Link : “ http://docs2x.smartfoxserver.com/AdvancedTopics/room-persistence-api ”

**» API duy trì phòng**

Trong **SmartFoxServer 2X 2.8.0** , chúng tôi đã giới thiệu một API duy trì phòng mới để giúp lưu trữ và truy xuất trạng thái của Phòng. Tính năng này có thể đặc biệt hữu ích để lưu trữ trạng thái của trò chơi đang diễn ra, Phòng cá nhân trong thế giới ảo, v.v.

**" Tổng quan**

Room Persistence API được phác thảo trong giao diện **IRoomStorage cơ bản được cung cấp trong gói** *com.smartfoxserver.v2.persistence.room* :

**IRoomStorage** :

* saveRoom(Phòng theRoom)
* saveAllRooms(String groupId)
* saveAllRooms()
* loadRoom(Tên chuỗi)
* loadAllRooms(String groupId)
* tảiAllRooms()
* removeRoom(Tên chuỗi)
* removeAllRooms(String groupId)
* removeAllRooms()

Mỗi nhóm phương thức cho phép làm việc với một Phòng cụ thể, một nhóm Phòng hoặc toàn bộ danh sách Phòng bên trong Vùng. Các phòng được lưu trữ bằng cách tuần tự hóa tất cả các thuộc tính của chúng (đối tượng CreateRoomSettings) và, tùy chọn, tất cả các Biến phòng của chúng.

**LƯU Ý:** khi lưu trữ **Biến phòng** , hãy nhớ rằng chỉ các biến do máy chủ sở hữu mới được lưu giữ. Điều này là do các biến do Người dùng sở hữu không thể được tạo lại sau này nếu không có sự hiện diện của chủ sở hữu Biến đó trong Phòng.

**» Triển khai**

Chúng tôi cung cấp hai cách triển khai khác nhau ngay lập tức:

* **Lưu trữ dựa trên hệ thống tệp** : cho phép lưu trữ dữ liệu Phòng trong hệ thống tệp cục bộ, theo mặc định trong thư mục data/roomData/. Đường dẫn mặc định có thể được cấu hình lại để trỏ đến bất kỳ thư mục nào khác. Xin lưu ý rằng việc sử dụng ổ đĩa được chia sẻ qua mạng có thể làm chậm đáng kể API lưu trữ.
* **Lưu trữ cơ sở dữ liệu** : sử dụng DBManager mặc định của Vùng để lưu trữ dữ liệu Phòng trong cơ sở dữ liệu bên ngoài. Nó cũng cho phép cung cấp một DBManager tùy chỉnh và tùy chỉnh một số tùy chọn khác.

**" Bắt đầu nhanh**

Mỗi Vùng máy chủ chạy một phiên bản khác nhau của Persistence API, vì vậy chúng ta có thể tùy chỉnh cài đặt Persistence cho từng Ứng dụng một cách độc lập.

Đoạn mã sau đây cho biết cách khởi tạo Room Persistence API trong Tiện ích mở rộng vùng của chúng tôi:

Java

[**?**](http://docs2x.smartfoxserver.com/AdvancedTopics/room-persistence-api)

|  |  |
| --- | --- |
| 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 | // Init method of a Zone Extension  **public** **void** init()  {      // Initialize Persistence API      getParentZone().initRoomPersistence(RoomStorageMode.FILE\_STORAGE, **new** FileRoomStorageConfig());        // Load all previously stored Rooms  **try**      {          List<CreateRoomSettings> allRooms = getParentZone().getRoomPersistenceApi().loadAllRooms();            // Recreate all Rooms  **for** (CreateRoomSettings settings : allRooms)          {              getApi().createRoom(getParentZone(), settings, **null**, **false**, **null**, **false**, **false**);          }      }  **catch** (SFSStorageException storageErr)      {          trace("Error loading rooms: " + storageErr);      }  **catch** (SFSCreateRoomException creationErr)      {          trace("Error creating room: "  + creationErr);      }  } |

Chúng tôi khởi tạo API bằng cách cung cấp *Chế độ lưu trữ phòng* ( **FILE\_STORAGE** hoặc **DB\_STORAGE** ) và phiên bản tương đối của lớp cấu hình ( **FileRoomStorageConfig** , **DBRoomStorageConfig** ). Sau đó, chúng tôi tiến hành tải tất cả các Phòng đã lưu trữ trước đó và tạo lại chúng trong hệ thống bằng phương thức SFSApi.createRoom(...) .

