**Thành phần Hỗ trợ đăng nhập**

Hầu hết mọi ứng dụng đều yêu cầu quyền truy cập của người dùng dựa trên mật khẩu và việc viết mã xử lý các tham số máy khách và kiểm tra chúng dựa trên cơ sở dữ liệu là khá tẻ nhạt và lặp đi lặp lại.

Thành phần **Hỗ trợ đăng nhập** là một lớp trợ giúp hỗ trợ các nhà phát triển tạo hệ thống đăng nhập dựa trên cơ sở dữ liệu mà không gặp rắc rối khi viết mã truy cập cơ sở dữ liệu của riêng bạn.

Tóm lại, đây là cách nó hoạt động:

1. định cấu hình DBManager trong Vùng ứng dụng của bạn
2. khởi tạo thành phần trong phương thức init() của Tiện ích mở rộng của bạn
3. cấu hình thành phần

**» Triển khai thành phần Hỗ trợ đăng nhập**

Nếu bạn đang sử dụng SmartFoxServer 2X v2.7.0 trở lên, hãy bỏ qua bước này vì thành phần này đã có sẵn.   
Mặt khác, để kích hoạt thành phần trong bản cài đặt SmartFoxServer 2X của bạn, tất cả những gì bạn cần làm là:

* [**tải thành phần tại đây**](http://docs2x.smartfoxserver.com/ExtensionsJava/_files/LoginAssistant.zip)
* giải nén tệp jar được cung cấp bên trong thư mục **SFS2X/lib/ của bạn**
* khởi động lại SmartFoxServer 2X

**» Cách sử dụng cơ bản**

Trước khi bạn bắt đầu, hãy đảm bảo định cấu hình quyền truy cập cơ sở dữ liệu của bạn trong cấu hình Vùng. Để làm như vậy, hãy mở SFS2X **AdminTool** , khởi chạy mô-đun Bộ **cấu hình vùng và chọn** tab **Trình quản lý cơ sở dữ liệu .** Bạn có thể tìm thêm chi tiết về cách tiến hành trong **[công thức này](http://smartfoxserver.com/blog/how-to-setup-a-connection-to-an-external-database/" \t "_blank)** .

Đối với ví dụ này, chúng tôi sẽ sử dụng một bảng *người dùng đơn giản* với các trường sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **name** | **pword** | **email** |
| **1** | Kermit | thefrog | kermit@muppets.com |
| **2** | Gonzo | thegreat | gonzo@muppets.com |
| **3** | Fozzie | thebear | fozzie@muppets.com |
| **4** | MissPiggy | thepig | piggy@muppets.com |

Bước tiếp theo, chúng tôi tạo một Tiện ích mở rộng mới và sử dụng mã này trong phương thức **init()** :

Java

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/login-assistant)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | **public** **class** MuppetsExtension **extends** SFSExtension  {  **private** LoginAssistantComponent lac;        @Override  **public** **void** init()      {          lac = **new** LoginAssistantComponent(**this**);            // Configure the component          lac.getConfig().loginTable = "muppets";          lac.getConfig().userNameField = "name";          lac.getConfig().passwordField = "pword";      }    **public** **void** destroy()      {          lac.destroy();      }  } |

Tất cả những gì chúng tôi đã làm chỉ đơn giản là định cấu hình thành phần bằng cách chỉ định tên của bảng sẽ được sử dụng và tên của các trường đại diện cho tên người dùng và mật khẩu. Thì đấy! Thành phần hiện được định cấu hình để sử dụng các trường chính xác và sẽ đảm nhiệm việc kiểm tra thông tin đăng nhập, gửi lỗi và làm sạch dữ liệu người dùng trước các lần tiêm SQL.

**» Kịch bản nâng cao hơn một chút**

Giả sử chúng ta có một tình huống phức tạp hơn một chút khi người dùng sẽ đăng nhập bằng địa chỉ email nhưng tên người dùng của họ trong hệ thống phải được liên kết với trường **tên** trong cơ sở dữ liệu. Ngoài ra, chúng tôi muốn đảm bảo rằng kiểm tra tên người dùng có **phân biệt chữ hoa chữ thường** .

