



# ColdFusion コンポーネント使い方

## cfmで使う側

### ▼ cfinvoke cfcのどのfunctionを使うか指示する

- method cfcのfunction要素
- component cfcのファイル名
- returnVariable 戻り値を受け取る名前

### ▼ cfinvokeargumentで渡したい引数を準備する（ないならなし）

- form送信した内容をusernameに入れて引数で渡している。

```
<!-- ユーザ追加のデータ渡す -->
<cfif isDefined("form.action") and form.action is "add">
    <cfinvoke method="createuser" component="login_function" returnvariable="message">
        <cfinvokeargument name="username" value="#form.username#">
        <cfinvokeargument name="password" value="#form.password#">
    </cfinvoke>
    <cfoutput>#message#</cfoutput>
</cfif>

<h2>ユーザの追加</h2>

<form action="#cfoutput>#cgi.script_name#</cfoutput>" method="post">
    <input type="hidden" name="action" value="add">
    ユーザー名:<input type="text" name="username" size="20"><br>
    パスワード:<input type="password" name="password" size="20"><br>
    <br>
    <input type="submit" value="追加">
</form>
```

## cfcで作業する側

▼ cfcomponent コンポーネントはまとめて1つでOK

- 名前を付けても多分使わない。cfmで使う名前はファイル名 login\_function.cfc を使うので。

▼ cffunction

- name cfmで付けた名前で
- access remote/public/private...
- returntype 戻り値の型。複数⇒**struct**、DB参照の結果⇒**query**

▼ cfargument 引数の受け取り（ないならなし）

- name 名前を付ける。cfmと一緒にの方が分かりやすい
- type 型
- required 引数の受け取りが必須かどうか

```
<!-- ユーザの追加 -->
<cfcomponent>
    <cffunction name="createuser" access="remote" returntype="string">
        <cfargument name="username" type="string" required="true">
        <cfargument name="password" type="string" required="true">

        <cfquery datasource="sample" name="adduserck">
            select *
            from accounttable
            where username = '#arguments.username#'
        </cfquery>

        <cfif adduserck.recordcount gt 0>
            <cfset message = "#arguments.username#様は既に存在しているため登録できません">
            <cfelse>

                <cfquery datasource="sample" name="useradd">
                    insert into accounttable values (
                        '#arguments.username#', '#encrypt(arguments.password, "aaa")#', null, null, null, 0, 0, null, 0, null
                    )
                </cfquery>
                <cfset message = "#arguments.username#様を追加しました">
            </cfif>
            <cfreturn message>
        </cffunction>
    </cfcomponent>
```



# Git管理の流れ

Quickly create a rich document.

## リポジトリ作成まで

1. githubでrepositoryを作っておく
2. gitbashで remote add origin URL をする。（gitcloneでもOK）
3. branch作成

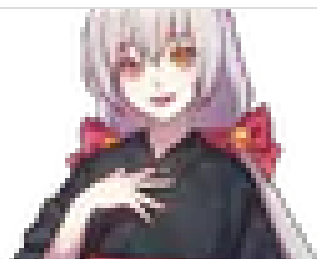
※この時masterに一度pushしないと怒られるので下記のコマンドやること。

```
- git commit --allow-empty -m "xxx"  
- git push origin master
```

### 【Git】fatal: Not a valid object name: 'master'.の解決方法 - Qiita

Git初心者にとって、Git公式ドキュメントを読むのはなかなか大変なことです。なぜなら、専門用語と情報量が多く読む気がなくなるからです。また、GitHubのドキュメントでは、英文で記載されている場合が有り これも

 [https://qiita.com/Sakuya\\_wd/items/b83161111b1b28098008](https://qiita.com/Sakuya_wd/items/b83161111b1b28098008)



## 「毎日OK」 pushするまで

1. branchの切り替え
2. `git add .`
3. `git commit (i qw)`
4. `git push origin [branch名]`

## branchが使い終わったら

---

1. github上でpull requestをする
2. レビューがOKだったgithubでmergeする
3. PC側がgithub上のコードを引っ張るので、gitbashでmasterに切り替えた後で `git pull` する
4. branchの破棄 `git branch -d ○○ブランチ名`