Khi khởi tạo API RoomPersistence, bạn cũng có thể muốn định cấu hình một số cài đặt:

* **storeInactiveRooms** : khi true nó sẽ lưu trữ cả những Phòng không hoạt động, tức là Room.isActive() == false. Cài đặt mặc định là **sai** .
* **storeRoomVariables** : khi đúng, nó sẽ lưu trữ tất cả các Biến phòng do máy chủ sở hữu cùng với các cài đặt Phòng khác. Cài đặt mặc định là **đúng** .
* **skipStaticRooms** : khi đúng nó sẽ đảm bảo không lưu các Phòng tĩnh. Cài đặt mặc định là **đúng** .

**LƯU Ý:** Phòng tĩnh là những phòng được xác định trong **Công cụ quản trị > Bộ cấu hình vùng** . Các Phòng này đã được tạo lại mỗi khi Máy chủ khởi động. Nếu bạn lưu trữ chúng qua Persistence API và sau đó cố gắng tạo lại chúng, bạn sẽ gặp lỗi vì các Phòng đó đã tồn tại. Thông thường, bạn không cần phải duy trì Phòng tĩnh.

Mỗi triển khai kiên trì cũng cung cấp một số cài đặt tùy chỉnh, ví dụ: triển khai dựa trên cơ sở dữ liệu cho phép chỉ định tên bảng, Trình quản lý quản lý cơ sở dữ liệu tùy chỉnh, v.v... Bạn có thể đọc tất cả các chi tiết trong **[javadoc API phía máy chủ](http://docs2x.smartfoxserver.com/api-docs/javadoc/server/" \t "_blank)** , bên dưới *com.smartfoxserver Gói .v2.persistence.room* .

**» Ví dụ về tính kiên trì của cơ sở dữ liệu**

[**?**](http://docs2x.smartfoxserver.com/AdvancedTopics/room-persistence-api)

|  |  |
| --- | --- |
| 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 | // Init method of a Zone Extension  **public** **void** init()  {      DBRoomStorageConfig cfg = **new** DBRoomStorageConfig();      cfg.tableName = "my\_game\_room\_data"        // Initialize Persistence API      getParentZone().initRoomPersistence(RoomStorageMode.DB\_STORAGE, cfg);        // Load all previously stored Rooms  **try**      {          List<CreateRoomSettings> allRooms = getParentZone().getRoomPersistenceApi().loadAllRooms();            // Recreate all Rooms  **for** (CreateRoomSettings settings : allRooms)          {              getApi().createRoom(getParentZone(), settings, **null**, **false**, **null**, **false**, **false**);          }      }  **catch** (SFSStorageException storageErr)      {          trace("Error loading rooms: " + storageErr);      }  **catch** (SFSCreateRoomException creationErr)      {          trace("Error creating room: "  + creationErr);      }  } |

Sự khác biệt duy nhất với ví dụ dựa trên tệp là việc sử dụng **DBRoomStorageConfig** để định cấu hình tha API. Ở đây chúng ta cần xác định ít nhất tên bảng mà chúng ta muốn sử dụng để lưu trữ. Liên quan đến chi tiết kết nối cơ sở dữ liệu, API sẽ sử dụng **Trình quản lý cơ sở dữ liệu của Vùng** được định cấu hình từ Trình cấu hình vùng **[của AdminTool](http://docs2x.smartfoxserver.com/GettingStarted/admintool-ZoneConfigurator)** .