Trên thực tế, theo mặc định, hầu hết các cơ sở dữ liệu sẽ so sánh các chuỗi theo cách không phân biệt chữ hoa chữ thường, trừ khi sử dụng một *đối chiếu cụ thể* . Hỗ trợ đăng nhập có thể được định cấu hình để sử dụng một trường cụ thể trong DB làm tên đăng nhập, hãy xem cách thực hiện việc này:

Java

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/login-assistant)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | @Override  **public** **void** init()  {      lac = **new** LoginAssistantComponent(**this**);        // Configure the component      lac.getConfig().loginTable = "muppets";      lac.getConfig().userNameField = "email";      lac.getConfig().passwordField = "pword";        lac.getConfig().nickNameField = "name";      lac.getConfig().useCaseSensitiveNameChecks = **true**;  } |

Tất cả những gì chúng tôi đã làm chỉ đơn giản là định cấu hình thành phần bằng cách chỉ định tên của bảng sẽ được sử dụng và tên của các trường đại diện cho tên người dùng và mật khẩu. Thì đấy! Thành phần hiện được định cấu hình để sử dụng các trường chính xác và sẽ đảm nhiệm việc kiểm tra thông tin đăng nhập, gửi lỗi và làm sạch dữ liệu người dùng trước các lần tiêm SQL.

**» Kịch bản nâng cao hơn một chút**

Giả sử chúng ta có một tình huống phức tạp hơn một chút khi người dùng sẽ đăng nhập bằng địa chỉ email nhưng tên người dùng của họ trong hệ thống phải được liên kết với trường **tên** trong cơ sở dữ liệu. Ngoài ra, chúng tôi muốn đảm bảo rằng kiểm tra tên người dùng có **phân biệt chữ hoa chữ thường** .

Trên thực tế, theo mặc định, hầu hết các cơ sở dữ liệu sẽ so sánh các chuỗi theo cách không phân biệt chữ hoa chữ thường, trừ khi sử dụng một *đối chiếu cụ thể* . Hỗ trợ đăng nhập có thể được định cấu hình để sử dụng một trường cụ thể trong DB làm tên đăng nhập, hãy xem cách thực hiện việc này:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **name** | **pword** | **email** | **avatar** | **isMod** |
| **1** | Kermit | thefrog | kermit@muppets.com | kermit.jpg | Y |
| **2** | Gonzo | thegreat | gonzo@muppets.com | gonzo.jpg | N |
| **3** | Fozzie | thebear | fozzie@muppets.com | fozzie.png | Y |
| **4** | Miss Piggy | thepig | piggy@muppets.com | piggy.png | N |

Chúng tôi sẽ lưu trữ tên hình ảnh bên trong đối tượng **Phiên của khách hàng** , để khi chúng tôi nhận được sự kiện **USER\_JOIN\_ZONE** , chúng tôi sẽ có thể đặt hình ảnh thông qua **Biến người dùng** . Ngoài ra, chúng tôi muốn có thể đặt máy khách làm người điều hành, nếu cờ cơ sở dữ liệu được đặt.

Hỗ **trợ đăng nhập** cung cấp một cách dễ dàng để thêm logic tùy chỉnh trước và sau khi kiểm tra thông tin đăng nhập, thông qua các lớp **preProcessPlugin** và **postProcessPlugin** . Hãy xem cách nó hoạt động:

Java

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/login-assistant)

|  |  |
| --- | --- |
| 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 | @Override  **public** **void** init()  {      lac = **new** LoginAssistantComponent(**this**);        // Configure the component      lac.getConfig().loginTable = "muppets";      lac.getConfig().userNameField = "email";      lac.getConfig().passwordField = "pword";      lac.getConfig().nickNameField = "name";      lac.getConfig().useCaseSensitiveNameChecks = **true**;        lac.getConfig().extraFields = Arrays.asList("avatar", "isMod");        lac.getConfig().postProcessPlugin = **new** ILoginAssistantPlugin ()      {  **public** **void** execute(LoginData loginData)          {              ISFSObject fields = loginData.extraFields;                String avatarPic = fields.getUtfString("avatar");  **boolean** isMod = fields.getUtfString("isMod").equalsIgnoreCase("Y");                // Store avatar in session object              loginData.session.setProperty("avatar", avatarPic)                // Set client as Moderator  **if** (isMod)                  loginData.session.setProperty("$permission", DefaultPermissionProfile.MODERATOR);          }      };  } |

Chúng tôi đã thêm một tham số cấu hình mới có tên là **extraFields,** chứa danh sách các tên trường tùy chỉnh mà chúng tôi muốn chọn khi đăng nhập.

Trong dòng tiếp theo, chúng tôi đã chuyển lớp **postProcessPlugin** bằng cách khai báo nó nội tuyến, dưới dạng một lớp ẩn danh để thuận tiện. Lớp này được yêu cầu triển khai giao diện **ILoginAssistantPlugin khai báo phương thức exec ()** .

