Hello, XMPP!

让我们开始Lithosphere IoT平台的学习！

在这第一篇教程中，我们来了解Lithosphere IoT平台提供的一些基础技术设施。通过这篇教程的学习，我们会初步接触了解Granite XMPP Server和Chalk客户端XMPP库。

我们编写一个简单的应用，来熟悉Chalk和Granite的使用，学习如何通过编写插件来扩展Lithosphere平台功能。

1 前置条件：

Java >= 11
Granite Lite Mini XMPP Server

点击这里下载Granite Lite Mini XMPP Server

2 检查Granite Lite Mini XMPP Server

2.1 解压granite-lite-mini-server

```
unzip granite-lite-mini-1.0.3-RELEASE.zip
```

2.2 启动Granite Lite XMPP Server

2.2.1 配置Domain

根据XMPP规范要求，每个XMPP Server必须指定Domain。
用户可以在${GRANITE_LITE_SERVER_HOME}/configuration/server.ini文件中配置Domain。

在server.ini文件中，找到domain.name的配置行

```
domain.name=localhost
```

这一行修改为你服务器的域名：

```
domain.name=${MY_XMPP_SERVER_DOMAIN_NAME}
```

如果你是在局域网内启动Granite XMPP Server，请使用你当前电脑的IP作为域名。假设你的电脑IP是192.168.1.80，修改后的配置应为：

```
domain.name=192.168.1.80
```

2.2.2 启动并检查Granite Lite XMPP Server的状态

启动Granite Lite XMPP Server

```
cd granite-lite-mini-1.0.3
java -jar granite-server-1.0.3-RELEASE.jar -console
```

带-console参数启动Granite Lite XMPP Serverw之后，能够看到Granite Server Console的界面。 我们可以在Console输入services命令来检查Granite XMPP Server的状态。

```
$services
```

如果能看到所有的services的状态都是available，说明granite lite server已经被正常的启动了。 可以用plugins命令，来检查可用的plugins。

```
$plugins
```

我们会看到，当前的服务器为最小部署版本，部署了最基本的5个插件：

- granite-lite-auth
- granite-lite-dba
- granite-lite-pipeline
- granite-lite-session
- granite-stream-standard

我们用exit指令，可以退出Granite Server Console，并关闭Granite XMPP Server。

```
$exit
```

3 编写第一个插件

XMPP协议基于典型的C/S架构模式，客户端需要一个服务器上的账号，才能登录到服务器进行通讯。

如何在Granite XMPP Server上创建一个用户呢？

Granite XMPP Server完全基于插件架构。我们可以通过编写插件，扩展Granite Server Console的功能，来帮助实现创建用户的任务。

3.1 创建hello-xmpp-server工程

我们创建一个maven工程。创建hello-xmpp-server目录，在目录下，建立pom.xml文件。

pom.xml的内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://maven.apache.org/POM/4.0.0"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>com.thefirstlineofcode.granite</groupId>
        <artifactId>com.thefirstlineofcode.granite</artifactId>
        <version>1.0.3-RELEASE</version>
    </parent>

    <groupId>com.thefirstlineofcode.lithosphere.tutorials.helloxmpp</groupId>
    <artifactId>hello-xmpp-server</artifactId>
    <version>0.0.1-RELEASE</version>
    <name>Hello XMPP server plugin</name>

    <dependencies>
        <dependency>
            <groupId>com.thefirstlineofcode.granite.framework</groupId>
            <artifactId>granite-framework-core</artifactId>
        </dependency>
    </dependencies>

    <repositories>
        <repository>
            <id>com.thefirstlineofcode.releases</id>
            <name>TheFirstLineOfCode Repository - Releases</name>
            <url>http://120.25.166.188:9090/repository/maven-releases/</url>
        </repository>
    </repositories>
</project>
```

代码说明

- 我们引用com.thefirstlineofcode.granite:com.thefirstlineofcode.granite作为parent POM，这样可以简化我们pom文件的build配置，例如插件配置和依赖库版本配置。

> - 目前，Lithosphere的开源库，仅被部署在TheFirstLineOfCode的私有的maven服务器上。为了构建时能够正确找到开源依赖库，需要配置com.thefirstlineofcode.releases的repository。
>
> - hello-xmpp-server插件依赖com.thefirstlineofcode.granite.framework:granite-framework-core包，因为我们需要用到granite framework库里的ICommandProcessor扩展点来扩展granite server console功能。

3.2 编写插件的代码

我们创建一个名为HelloXmppCommandsProcessor的类，它继承AbstractCommandsProcessor类。