử dụng đầy đủ.

Máy khách cần truyền dữ liệu sau đến Tiện ích mở rộng máy chủ ( sfs là thể hiện của lớp *SmartFox* ):

C# JS AS3

[**?**](http://docs2x.smartfoxserver.com/DevelopmentBasics/sfsobject-sfsarray)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | **void** sendSomeData()  {      // Prepare data to be sent to the server Extension      ISFSObject sfso = SFSObject.NewInstance();      sfso.PutByte("id", 10);      sfso.PutShort("health", 5000);      sfso.PutIntArray("pos", **new** Int[] {120, 150});      sfso.PutUtfString("name", "Hurricane");        // Send request to Zone level Extension on server side      sfs.Send(**new** Sfs2X.Requests.ExtensionRequest("data", sfso));  } |

Phần mở rộng máy chủ sẽ nhận được dữ liệu giống như đối tượng tham số trong một trong các trình xử lý yêu cầu của nó:

Java

[**?**](http://docs2x.smartfoxserver.com/DevelopmentBasics/sfsobject-sfsarray)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | **public** **class** DataRequestHandler **extends** BaseClientRequestHandler  {      @Override  **public** **void** handleClientRequest(User sender, ISFSObject params)      {          // Get the client parameters  **byte** id = params.getByte("id");  **short** health = params.getShort("health");          Collection<Integer> pos = params.getIntArray("pos");          String name = params.getUtfString("name");            // Do something cool with the data          ...      }  } |

**» Kiểm tra một SFSObject/SFSArray**

Chúng ta hãy xem xét kỹ hơn hai lớp này và xem chi tiết những gì xảy ra đằng sau hậu trường. Cả hai đối tượng đều cung cấp hai phương thức hữu ích để kết xuất nội dung của chúng ở định dạng phân cấp hoặc kết xuất thập lục phân.

Nếu chúng ta lấy ví dụ tương tự được cung cấp ở đầu bài viết và gọi phương thức **SFSObject.getDump** trên SFSObject, chúng ta sẽ thấy kết quả này:

(short) health: 5000

(utf\_string) name: Hurricane

(byte) id: 10

(int\_array) pos: [120, 150]

Mỗi phần tử trong đối tượng được liệt kê với định dạng: ( *type* ) *key-name* : *value*

Nếu bạn muốn có chế độ xem cấp thấp hơn về cách đối tượng được biểu thị ở dạng nhị phân, bạn có thể gọi phương thức **SFSObject.getHexDump** :

Binary size: 54

12 00 04 00 06 68 65 61 6C 74 68 03 13 88 00 04 .....health.....

6E 61 6D 65 08 00 09 48 75 72 72 69 63 61 6E 65 name...Hurricane

00 02 69 64 02 0A 00 03 70 6F 73 0C 00 02 00 00 ..id....pos.....

00 78 00 00 00 96

**» Mảng byte**

Một đề cập đặc biệt nên đi đến loại **ByteArray** cung cấp phương tiện truyền dữ liệu nhị phân đến và từ máy chủ. Điều này có thể được sử dụng để truyền các tệp nhỏ, hình ảnh, tệp phương tiện, dữ liệu được mã hóa, v.v. Ví dụ: các nhà phát triển Flash có thể cải thiện tính bảo mật cho ứng dụng của họ bằng cách gửi các tệp SWF bên ngoài qua ổ cắm sẽ không bị phát hiện khi theo dõi lưu lượng HTTP.

Khi truyền các khối dữ liệu lớn, chúng tôi khuyên bạn nên **nén trước** chúng để tối ưu hóa kích thước và tránh quá tải cho phía Máy chủ. Cụ thể, tính năng nén giao thức động sẽ không khởi động đối với số lượng lớn dữ liệu (hàng chục/hàng trăm MB) để tránh suy giảm hiệu suất đáng kể (đặc biệt là trong điều kiện đồng thời nặng). Nên nén hoặc gzipping dữ liệu trước khi truyền.