Nếu bạn muốn xác định một kết nối riêng cho tính bền bỉ của API, bạn cũng có thể làm như vậy bằng cách chỉ định kết nối đó trong đối tượng **DBRoomStorageConfig** :

[**?**](http://docs2x.smartfoxserver.com/AdvancedTopics/room-persistence-api)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | **public** **void** init()  {      DBConfig cfg = **new** DBConfig();      cfg.active = **true**;      cfg.driverName = "com.mysql.jdbc.Driver";      cfg.connectionString = "jdbc:[**mysql://127.0.0.1/database\_name**](mysql://127.0.0.1/database_name)";      cfg.userName = "db\_user\_name";      cfg.password = "db\_user\_pass";      cfg.testSql = "SELECT name FROM some\_table LIMIT 1";        DBRoomStorageConfig cfg = **new** DBRoomStorageConfig();      cfg.tableName = "my\_game\_room\_data"      cfg.dbManager = **new** SFSDBManager(cfg);    } |

**" Đê nghị sử dụng**

Dưới đây, chúng tôi thảo luận về hai trường hợp sử dụng khác nhau cho **Room Persistence API** và cung cấp một số mẹo về cách đạt được hiệu suất tốt nhất.

**»Lưu và tải trạng thái trò chơi cho các trận đấu kéo dài**

Tình huống đầu tiên là một ván cờ mà trận đấu có thể kéo dài hàng ngày, hàng tuần hoặc thậm chí hàng tháng. Đối thủ sẽ có thể kết nối với máy chủ và khôi phục trò chơi từ nơi họ đã rời khỏi kể từ lần di chuyển cuối cùng. Để thực hiện điều này, chúng ta sẽ cần:

* Tên phòng duy nhất cho mỗi trận đấu trong trò chơi Cờ vua. Điều này rất quan trọng để tránh xung đột giữa các tên Phòng, do đó tạo ra lỗi khi cố gắng tạo một Phòng đã tồn tại.
* Trạng thái trò chơi sẽ được lưu trữ trong một số RoomVariables sẽ được duy trì khi người chơi rời đi.

Với suy nghĩ này, chúng tôi sẽ có thể dễ dàng triển khai một trò chơi cho phép người chơi đóng băng trận đấu và quay lại vào thời điểm khác để tiếp tục từ nơi họ đã rời đi.

Tạo một tên Phòng duy nhất có thể được xử lý theo nhiều cách khác nhau. Ý tưởng đơn giản nhất là xác định Phòng theo tên của hai đối thủ. Vì tất cả tên người dùng là duy nhất trong một Khu vực, nên việc kết hợp hai tên duy nhất sẽ cho chúng ta một tên Phòng duy nhất cho mỗi trò chơi. Ví dụ: chúng ta có thể sử dụng quy ước như *<Player1>\_vs\_<Player2>* cho tên Phòng.

Tiếp theo, chúng tôi cần một cách để người chơi nói với hệ thống rằng họ muốn tạm dừng trò chơi. Điều này có thể được thực hiện theo nhiều cách:

* Chỉ cần cho phép người chơi rời khỏi trò chơi bất cứ lúc nào, nắm bắt sự kiện từ phía Máy chủ (sự kiện ROOM\_REMOVED) và gọi API Persistence đi qua Phòng cụ thể. Điều này có thể hữu ích để đảm bảo rằng trò chơi được lưu trữ ngay cả khi người chơi vô tình rời khỏi trò chơi (tức là bị ngắt kết nối).   
  **LƯU Ý:** trong trường hợp này, bạn cần đảm bảo rằng các Phòng không hoạt động đã được lưu. Phòng có được thông qua sự kiện sau khi bị xóa sẽ không hoạt động.
* Cho phép người chơi bày tỏ ý muốn rời khỏi trò chơi bằng cách thêm một yêu cầu cụ thể trong ứng dụng khách. Khi máy chủ nhận được yêu cầu của cả hai người chơi, Mã mở rộng có thể lưu trạng thái Phòng và xóa Phòng khỏi hệ thống.

