

Karrigell 官方文档(中文版)

---

# Karrigell 官方文档

(中文版)

## Karrigell 官方文档(中文版)

---

### 内容目录

|   |    |
|---|----|
| 原文地址: <a href="http://karrigell.sourceforge.net/en/include.htm">http://karrigell.sourceforge.net/en/include.htm</a> ..... | 1  |
| 2. Installing Karrigell.....  | 3  |
| 3. The Web server.....  | 3  |
| 3.1 The Root Directory.....   | 3  |
| 3.2 Serving static files.....   | 3  |
| 3.3 Command line options.....   | 4  |
| 5. Karrigell with Apache, lighttpd and Xitami.....  | 7  |
| 6.1 Accessing HTTP environment.....   | 7  |
| 6.2 Form fields.....  | 8  |
| 6.3 Smart urls.....   | 8  |
| 6.4 File uploads.....   | 9  |
| 6.5 Exceptions.....   | 10 |
| 6.6 HTMLStream .....  | 10 |
| 7. Debugging.....   | 11 |
| 7.1 Debug mode .....  | 11 |
| 7.2 Error handling and debugging.....   | 11 |
| 8. Python scripts .....   | 13 |
| 9. CGI scripts.....   | 14 |
| New in version 2.3.1 .....  | 14 |
| 10.1 Syntax.....  | 14 |
| 10.2 Smart urls.....  | 15 |
| 11.1 Python variables.....  | 16 |
| 11.2 Strings for translation.....   | 17 |
| 11.3 Indentation.....   | 17 |
| 11.3.1 Basics.....  | 17 |
| 11.3.3 The <indent> tag.....  | 18 |
| 11.4 PIH as a templating system.....  | 19 |
| 11.5 Debugging.....   | 19 |
| 13. HTMLTags - generate HTML in Python.....   | 20 |
| 14.1 Namespace.....   | 22 |
| 14.2 Passing parameters.....  | 22 |
| 15. Sessions.....   | 23 |
| 15.1 Example.....   | 23 |
| 16.2 The RestrictToAdmin() function.....  | 25 |
| 17. Internationalization and Unicode.....   | 26 |
| 17.1 Translation.....   | 26 |
| 17.2 Unicode support.....   | 27 |
| New in version 2.2.2.....   | 27 |
| 18. Karrigell namespace.....  | 28 |

## Karrigell 官方文档(中文版 )

---

### 1. Introduction

Karrigell 是一个用 Python 写的简单的 Web 解决方案。它被设计的简单易用：集成 web server 和数据库(PyDbLite)，可以通过简单的方式访问环境变量和表单字段，但是 Karrigell 的功能仍然很强大：Python 脚本在服务器的同一个进程被执行，服务器页面可以包含 HTML 和 Python 代码(就像 PHP,JSP,ASP)，简单的身份验证， session 管理，国际化等。

Karrigell 也可以同其它 web server 一齐工作，像 Apache,Xitami 和 LightTPD 都被 Karrigell 支持。并且当前流行的数据库(sqlite,MySQL,PostgreSQL,等)都可以通过相关的 Python API 在 Karrigell 中使用。

Karrigell 的分发包里自带有一组示例文件，包括一个 Wiki server,一个论坛，一个 blog 引擎和一个 portal。

### 2. Installing Karrigell

首先我们要到 Python 语言的官方网站去下载并安装一个 2.3 或更高版本的 Python。

然后我们把下载的 Karrigell 压缩文件解压到一个文件夹如 c:\Karrigell,然后进入这个叫 Karrigell-2.3.5 的文件夹，我们把这个文件夹叫做服务器文件夹。

为了运行 Karrigell，进入服务器文件夹然后打开控制台窗口运行：

```
(server dir)>Karrigell-2.3.5>python Karrigell.py
```

我们会看到如下信息:

```
Karrigell 2.3.5 running on port 80
```

```
Debug level 1
```

```
Press Ctrl+C to stop
```

在浏览器中输入 <http://localhost>，我们会看到 Karrigell 的主页。

我建议设置管理员用户名和密码：它们可用于许多程序，包括可用来管理 MySQL 或 sqlite 数据库。因此我们打开控制台窗口然后进入 admin 文件夹运行 script k\_password.py

```
(server dir)>Karrigell-2.3.5/admin>python k_password.py
```