**» Các phương pháp hay nhất về SFSObject/SFSArray**

SFSObject và SFSArray không phải là luồng an toàn, do đó, cần thận trọng hơn khi chia sẻ các đối tượng này trong môi trường đa luồng. Trong 90% trường hợp, SFSObject/SFSArray chỉ được sử dụng để truyền dữ liệu và được xử lý dưới dạng các biến cục bộ, do đó không xảy ra đồng thời.

Nếu bạn sử dụng một số lớp để biểu diễn mô hình trong trò chơi của mình, bạn nên thêm các phương thức **toSFSObject** và **newFromSFSObject** vào các lớp của mình để giúp chuyển đổi chúng thành biểu diễn SFSObject và ngược lại. Ít nhất bạn nên làm điều đó cho những lớp được truyền qua mạng thường xuyên hơn.

Đây là một ví dụ trong Java, nơi chúng tôi cho rằng chúng tôi có một trò chơi RTS với một số phương tiện được cập nhật thường xuyên cho khách hàng. Cụ thể, chúng tôi có một lớp *CombatQuad* đại diện cho một trong các phương tiện trong trò chơi. Về phía máy chủ, có lẽ chúng ta sẽ cần ít nhất phương thức *toSFSObject* để trích xuất các thuộc tính có liên quan từ thể hiện và gửi chúng trong bản cập nhật.

Java

[**?**](http://docs2x.smartfoxserver.com/DevelopmentBasics/sfsobject-sfsarray)

|  |  |
| --- | --- |
| 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 | **public** **class** CombatQuad  {  **private** **int** unitID;  **private** **int** posx;  **private** **int** posy;  **private** **int** energyLevel;  **private** **int** bulletCount;    **public** CombatQuad(**int** unitID)      {  **this**.unitID = unitID;  **this**.energyLevel = 100;  **this**.bulletCount = 20;      }        // ...More getters and setters...    **public** ISFSObject toSFSObject()      {          ISFSObject sfso = **new** SFSObject();            sfso.putByte("id", unitID);          sfso.putShort("px", posx);          sfso.putShort("py", posy);          sfso.putByte("el", energyLevel);          sfso.putShort("bc", bulletCount);    **return** sfso;      }  } |

Chúng tôi đã bỏ qua tất cả các getters/setters để đi thẳng vào vấn đề và chỉ ra cách thức hoạt động của nó. Phương thức **toSFSObject** đảm nhận việc trích xuất các thuộc tính chúng ta cần và cũng định dạng chúng bằng loại dữ liệu nhỏ nhất có thể. Mã mở rộng của chúng tôi sẽ được đơn giản hóa rất nhiều khi đến lúc gửi các dữ liệu đối tượng này trong một bản cập nhật:

Java

[**?**](http://docs2x.smartfoxserver.com/DevelopmentBasics/sfsobject-sfsarray)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | **public** **void** sendMapUpdate(CombatQuad quad, OtherObject other, User recipient)  {      ISFSObject responseObj = **new** SFSObject();      responseObj.putSFSObject("quad", quad.toSFSObject());      responseObj.putSFSObject("other", other.toSFSObject());        // Send data to client      send("quadUpdate", responseObj, recipient);  } |

Về phía máy khách, chúng tôi sẽ sử dụng một cách tiếp cận tương tự, mặc dù ngược lại: chúng tôi sẽ xây dựng lại thể hiện của lớp từ các thuộc tính được truyền qua SFSObject. Để làm như vậy, chúng tôi sẽ triển khai một hàm tạo tĩnh có tên **newFromSFSObject** .