Để khôi phục trò chơi khi Người dùng quay lại, cũng có nhiều tùy chọn. Ý tưởng tốt nhất là lưu trữ tên Phòng trò chơi đang diễn ra trong chính hồ sơ Người dùng để khi người chơi đăng nhập lại, hệ thống có thể ngay lập tức tạo lại trò chơi bị treo.

**» Lưu và tải trạng thái Phòng cho các Phòng cá nhân.**

Một đặc điểm chung của thế giới ảo trực tuyến là cung cấp cho mỗi Người dùng một "ngôi nhà" tùy chỉnh mà họ có thể tùy chỉnh và định cấu hình với đủ loại đồ trang trí và vật dụng cá nhân.

Sử dụng **Room Persistence API** có thể là cách dễ dàng nhất để lưu và tải các Phòng này theo yêu cầu khi người chơi hoạt động trong hệ thống. Các yêu cầu cơ bản cho chức năng này rất giống với trường hợp sử dụng trước đó:

* Đảm bảo rằng chúng tôi chỉ định một tên Phòng duy nhất cho nhà riêng của mỗi Người dùng
* Người dùng một hoặc một số Biến phòng để lưu trữ tất cả cài đặt và tùy chỉnh của Người dùng

Một lần nữa, chúng ta có thể tận dụng tính duy nhất của tên Người dùng để tạo tên Phòng cá nhân riêng biệt. Một cái gì đó đơn giản như *Home\_<PlayerName>* sẽ thực hiện thủ thuật. Tiếp theo, chúng ta cần tạo các Biến phòng để theo dõi các đồ trang trí khác nhau của Người dùng.

Nếu số lượng tùy chỉnh rất cao (giả sử hơn 30 tham số), chúng ta có thể tận dụng sự tiện lợi của SFSObject thay vì sử dụng một cách tiếp cận RoomVariable-per-setting. Ví dụ: chúng ta có thể thu gọn tất cả các cài đặt cho nội thất của Phòng trong một SFSObject và tất cả các chi tiết cá nhân trong một SFSObject khác, v.v... và cuối cùng lưu trữ các SFSObject này dưới dạng Biến phòng.

Để lưu trữ và truy xuất các Phòng cá nhân này theo yêu cầu, chúng ta chỉ cần dựa vào các sự kiện đăng nhập và ngắt kết nối của người dùng. Mỗi khi khách hàng tham gia hệ thống, chúng tôi có thể tải dữ liệu Phòng cá nhân của họ và tạo Phòng trong hệ thống. Tương tự, khi người dùng rời khỏi hệ thống bằng cách ngắt kết nối, chúng tôi có thể lưu trữ Phòng và loại bỏ nó cho đến lần đăng nhập tiếp theo.

**» Loại trừ các biến Phòng cụ thể**

Vì **SFS2X 2.14** RoomVariables hiển thị một cờ mới có tên là **có thể lưu trữ** có thể được sử dụng để loại trừ một cách có chọn lọc các Biến cụ thể khỏi việc lưu bởi API Room Persistence.

Cờ được đặt thành **true** theo mặc định trong mọi RoomVariable, nhưng bạn có thể tắt cờ này tại thời điểm tạo để báo hiệu những Biến nào sẽ bị API Persistence bỏ qua.

Ví dụ:

Java

[**?**](http://docs2x.smartfoxserver.com/AdvancedTopics/room-persistence-api)

|  |  |
| --- | --- |
| 1  2 | RoomVariable rv = **new** SFSRoomVariable("TempName", "Sir No Name");  rv.setStorable(**false**); |

**» Lưu trữ phòng tùy chỉnh với SFS2X 2.18**

Kể từ khi phát hành **SmartFoxServer 2X 2.18.0,** chúng tôi đã giới thiệu một cách ghi đè các triển khai mặc định cho **API Room Persistence** để chúng có thể được tùy chỉnh theo nhu cầu của bạn.

[**Bạn có thể đọc tất cả các chi tiết trong bài viết này**](https://smartfoxserver.com/blog/custom-room-storage-with-sfs2x-2-18/) .