为项目编写Readme.MD文件

git Git

更多 0

了解一个项目，恐怕首先都是通过其Readme文件了解信息。如果你以为Readme文件都是随便写写的那你就错了。github，oschina git gitcafe的代码托管平台上的项目的Readme.MD文件都是有其特有的语法的。称之为Markdown语法。基本规则如下：

> Markdown 语法速查表
> 1 标题与文字格式
> 标题
> # 这是 H1 <一级标题>
> ## 这是 H2 <二级标题>
> ###### 这是 H6 <六级标题>
> 文字格式
> **这是文字粗体格式**
> *这是文字斜体格式*
> ~~在文字上添加删除线~~
> 2 列表
> 无序列表
> * 项目1
> * 项目2
> * 项目3
> 有序列表
> 1. 项目1

2. 项目2
3. 项目3
   * 项目1
   * 项目2

3 其它

图片

```
![图片名称](http://gitcafe.com/image.png)
```

链接

```
[链接名称](http://gitcafe.com)
```

引用

```
> 第一行引用文字
> 第二行引用文字
```

水平线

```
***
```

代码

```
`<hello world>`
```

代码块高亮

```
```ruby
 def add(a, b)
   return a + b
 end
```
```

表格

```
 表头 | 表头
 ---------- | ----------
 单元格内容 | 单元格内容
 单元格内容l | 单元格内容
```


最近更新博客

- Antirez weblog
- Lunar的oracle实验室
- the5fire的技术博客
- the5fire的技术博客
- 刻骨铭心
- 程序师
- Java译站
- Coding 4 Fun
- 四火的唠叨
- Ahoo

如果直接记语法，那似乎困难了些。这里OneCoder推荐两个Markdown的编辑器。

在线编辑器：stackedit

网址：https://stackedit.io/

Mac下离线编辑器Mou

下载地址：http://mouapp.com/

OneCoder这里使用的是后者为自己的shurnim-storage项目写Readme。至于这个项目是什么，见Readme文档，OneCoder也会在另外的博文做一些补充说明。成品Readme如下：

```
# shurnim-storage

![Shurnim icon](http://pic.yupoo.com/8wuliao/DLPii2Jx/rEBO.jpg)

## 目录
* [背景介绍](#背景介绍)
* [项目介绍](#项目介绍)
* [使用说明](#使用说明)
  * [获取代码](#获取代码)
  * [开发插件](#开发插件)
  * [使用ShurnimStorage接口](#使用ShurnimStorage接口)
      * [接口介绍](#接口介绍)
      * [使用样例](#使用样例)
* [其他](#其他)

<a name="背景介绍"></a>
## 背景介绍

*Shurnim*，是我和我老婆曾经养过的一只仓鼠的名字。<br/>
*shurnim-storage*，是一个插件式云存储/网盘同步管理工具。是在参加又拍云开发大赛的过程中设计并开发。

<a name="项目介绍"></a>
## 项目介绍

*shurnim-storage* 的设计初衷是给大家提供一个可方便扩展的云存储/网盘同步工具。分后端接口和前端UI界面两部分。<br>
```

由于目前各种云存储和网盘系统层出不穷，单一工具往往支持支持某几个特定存储之间的同步，如**又拍云**到**七牛云存储**的同步工具，此时如若想同步到其他存则可能需要新的工具，给用户带来不便。*shurnim-storage* 正是为了解决此问题而设计的。

在*shurnim-storage*中，用户使用的固定的统一的后端接口。而所有云存储/网盘API的支持则是以插件的形式部署到系统中的。如此，如果用户想要一个从**又拍云**到**Dropbox**的同步工具，则只需要在原有基础上，增加**Dropbox**的插件，即可实现互通，方便快捷。

同时，后端统一接口的设计也考虑到界面开发的需求，可直接通过后端提供的接口开发具有上述扩展功能的云存储UI工具。

目前，后端整体框架的核心部分已经基本开发完成。只需逐步补充后端接口和插件开发接口的定义即可。但由于个人时间和能力所限，UI部分没有开发，有兴趣的同学可以一试。

使用说明

获取代码

* gitcafe项目主页: <https://gitcafe.com/onecoder/shurnim-storage-for-UPYUN>
* OSChina项目主页: <http://git.oschina.net/onecoder/shurnim-storage>
OSChina上的会持续更新。

另外你也可以通过OSChina的Maven库获取依赖，或者自己编译jar包。

* maven

    1. 加入OSC仓库

```
<repositories>
  <repository>
```

```
                <id>nexus</id>
                <name>local private nexus</name>
                <url>http://maven.oschina.net/content/groups/public/</url>
                <releases>
                    <enabled>true</enabled>
                </releases>
                <snapshots>
                    <enabled>false</enabled>
                </snapshots>
            </repository>
        </repositories>
```