然后输入用户名和密码。

### 3. The Web server

默认内置的 web server 是异步的，就像 Python 的标准模块 `asyncore` 和 `asynchat`。经过我测试后，发现它们很好用响应速度很快。

#### 3.1 The Root Directory

根目录是用来放其它文件的文件夹它在 URL 中处于最顶端,如，我们把 `myfile.htm` 放在根目录里，那么我们就可以通过 <http://myhost/myfile.htm> 来访问这个页面，其中 `myhost` 是我们主机的名字(本机是 `localhost`)。

刚安装 Karrigell 时，根目录默认是服务器目录下的 `webapps` 子目录。我们可以修改配置文件来改变这个设置。

#### 3.2 Serving static files

要在 Karrigell 里使用静态文件(HTML 文件，GIF 或 JPEG 图像，等)我们可以使用自己喜欢的工具创建然后把它们放在根目录里。

假设我们新建一个 HTML 文档叫做 `myFile.htm` 并把它放在根目录里，然后打开我们的浏览器输入 d 地址：

<http://myhost/myFile.htm>，我们就可以看到文件显示在浏览器里了。

如果我们把静态文件放在一个子目录里，那么我们就需要给它起一个名字：如文件 `myImage.gif` 放在子目录 `Images` 里，我

## Karrigell 官方文档(中文版 )

---

们就可以访问 <http://myhost/Images/myImage.gif>。

要想把这些文件放在根目录以外，我就要用到别名了。

如果一个地址没有指定文件名但是匹配一个目录，那么服务器将会依次查找 index.html,index.htm,index.py,index.pih,index.hip 或 index.ks。如果都没有找到就会打印出这个目录的所有子目录和文件，如果找到多个 index 文件将会触发一个异常。

如果在路径里没有指定文件的扩展名，服务器就会查找文件可能的一个扩展名 html,htm,py,pih,hip 或 ks。如果找到一个，这个文件就会被访问；如果找到多个文件名相同的文件，将会触发一个异常；如果文件没找到也会触发一个异常。

### 3.3 Command line options

命令行是：

```
python Karrigell.py [-P port] [-S] [-D] [initFile]
```

参数：

- port 是 HTTP 的端口号(默认是 80)
- S 代表 " silent " 模式：默认的 Karrigell 打印出收到的每一个请求的跟踪信息。使用 -S 选项，将不会打印出来。
- D 设置调试等级为 1
- initFile 是一个配置文件。默认的是服务器目录下的 Karrigell.ini 文件。

## 4. Configuration options

所有的配置选项都设置在一个配置文件里，默认的是服务器目录下的 Karrigell.ini 文件，我们可以在命令行指定一个不同的文件。

变量 base 在加载配置文件前设置 Karrigell 的位置，它的值就是服务器目录。可以参见 Karrigell.ini 文件里使用%  
(base)s/的地方。

配置文件被分成几个部分：

```
[Directories]
root
```

设置 root 选项为根目录的全称，它是我们访问文件的起始目录。在安装时是没有被设定的，默认是服务器目录下的 webapps。

cgi

Cgi scripts 存放的目录列表。

默认是服务器目录下的 webapps/cgi-bin 目录。

protected

它的值是一组被保护的目录；在这些目录里都要有一个叫做 AuthentScript.py 的脚本，在访问这些目录时首先执行这个脚本。这个脚本是把 admin 目录下的 AuthentScript.py 复制过来的。

## Karrigell 官方文档(中文版)

默认的只保护 admin 目录。可以使用 “;” 来分隔目录。

allow\_directory\_listing

如果一个 URL 没有指定 index 文件而匹配了一个目录，这个选项用来决定哪些用户可以看到这个目录的内容：

- all = all users
- none = no user

默认是” none ”。

hide\_extensions

哪类文件被隐藏起来不能访问(会返回一个 403 错误)。

hide\_paths

一个匹配 url 的正则表达式；如果匹配成功服务器会返回 403 错误。

它可以用来禁止访问一组目录。例如新建一个叫 cvs 的版本控制的目录并且不希望别人访问，可以这样设置：  
hide\_paths = .\*/cvs/.\*

loggingFile

日志文件名。如果没有设置，服务器就不会记录日志。

loggingParameters

日志处理器的参数，可以查阅 Python 的 logging 模块文档。

[Applications]

映射文件扩展名到 MIME 类型；我们知道它们在 Netscape 里才会有用，Internet Explorer 不会检查这些设置。

[Alias]