Như bạn có thể thấy trong phương thức này, chúng tôi được chuyển tất cả dữ liệu đăng nhập có liên quan, bao gồm tất cả các trường được trích xuất từ cơ sở dữ liệu (dưới dạng SFSObject) và Phiên của máy khách. Bằng cách này, rất đơn giản để áp dụng logic bổ sung cho quy trình đăng nhập mà không cần phải viết trình xử lý Đăng nhập tùy chỉnh.

Bạn có thể tìm thấy tất cả chi tiết về các lớp được sử dụng trong ví dụ này bằng cách tham khảo **[javadoc phía máy chủ](http://docs2x.smartfoxserver.com/api-docs/javadoc/server/" \t "_blank)** trong gói *com.smartfoxserver.v2.components.login* .

**Kể từ 2.10.0**

**» Kiểm tra và mã hóa mật khẩu tùy chỉnh**

Kể từ **SmartFoxServer 2X 2.10** , chúng tôi đã thêm mã hóa TLS vào giao thức mở ra khả năng xử lý tùy chỉnh kiểm tra mật khẩu.

Trước khi mã hóa giao thức (phiên bản < 2.10), chỉ mật khẩu được mã hóa trong LoginRequest bằng **[Giao thức xác thực bắt tay thử thách](http://en.wikipedia.org/wiki/Challenge-Handshake_Authentication_Protocol" \t "_blank)** (CHAP), bằng cách sử dụng thuật toán băm. Điều này làm cho mật khẩu rất an toàn trong quá trình truyền nhưng không cho phép nhà phát triển sử dụng muối mật khẩu để lưu trữ trong cơ sở dữ liệu.

Với mã hóa giao thức, mật khẩu có thể được gửi mà không cần băm, bỏ qua CHAP và cho phép Tiện ích mở rộng máy chủ nhận mật khẩu ban đầu, mật khẩu này sau đó có thể được chọn và lưu trữ như vậy trong cơ sở dữ liệu.

Để bỏ qua muối, bạn cần sử dụng **SmartFoxServer 2X 2.10** (hoặc cao hơn) và đặt cờ customPasswordCheck của **LoginAssistant** thành true trong cấu hình.

Java

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/login-assistant)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | @Override  **public** **void** init()  {      lac = **new** LoginAssistantComponent(**this**);        // Configure the component      lac.getConfig().loginTable = "muppets";        ...        lac.getConfig().customPasswordCheck = **true**;  } |

Chúng tôi đã thêm một tham số cấu hình mới có tên là **extraFields,** chứa danh sách các tên trường tùy chỉnh mà chúng tôi muốn chọn khi đăng nhập.

Trong dòng tiếp theo, chúng tôi đã chuyển lớp **postProcessPlugin** bằng cách khai báo nó nội tuyến, dưới dạng một lớp ẩn danh để thuận tiện. Lớp này được yêu cầu triển khai giao diện **ILoginAssistantPlugin khai báo phương thức exec ()** .

Như bạn có thể thấy trong phương thức này, chúng tôi được chuyển tất cả dữ liệu đăng nhập có liên quan, bao gồm tất cả các trường được trích xuất từ cơ sở dữ liệu (dưới dạng SFSObject) và Phiên của máy khách. Bằng cách này, rất đơn giản để áp dụng logic bổ sung cho quy trình đăng nhập mà không cần phải viết trình xử lý Đăng nhập tùy chỉnh.

Bạn có thể tìm thấy tất cả chi tiết về các lớp được sử dụng trong ví dụ này bằng cách tham khảo **[javadoc phía máy chủ](http://docs2x.smartfoxserver.com/api-docs/javadoc/server/" \t "_blank)** trong gói *com.smartfoxserver.v2.components.login* .

**Kể từ 2.10.0**

**» Kiểm tra và mã hóa mật khẩu tùy chỉnh**

Kể từ **SmartFoxServer 2X 2.10** , chúng tôi đã thêm mã hóa TLS vào giao thức mở ra khả năng xử lý tùy chỉnh kiểm tra mật khẩu.

Trước khi mã hóa giao thức (phiên bản < 2.10), chỉ mật khẩu được mã hóa trong LoginRequest bằng **[Giao thức xác thực bắt tay thử thách](http://en.wikipedia.org/wiki/Challenge-Handshake_Authentication_Protocol" \t "_blank)** (CHAP), bằng cách sử dụng thuật toán băm. Điều này làm cho mật khẩu rất an toàn trong quá trình truyền nhưng không cho phép nhà phát triển sử dụng muối mật khẩu để lưu trữ trong cơ sở dữ liệu.