2. 加入依赖

```
        <dependency>
          <groupId>com.coderli</groupId>
          <artifactId>shurnim-storage</artifactId>
           <version>0.1-alpha</version>
        </dependency>
```

* Gradle 编译Jar

在项目目录执行

```
gradle jar
```


开发插件

在*shurnim-storage*中，插件就像一块一块的积木，不但支撑着框架的功能，也是框架可扩展性的基石。开发一个

> 插件，仅需两步：
>
> 1. 实现PluginAPI接口
>
> ```
> package com.coderli.shurnim.storage.plugin;
>
> import java.io.File;
> import java.util.List;
>
> import com.coderli.shurnim.storage.plugin.model.Resource;
>
> /**
> * 各种云存储插件需要实现的通用接口
> *
> * @author OneCoder
> * @date 2014年4月22日 下午9:43:41
> * @website http://www.coderli.com
> */
> public interface PluginAPI {
>
>     /**
>      * 初始化接口
>      *
>      * @author OneCoder
>      * @date 2014年5月19日 下午10:47:40
>      */
>     void init();
>
>     /**
>      * 获取子资源列表
>
> ```

```
     *
     * @param parentPath
     * @return
     * @author OneCoder
     * @date 2014年4月24日 下午11:29:14
     */
    List<Resource> getChildResources(String parentPath);

    /**
     * 下载特定的资源
     *
     * @param parentPath
     *            目录路径
     * @param name
     *            资源名称
     * @param storePath
     *            下载资源保存路径
     * @return
     * @author OneCoder
     * @date 2014年4月24日 下午11:30:19
     */
    Resource downloadResource(String parentPath, String name, String storePath);

    /**
     * 创建文件夹
     *
     * @param path
     *            文件夹路径
```

```
     * @param auto
     *          是否自动创建父目录
     * @return
     * @author OneCoder
     * @date 2014年5月15日 下午10:10:04
     */
    boolean mkdir(String path, boolean auto);

    /**
     * 上传资源
     *
     * @param parentPath
     *          父目录路径
     * @param name
     *          资源名称
     * @param uploadFile
     *          待上传的本地文件
     * @return
     * @author OneCoder
     * @date 2014年5月15日 下午10:40:13
     */
    boolean uploadResource(String parentPath, String name, File uploadFile);
}
```

目前插件的接口列表仅为同步资源设计，如果想要支持更多操作(如删除，查找等)，可扩展该接口定义。

接口中，所有的参数和返回值均为*shurnim-storage*框架中定义的通用模型。因此，您在开发插件过程中需要将特定SDK中的模型转换成接口中提供的模型。

插件实现类只要与*shurnim-storage*工程在同一个classpath即可使用。您既可以直接在源码工程中开发插件，就如工程里提供的*upyun*和*qiniu*插件一样，也可以作为独立工程开发，打成jar，放置在同一个classpath下。

*upyun*插件样例(功能不完整):

```
package com.coderli.shurnim.storage.upyun.plugin;

import java.io.File;
import java.util.List;

import com.coderli.shurnim.storage.plugin.AbstractPluginAPI;
import com.coderli.shurnim.storage.plugin.model.Resource;
import com.coderli.shurnim.storage.plugin.model.Resource.Type;
import com.coderli.shurnim.storage.upyun.api.UpYun;

public class UpYunPlugin extends AbstractPluginAPI {

    private UpYun upyun;
    private String username;
    private String password;
    private String bucketName;

    public UpYun getUpyun() {
        return upyun;
    }

    public void setUpyun(UpYun upyun) {
        this.upyun = upyun;
    }

    public String getUsername() {
```

```
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getBucketName() {
        return bucketName;
    }

    public void setBucketName(String bucketName) {
        this.bucketName = bucketName;
    }

    /*
     * (non-Javadoc)
     *
     * @see
     * com.coderli.shurnim.storage.plugin.PluginAPI#getChildResources(java.lang
     * .String)
     */
    @Override
```

```
    public List<Resource> getChildResources(String parentPath) {
        return null;
    }

    /*
     * (non-Javadoc)
     *
     * @see
     * com.coderli.shurnim.storage.plugin.PluginAPI#downloadResource(java.lang
     * .String, java.lang.String, java.lang.String)
     */
    @Override
    public Resource downloadResource(String parentPath, String name,
            String storePath) {
        File storeFile = new File(storePath);
//        if (!storeFile.exists()) {
//            try {
//                storeFile.createNewFile();
//            } catch (IOException e) {
//                e.printStackTrace();
//            }
//        }
        String filePath = getFullPath(parentPath, name);
        upyun.readDir("/api");
        if (upyun.readFile(filePath, storeFile)) {
            Resource result = new Resource();
            result.setName(name);
            result.setPath(parentPath);
```

```
            result.setType(Type.FILE);
            result.setLocalFile(storeFile);
            return result;
        }
        return null;
    }

    String getFullPath(String parentPath, String name) {
        if (!parentPath.endsWith(File.separator)) {
            parentPath = parentPath + File.separator;
        }
        return parentPath + name;
    }

    /*
     * (non-Javadoc)
     *
     * @see com.coderli.shurnim.storage.plugin.PluginAPI#mkdir(java.lang.String,
     * boolean)
     */
    @Override
    public boolean mkdir(String path, boolean auto) {
        // TODO Auto-generated method stub
        return false;
    }

    /*
     * (non-Javadoc)
     *
```

```
     * @see
     * com.coderli.shurnim.storage.plugin.PluginAPI#uploadResource(java.lang
     * .String, java.lang.String, java.io.File)
     */
    @Override
    public boolean uploadResource(String parentPath, String name,
            File uploadFile) {
        // TODO Auto-generated method stub
        return false;
    }

    /*
     * (non-Javadoc)
     *
     * @see com.coderli.shurnim.storage.plugin.AbstractPluginAPI#init()
     */
    @Override
    public void init() {
        upyun = new UpYun(bucketName, username, password);
    }

}
```