我们可以把一个别名关联到文件系统的一个路径。例如，我们创建这样一个别名：

scripts=c:\My Documents\Karrigell scripts

然后访问 <http://localhost/scripts/index.htm> 实际上访问的是 c:\My Documents\Karrigell scripts\index.html

PS：有点类似于 IIS 里的虚拟目录，但也不完全一样。

[Server]

port

设置 port 选项用来改变 HTTP 端口(默认是 80)。

## Karrigell 官方文档(中文版 )

---

`reload_modules`

如果设为 1，所有 import 的模块会在每次 import 时重新 import，因此我们在改变 import 的脚本时不需要重新启动服务器。默认是 1。

`debug`

`debug` 指示是否希望在错误页面显示“Debug”按钮。我们可能为了安全的原因把它设置为 0. 默认是 1(显示 debug 按钮)。

`silent`

如果 `silent` 选项设置为 1，那么控制台窗口将不会打印任何东西。

`gzip`

如果 `gzip` 选项设置为 1，并且用户代理支持 gzip 编码(大多数的浏览器都支持)，服务器将会压缩发送到用户代理的数据。这会减少用户的网路流量但是服务器会损失一些效能。

`global`

在 `global` 选项中，可以指定模块的路径它们会在运行 Karrigell 时被 import。我们可以像这样设置一行：

```
global=%(base)s/myScript.py;%(base)s/myScript1.py
```

`myScript` 和 `myScript1` 模块将会在命名空间里可以被所有的脚本使用。

`ignore`

当没有找到时而被忽略的一组 url 列表(返回 HTTP 代码 204 代替 404)。

[Translation]

指定应用程序使用的语言，而忽略 web 浏览器里的设置。如果不想要任何转换可以设置 `lang=default`，如果想转换成英语可以设置 `lang=en`，等。

[VirtualHost name]

Karrigell 支持虚拟主机的概念，允许在相同的服务器上使用不同的名字访问不同的主机。

为了指定一个虚拟主机我们要像下面这样在配置文件里加一行：

```
[VirtualHost name]  
name 是主机的名字(像 www.test.org)
```

在这个部分，我们要为这个指定的主机接收请求访问指定一个目录：

```
root = /web/mydir
```

## 5. Karrigell with Apache, lighttpd and Xitami

未完成

## 6. Programming

在 Karrigell 中有好几种写程序的方式:

- 纯 Python 脚本，使用简单的变量，还有 HTML Inside Python
- Karrigell Services 方式，这是一个 Python 脚本它可以处理整个站点(或者一些部分页面)
- 像 PHP 一样混和 HTML 与代码的方式，即 Python Inside HTML

这些不同的编程方式都使用相同的方式访问 HTTP 环境变量和表单数据，这些都会在下面的文档里分别描述。

我们也可以使用其它几种受支持的脚本：如果想使用.foo 来管理脚本，我们可以写一个叫作 mod\_foo 的模块。可以参见使用 mod\_tmpl.py 管理 Cheetah 模板系统管理的例子。

### 6.1 Accessing HTTP environment

在 Karrigell 脚本运行的名字空间里提供通过全局变量来访问 HTTP 环境变量：

- **HEADERS** 是一个带有 HTTP 头数据的字典：它的键值是头名字，值是相应头的值。比如 HEADERS["accept-language"] 将返回 accept-language 头值
- **RESPONSE** 是一个用来设置响应头以发送到服务器端的字典。这个字典键的名字是不区别大小写的：RESPONSE['Content-type'] 与 RESPONSE['CONTENT-type'] 返回同样的结果
- **AUTH\_USER** 和 **AUTH\_PASSWORD** 是用于身份认证的值
- **COOKIE** 是一个与 Python 的 Cookie 模块中的 SimpleCookie 对象相同的字典，它存储的 cookies 随着 HTTP 请求被 web 浏览器发送到服务器端
- **SET\_COOKIE** 是另一个 SimpleCookie 对象，我们可以设置它的键和值它将被当作 cookies 存储在客户端的浏览器中
- **ACCEPTED\_LANGUAGES** 是一个客户端浏览器可接受的语言列表，它按优先顺序排序。列表中的项都是根据 ISO 639 规范使用两个字符的字符串来标识一种语言(如 en 代表 English, fr 代表 French, 等)
- 在 更高级的应用中，**REQUEST\_HANDLER** 代表当前 RequestHandler 实例。它暴露出一些属性如 **client\_address** 它使用一个 元组来表示客户端的 IP 地址和端口。参见