Với mã hóa giao thức, mật khẩu có thể được gửi mà không cần băm, bỏ qua CHAP và cho phép Tiện ích mở rộng máy chủ nhận mật khẩu ban đầu, mật khẩu này sau đó có thể được chọn và lưu trữ như vậy trong cơ sở dữ liệu.

Để bỏ qua muối, bạn cần sử dụng **SmartFoxServer 2X 2.10** (hoặc cao hơn) và đặt cờ customPasswordCheck của **LoginAssistant** thành true trong cấu hình.

C# JS AS3

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/login-assistant)

|  |  |
| --- | --- |
| 1  2  3  4 | ISFSObject sfso = SFSObject.NewInstance();  sfso.PutUtfString("passwd", tf\_password.text);    sfs.Send(**new** Sfs2X.Requests.LoginRequest(ti\_userName.text, "", sfs.Config.Zone, sfso)); |

Điều quan trọng cần lưu ý là chúng tôi đang gửi mật khẩu **dưới dạng trường tùy chỉnh** của SFSObject tùy chọn trong **LoginRequest** . Tên trường có thể là bất kỳ mã định danh nào miễn là cùng một chuỗi được sử dụng ở cả phía máy khách và máy chủ.

Về phía máy chủ, chúng tôi phải quan tâm đến việc xác thực mật khẩu trong plugin **PreProcess** :

[**?**](http://docs2x.smartfoxserver.com/ExtensionsJava/login-assistant)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | **class** LoginPreProcess **implements** ILoginAssistantPlugin  {      @Override  **public** **void** execute(LoginData ld)      {          String clientPass = ld.clientIncomingData.getUtfString("passwd");            // Let's see if the password from the DB matches that of the user  **if** (!ld.password.equals(clientpass))  **throw** **new** PasswordCheckException();            // Success!      }  } |

Việc này thật thẳng thắn. Chúng tôi lấy mật khẩu bằng tên khóa mà chúng tôi đã chọn, từ SFSObject tùy chọn và kiểm tra mật khẩu đó với mật khẩu từ DB. Nếu chúng không khớp, chúng ta chỉ cần tăng **PasswordCheckException** để làm gián đoạn luồng. Đến lượt nó, điều này sẽ thông báo cho khách hàng về việc sử dụng sự kiện LOGIN\_ERROR thông thường.

Ngược lại, nếu điều kiện được thỏa mãn, chúng tôi đã hoàn thành. Luồng LoginAssistant sẽ tiếp tục với phần logic còn lại và khách hàng sẽ đăng nhập.

**LƯU Ý**   
Nếu bạn định sử dụng phương thức mới này, chúng tôi thực sự khuyên bạn nên kích hoạt cờ **useEncryption** trong cấu hình Vùng của bạn (AdminTool > Trình cấu hình Vùng) để đảm bảo rằng máy chủ sẽ chỉ chấp nhận các kết nối được mã hóa, do đó ẩn thông tin liên lạc khỏi những kẻ nghe trộm có thể có.

**» Xử lý lỗi**

Các lỗi liên quan đến thông tin đăng nhập sai, quyền truy cập bị cấm, tên người dùng trùng lặp, v.v. đều được xử lý bởi thành phần **Hỗ trợ đăng nhập** bằng cách sử dụng các quy tắc SmartFoxServer 2X tiêu chuẩn. Điều này có nghĩa là tất cả các thông báo lỗi có thể được tùy chỉnh ở phía máy khách thông qua [**lớp SFSErrorCodes**](http://docs2x.smartfoxserver.com/AdvancedTopics/client-error-messages) .

Nơi duy nhất mà nhà phát triển chịu trách nhiệm xử lý các trường hợp ngoại lệ là trong các plugin **tiền xử lý** và **hậu xử lý .** Bất kỳ ngoại lệ thời gian chạy nào trong các lớp đó sẽ di chuyển lên đầu chuỗi API SFS2X và được ghi lại và ghi lại ở đó. Điều này cũng có nghĩa là chúng sẽ **làm gián đoạn quy trình đăng nhập** , do đó bạn có quyền quyết định ngoại lệ nào cần được bắt (để không làm gián đoạn quy trình) và ngoại lệ nào có thể dừng quy trình.