2. 编写插件配置文件

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

```
<plugin>
    <id>qiniu</id>
    <name>七牛云存储</name>
    <api>
        <className>com.coderli.shurnim.storage.qiniu.QiniuPlugin</className>
        <params>
            <param name="access_key"
displayName="ACCESS_KEY">EjREKHI_GFXbQzyrKdVhhXrIRyj3fRC1s9UmZPZO
            </param>
            <param name="secret_key"
displayName="SECRET_KEY">88NofFWUvkfJ6T6rGRxlDSZOQxWkIxY2IsFIXJLX
            </param>
            <param name="bucketName" displayName="空间名">onecoder
            </param>
        </params>
    </api>
</plugin>
```

  * **id** 为该插件在*shurnim-storage*框架下的唯一标识，不可重复，必填。
  * **name** 为显示值，为UI开发提供可供显示的有语义的值。
  * **className** 为插件接口实现类的完整路径。必填
  * **params/param** 为插件需要用户配置的参数列表。其中
    * *name* 代表参数名，需要与接口实现类中的参数名严格一致，且必须有相应的set方法的格式要求严格，即set+首字母大写的参数名。例如:setAccess_key(String arg); 目前只支持*String*类型的参数。
    * *displayName* 为参数显示名，同样是为了UI开发的考虑，方便用户开发出可根据参数列表动态显示的UI界面。
    * 参数的值可以直接配置在配置文件中，也可以在运行期动态赋值。直接配置值，对于直接使用后端接口来说

较为方便。对于UI开发来说，运行期动态赋值更为合理。
</br>

在使用源码工程时，插件配置文件统一放置在工程的*plugins*目录下。你也可以统一放置在任何位置。此时，在构造后端接口实例时，需要告知接口该位置。

使用*ShurnimStorage*接口

接口介绍

ShurnimStorage接口是*shurinm-storage*框架全局的也是唯一的接口，目前定义如

```
package com.coderli.shurnim.storage;

import java.util.List;
import java.util.Map;

import com.coderli.shurnim.storage.plugin.model.Plugin;
import com.coderli.shurnim.storage.plugin.model.Resource;

/**
* 后台模块的全局接口<br>
* 通过该接口使用后台的全部功能。<br>
* 使用方式:<br>
* <li>
* 1.先通过{@link #getSupportedPlugins()}方法获取所有支持的平台/插件列表。 <li>
* 2.将列表中返回的ID传入对应的接口参数中，进行对应的平台的相关操作。<br>
* 需要注意的是，不同平台的插件需要给不同的参数赋值，该值可以直接配置在配置文件中。<br>
* 也可以在运行期动态赋值。(会覆盖配置文件中的值。)<br>
```

```
*
* 参数列表的设计，方便UI开发人员动态的根据参数列表生成可填写的控件。并给参数赋值。增强了可扩展性。
*
* @author OneCoder
* @date 2014年4月22日 下午9:21:58
* @website http://www.coderli.com
*/
public interface ShurnimStorage {

    /**
     * 获取当前支持的插件列表<br>
     * 没有支持的插件的时候可能返回null
     *
     * @return
     * @author OneCoder
     * @date 2014年5月7日 下午8:53:25
     */
    List<Plugin> getSupportedPlugins();

    /**
     * 给指定的插件的对应参数赋值<br>
     * 此处赋值会覆盖配置文件中的默认值
     *
     * @param pluginId
     *            插件ID
     * @param paramsKV
     *            参数键值对
     * @author OneCoder
```

```
 * @date 2014年5月9日 上午12:41:53
 */
void setParamValues(String pluginId, Map<String, String> paramsKV);

/**
 * 获取插件对应目录下的资源列表
 *
 * @param pluginId
 *            插件ID
 * @param path
 *            指定路径
 * @return
 * @author OneCoder
 * @date 2014年5月11日 上午8:52:00
 */
List<Resource> getResources(String pluginId, String path);

/**
 * 同步资源
 *
 * @param fromPluginId
 *            待同步的插件Id
 * @param toPluginIds
 *            目标插件Id
 * @param resource
 *            待同步的资源
 * @return 同步结果
 * @author OneCoder
```

```
     * @date 2014年5月11日 上午11:41:24
     */
    boolean sycnResource(String fromPluginId, String toPluginId,
                Resource resource) throws Exception;
}
```