## Karrigell 官方文档(中文版)

---

Python 标准分发包文档中的 BaseHTTPServer 和 SimpleHTTPServer 模块

- THIS 是一个代表当前脚本的脚本类的实例

### 6.2 Form fields

QUERY 是一个字典变量代表着当使用 HTTP GET 方法或 HTTP POST 方法提交表单数据时的查询字符串。QUERY 的键名字就是表单中的字段名，它的值就是字符串形式的字段值，或当字段名以[ ]结尾时使用列表来表示字段的值(比如一个<SELECT MULTIPLE>表单字段)

假设我们有这样一个 HTML 表单：

```
<form action="myScript.py">
Spam <input name="spam">
<br><select multiple name="animal[]">
<option value="dog">Dog
<option value="cat">Cat
<option value="frog">Frog
</select>
<br><input type="submit" value="Ok">
</form>
```

在 myScript.py 中 input 字段将会这样被显示：

```
print "<br>Spam is ", QUERY["spam"]
if QUERY.has_key("animal"):
    print "<br>Animal is ", str(QUERY["animal"])
```

可以通过一种快捷方式来访问这些数据，即使用\_与字段名组合起来。下面的代码将会像这样写：

```
print "<br>Spam is ",_spam
if QUERY.has_key("animal"):
    print "<br>Animal is ",str(_animal)
```

使用下划线是为了避免字段名与 Python 保留字或相同的模块名字冲突

### 6.3 Smart urls

通常情况下往一个脚本里传递参数是在 url 里使用查询字符串或者 post 数据，而另一种选择是把参数作为 url 的一部分来传递，就像 http://host/path/script.py/foo/bar，这里的参数就是 foo 和 bar

在脚本里我们可以通过一个列表来访问这些参数，它是 THIS 的一个属性即 THIS.subpath:

```
print "The parameters are %s" %THIS.subpath
```

## Karrigell 官方文档(中文版)

---

这个 url 存在的问题是当我们使用 `Include` 或写一个相关的链接或插入一个图像或 JavaScript, 如果调用这个脚本没有使用 `subpath` 或 `subpath` 中有一些参数这时 url 肯定会有所不同.

例如在脚本里写这样一个相关的 url:

```
print '<IMG SRC="images/pic.png">'
```

然后调用这个脚本 <http://host/path/script.py/foo/bar>, 浏览器将会计算绝对路径, 点击这个链接后:

<http://host/path/script.py/foo/images/pic.png>, 使用参数 `foo`, `images`, `pic.png` 执行相同的脚本显然这不是我们想要的。

因此如果我们计划在一个子路径里传递一个参数, 将必须像这样写相关的 url:

```
print '<IMG SRC="%simages/pic.png">' %THIS.up
```

`THIS.up` 代表着有多少参数就会有多少个字符串'...'

我们也可以使用 `baseurl` 属性, 它将会返回当前的路径 `path/`

## 6.4 File uploads

为了从客户端上传一个文件到服务器端, 表单中的 `input` 标签必须有一个类型为 '`file`' 的。比如有一个表单像这样:

```
<FORM ENCTYPE="multipart/form-data" ACTION="fileUpload.py" METHOD=POST>
File to process: <INPUT NAME="myfile" TYPE="file">
<INPUT TYPE="submit" VALUE="Send File">
</FORM>
```

那个脚本使用变量 `QUERY['myfile']` 或 `_myfile` 来处理要上传的文件, 它是 `cgi` 模块内置类 `FieldStorage` 的一个实例。

这个对象有两个属性:

`filename`: 文件的名字

- `file`: 一个像 `file` 的对象可用来读取文件的内容

例如我们想在服务器端的文件系统使用相同的文件名保存文件:

```
import os
f = _myfile.file # file-like object
dest_name = os.path.basename(_myfile.filename)
out = open(dest_name, 'wb')
```

## Karrigell 官方文档(中文版)

---

```
# copy file
import shutil
shutil.copyfileobj(f, out)
out.close()
```

### 6.5 Exceptions

在 Karrigell 的 Python 脚本中可以触发特殊的异常来处理脚本

**SCRIPT\_END**

当我们想停止发送数据到浏览器又不想让脚本执行到文件末尾可以使用这个异常。这个异常在我们调试脚本或想让它某个

地方停止运行以查看脚本状态或变量值时非常有用

```
myVar=10
...
print myVar
raise SCRIPT_END
... (rest of code - won't be run)
SCRIPT_ERROR
```