C# JS (ES6) AS3

[**?**](http://docs2x.smartfoxserver.com/DevelopmentBasics/sfsobject-sfsarray)

|  |  |
| --- | --- |
| 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 | **public** **class** CombatQuad  {  **private** **int** unitID;  **private** **int** posx;  **private** **int** posy;  **private** **int** energyLevel;  **private** **int** bulletCount;    **public** **static** CombatQuad NewFromSFSObject(ISFSObject sfso)      {          CombatQuad combatQuad = **new** CombatQuad(sfso.GetByte("id"));            combatQuad.posx = sfso.GetShort("px");          combatQuad.posy = sfso.GetShort("py");          combatQuad.energyLevel = sfso.GetByte("el");          combatQuad.bulletCount = sfso.GetByte("bc");    **return** combatQuad;      }    **public** CombatQuad(**int** unitID)      {  **this**.unitID = unitID;      }        // ...More getters and setters...  } |

**LƯU Ý:** Chuỗi UTF-8/nhiều byte không được hỗ trợ trong tên khóa SFSObject. Nói cách khác, bạn nên giới hạn tên khóa đối với các ký tự ASCII tiêu chuẩn. Bạn cũng nên giữ tên khóa thật ngắn để tiết kiệm băng thông.

**» Chỉ AS3: Chuyển đổi Object/Array thành SFSObject/SFSArray**

Trong ActionScript 3, các lớp SFSObject và SFSArray cung cấp chuyển đổi trực tiếp sang và từ các kiểu gốc tương ứng của chúng: Đối tượng và Mảng. Điều này được giới thiệu để giúp di chuyển từ mã trước đó được viết cho SmartFoxServer 1.x và để cung cấp thêm một số tiện ích cho các nhà phát triển sử dụng các loại động.

Cần phải nhấn mạnh rằng việc sử dụng các Đối tượng chung chung thay vì các Lớp cho mô hình dữ liệu nói chung không được khuyến nghị, vì nó sẽ không tận dụng được các tối ưu hóa kiểu được giới thiệu trong ActionScript 3. Các thuộc tính được gắn động vào một Đối tượng không được- đã nhập và trình biên dịch không thể tối ưu hóa chúng.

Dù sao thì điều này có thể hữu ích trong một số trường hợp và nó có thể giúp đẩy nhanh quá trình phát triển. Các phương pháp bổ sung như sau.

* **SFSObject.newFromObject** : hàm tạo tĩnh, tạo một thể hiện SFSObject mới từ một Đối tượng. Các loại được hỗ trợ là: null, Boolean, Number, String, Object, Array. Hoạt động hỗ trợ các đối tượng lồng nhau.
* **SFSObject.toObject** : chuyển đổi thể hiện SFSObject thành một Đối tượng, bao gồm các đối tượng lồng nhau.
* **SFSArray.newFromArray** : hàm tạo tĩnh, tạo một thể hiện SFArray mới từ một Mảng. Các loại được hỗ trợ là: null, Boolean, Number, String, Object, Array. Hoạt động hỗ trợ các đối tượng lồng nhau.
* **SFSArray.toArray** : chuyển thể hiện SFSArray thành một Mảng, bao gồm các đối tượng lồng nhau.

Nhược điểm của việc sử dụng phương pháp này là bạn mất khả năng tinh chỉnh các loại số: tất cả các số sẽ được chuyển đổi thành số nguyên 32 bit (giá trị không phải số thập phân) hoặc gấp đôi 64 bit (giá trị thập phân).

**" Nhiêu tai nguyên hơn**

Để biết thêm chi tiết và ví dụ về cách sử dụng SFSObject và SFSArray, chúng tôi khuyên bạn nên xem qua các ví dụ được cung cấp với SFS2X và tham khảo tài liệu có trong phần **Tài liệu API** của trang web này. Ngoài ra, vui lòng kiểm tra hướng dẫn nâng cao về **[Tuần tự hóa lớp](http://docs2x.smartfoxserver.com/AdvancedTopics/class-serialization)** .

Tài liệu tham khảo :

[0] : “ <http://docs2x.smartfoxserver.com/DevelopmentBasics/sfsobject-sfsarray> ”