当前接口实际仅包含了获取资源列表*getResources*和同步资源*sycnResource*功能，*getSupportedPlugins*和*setParamValues*实际为辅助接口，在UI开发时较为有用。

同样，您也可以扩展开发该接口增加更多的您喜欢的特性。例如，同时删除给定存储上的文件。当然，这需要插件接口的配合支持。

这里，*sycnResource*设计成插件间一对一的形式，是考虑到获取同步是否成功的结果的需求。如果您想开发一次同步到多个存储的功能，建议您重新开发您自己的接口实现类，因为默认实现会多次下次资源(每次同步后删除)，造成网络资源的浪费。

接口的默认实现类是: **DefaultShurnimStorageImpl**

使用样例

```
package com.coderli.shurnim.test.shurnimstorage;

import org.junit.Assert;
import org.junit.BeforeClass;
import org.junit.Test;

import com.coderli.shurnim.storage.DefaultShurnimStorageImpl;
import com.coderli.shurnim.storage.ShurnimStorage;
import com.coderli.shurnim.storage.plugin.model.Resource;
import com.coderli.shurnim.storage.plugin.model.Resource.Type;
```

```
/**
* 全局接口测试类<br>
* 时间有限，目前仅作整体接口测试。细粒度的单元测试，随开发补充。
*
* @author OneCoder
* @date 2014年5月19日 下午10:50:27
* @website http://www.coderli.com
*/
public class ShurnimStorageTest {

    private static ShurnimStorage shurnim;

    @BeforeClass
    public static void init() {
        shurnim = new DefaultShurnimStorageImpl(
                "/Users/apple/git/shurnim-storage-for-UPYUN/plugins");
    }

    @Test
    public void testSycnResource() {
        Resource syncResource = new Resource();
        syncResource.setPath("/api");
        syncResource.setName("api.html");
        syncResource.setType(Type.FILE);
        try {
            Assert.assertTrue(shurnim.sycnResource("upyun", "qiniu",
                    syncResource));
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
}
```


其他

时间仓促，功能简陋，望您包涵。OneCoder(Blog:http://www.coderli.com)特别希望看到该项目对您哪怕一点点的帮助。任意的意见和建议，欢迎随意与我沟通,联系方式：

* Email: <wushikezuo@gmail.com>
* QQ:57959968
* Blog:[OneCoder](http://www.coderli.com)

项目的Bug和改进点，可在OSChina上以issue的方式直接提交给我。

效果预览：

本文出自：http://www.coderli.com, 原文地址：http://www.coderli.com/write-readme-for-your-project, 感谢原作者分享。

更多 0

0 顶 0 踩

上一篇：30岁码农从零单排iOS 第一季 2. 类，对象和方法

下一篇：OneCoder的shurnim-storage项目

相关文章推荐：

如何在虚拟主机BlueHost上安装git和升级vim	1年前	windows git客户端安装（msysgit+tortoisegit）	1年前
用 gvim 比较 git diff	4年前	git 和 mercurial 的区别	4年前
"Are you doing science as well as you could b	1年前	基于Dropbox实现的免费私有Git版本库托管	10月前
搭建http协议的git服务器	9月前	迁移Bzr代码库到Git库中	5月前
使用Github和Vundle管理Vim插件和配置文件	1年前	写给 Git 初学者的 7 个建议	9月前
Git学习总结	3月前		

发表评论

发表评论