使用 `raise SCRIPT_ERROR, msg` 来停止脚本的执行并输出 `msg`

**HTTP\_ERROR**

`raise HTTP_ERROR, (errorCode, errorMessage)` 将导致 Karrigell 使用一个特定的错误代码和信息发送一个 HTTP 错误信息。

**HTTP\_REDIRECTION**

`raise HTTP_REDIRECTION, url` 将导致 Karrigell 重定向一个请求到一个给定的 URL

### 6.6 HTMLStream

HTMLStream 是 HIP 模块里面的一个类它比重复使用 `print` 语句要容易。它的设计像 HTML Inside Python 但使用不同的实现方式

创建这个类的一个实例:

```
import HIP
H = HIP.HTMLStream()
```

然后使用 "+" 和 "-" 来打印数据到标准输出: 使用 "+" 数据将当作字符串被打印出来, 使用 "-" 将使用 cgi-escaped 格式输出

```
aDict={"one":"unan","two":"daou","three":"tri"}
```

## Karrigell 官方文档(中文版)

H + aDict - type(aDict)

它们和下面的代码是一样的：

```
aDict={"one":"unan","two":"daou","three":"tri"}  
print str(aDict),cgi.escape(type(aDict))
```

## 7. Debugging

### 7.1 Debug mode

当我们测试新版本程序时，我们要重新导入改变后的模块通常会有些情形。考虑到性能的原因，标准 Python 解释器会导入模块已编译过的版本，而不会检查它的源代码是否修改过。我们可以设置配置文件里的 reloadModules 选项为 1，它会强制加载这个模块最后一次修改的版本。

一旦这个程序正式工作后，我们可以设置它的调试模式为 0

### 7.2 Error handling and debugging

当我们调用一个 URL 发生错误时，将会在浏览器里显示一个调用栈。

这些信息包括以下几个部分：

- 调用的 url
- 在一个表格里显示错误发生的地方：也有可能显示的不是脚本代码而是一个名字匹配的 url，在这种情况下一般是错误发生在使用 Include() 函数被包含的脚本里。这时会显示一个树状的结构。表格里会显示脚本名字，异常名字，在错误脚本里的行号与所在行的文本
- 未经过处理的 Python 调用栈
- 如果配置文件里的 debug 选项设置为 1 还会显示一个"Debug"按钮

例如：

**Error in /demo/errors/ErrorInIncludedTest1.py**

```
/demo/errors/ErrorInIncludedTest1.py  
includes /demo/errors/ErrorInIncludedTest2.py
```

Script **/demo/errors/ErrorInIncludedTest1.py**

```
NameError: name 'bonjour' is not defined  
Line 1
```

## Karrigell 官方文档(中文版)

```
print 'Script 1'
```

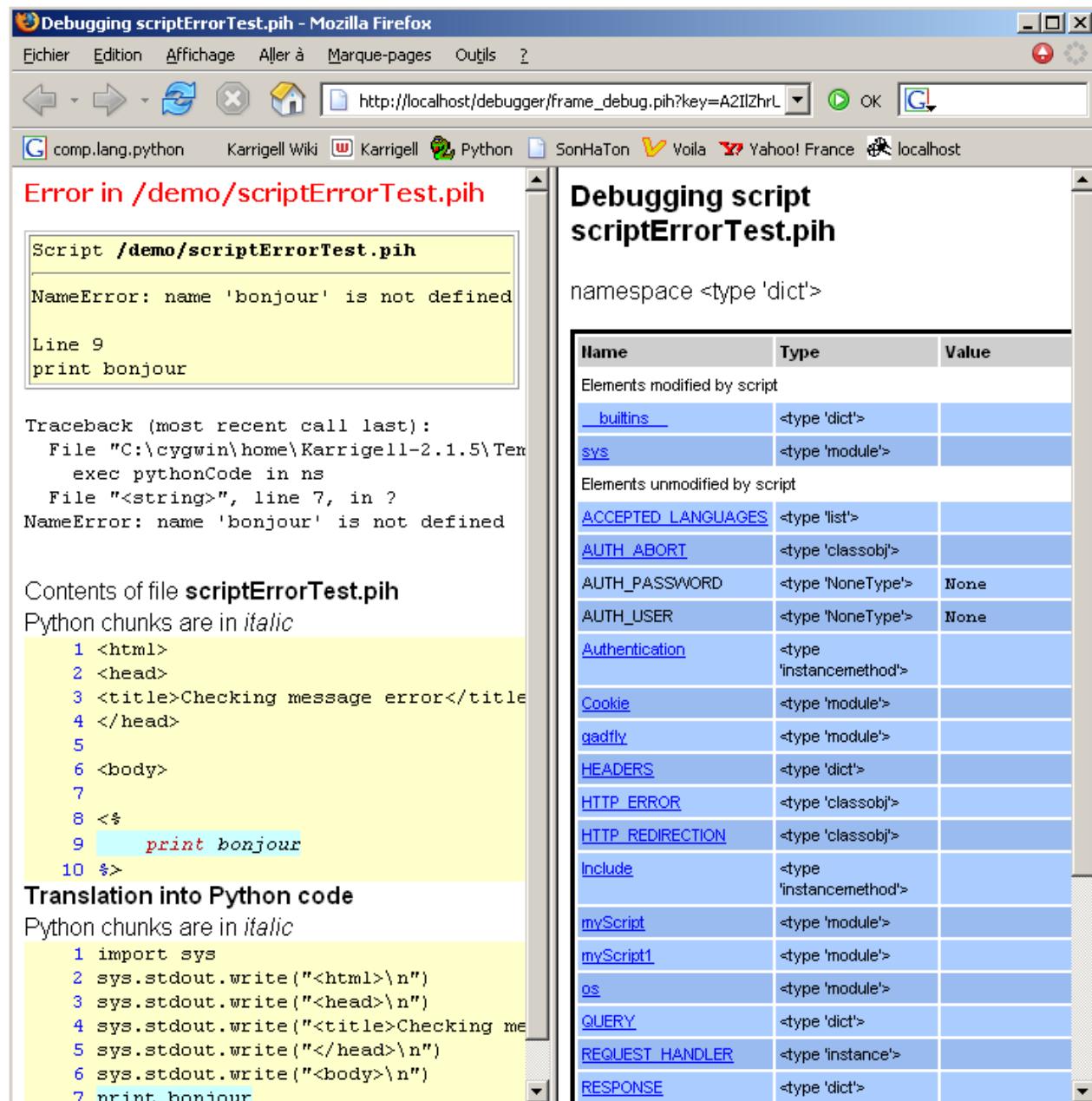
Traceback (most recent call last):

```
File "C:\cygwin\home\Karrigell\Template.py", line 153, in render
  exec self.pythonCode() in ns
File "<string>", line 1, in ?
NameError: name 'bonjour' is not defined
```

点击这个"Debug"按钮我们还可以得到关于这个错误的更多信息:

- 会语法加亮显示源代码，并且在错误发生的地方会高亮显示
- 一个环境变量的视图，我们可以在这个视图里浏览并查找变量的值和函数或方法的源代码等

屏幕截图:



**Error in /demo/scriptErrorTest.pih**

Script **/demo/scriptErrorTest.pih**

NameError: name 'bonjour' is not defined

Line 9  
print bonjour

Traceback (most recent call last):  
File "C:\cygwin\home\Karrigell-2.1.5\Template.py", line 153, in render  
exec self.pythonCode() in ns  
File "<string>", line 1, in ?  
NameError: name 'bonjour' is not defined

Contents of file **scriptErrorTest.pih**

Python chunks are in *italic*

```
1 <html>
2 <head>
3 <title>Checking message error</title>
4 </head>
5
6 <body>
7
8 <%
9   print bonjour
10 %>
```

Translation into Python code

Python chunks are in *italic*

```
1 import sys
2 sys.stdout.write("<html>\n")
3 sys.stdout.write("<head>\n")
4 sys.stdout.write("<title>Checking me
5 sys.stdout.write("</head>\n")
6 sys.stdout.write("<body>\n")
7 print bonjour
```

**Debugging script scriptErrorTest.pih**

namespace <type 'dict'>

| Name                               | Type                    | Value |
|------------------------------------|-------------------------|-------|
| Elements modified by script        |                         |       |
| <a href="#">builtins</a>           | <type 'dict'>           |       |
| <a href="#">sys</a>                | <type 'module'>         |       |
| Elements unmodified by script      |                         |       |
| <a href="#">ACCEPTED_LANGUAGES</a> | <type 'list'>           |       |
| <a href="#">AUTH_ABORT</a>         | <type 'classobj'>       |       |
| AUTH_PASSWORD                      | <type 'NoneType'>       | None  |
| AUTH_USER                          | <type 'NoneType'>       | None  |
| <a href="#">Authentication</a>     | <type 'instancemethod'> |       |
| <a href="#">Cookie</a>             | <type 'module'>         |       |
| <a href="#">gadfly</a>             | <type 'module'>         |       |
| <a href="#">HEADERS</a>            | <type 'dict'>           |       |
| <a href="#">HTTP_ERROR</a>         | <type 'classobj'>       |       |
| <a href="#">HTTP_REDIRECT</a>      | <type 'classobj'>       |       |
| <a href="#">Include</a>            | <type 'instancemethod'> |       |
| <a href="#">myScript</a>           | <type 'module'>         |       |
| <a href="#">myScript1</a>          | <type 'module'>         |       |
| <a href="#">os</a>                 | <type 'module'>         |       |
| <a href="#">QUERY</a>              | <type 'dict'>           |       |
| <a href="#">REQUEST_HANDLER</a>    | <type 'instance'>       |       |
| <a href="#">RESPONSE</a>           | <type 'dict'>           |       |

## Karrigell 官方文档(中文版)

---

如果我们不想让用户看到这些信息，我们可以设置配置文件里的 debug 选项为 0

## 8. Python scripts

在 Karrigell 中运行 Python 脚本是非常简单的。打开我们喜欢的 Python 代码编辑器创建下面的脚本：

```
print "Hello Karrigell !"
```

然后把它保存在 Root Directory 下面的 hello.py 里。然后访问 <http://localhost/hello.py>，我们就可以在浏览器里看到信息

Python 脚本在 Karrigell 里是以普通的脚本方式运行的，它接受 print 语句发送数据到客户端浏览器而不是控制台窗口。

因此我们需要以适当的格式书写 HTML 代码；一个简单的表格我们可以这样写：

```
print "<TABLE>"  
print "<TR>"  
print "<TD>Name</TD>"  
print "<TD>Address</TD>"  
print "</TR>"  
print "</TABLE>"
```

或者使用 Python 多行语法：

```
print """<TABLE>  
    <TR>  
        <TD>Name</TD>  
        <TD>Address</TD>  
    </TR>  
</TABLE>"""
```

或者我们使用 HTMLTags 模块：

```
from HTMLTags import *  
print TABLE(TR(TD("Name")+TD("Address")))
```

如果我们不想把脚本在 karrigell 里运行我们可以在脚本的最顶端包含这段代码：

```
try:  
    SCRIPT_END  
except NameError:  
    pass  
else:  
    print "This script can't be executed by Karrigell"
```

## Karrigell 官方文档(中文版)

---

```
raise SCRIPT_END

(... rest of your script here ...)
```

如果 `SCRIPT-END` 在 Python 脚本的运行名字空间里那就是运行在 Karrigell 里，在这个情况下就会结束执行；如果从命令行执行就会触发 `NameError` 异常并且会忽略异常处理的脚本

## 9. CGI scripts

### New in version 2.3.1

如果我们要在另一个环境上开发 CGI 脚本，我们可以不经过修改就能运行在 Karrigell 上。

提醒：在 Karrigell 2.3.1 里 CGI 脚本只能运行在异步服务器里，将会在以后添加其它服务器的支持

我们需要把 CGI 脚本放在一个特定的目录里。默认的是 `webapps/cgi-bin` 目录，我们也可以设置配置文件里 `Directories` 下的 `cgi` 选项来指定其它的路径

## 10. Karrigell Services

### 10.1 Syntax

"Karrigell Services" 也是一种 Python 脚本它可以处理一些 URL，因此由不同的 HTML 页面组成的一个完整的服务可以创建在一个脚本文件里。

为了达到这样的目的，在 Karrigell Service 中的一个函数就映射着一个 URL：在脚本 `dump.ks` 中的函数 `foo()` 可以调用 `dummy.ks/foo`。

要调用方法 `foo(arg1,arg2)`，URL 必须要像这样 `dummy.ks/foo?arg1=val1&arg2=val2` 或者通过一个带有字段 `arg1` 和 `arg2` 的表单来调用。

如果函数没有被指定，Karrigell 会查找一个不带参数的 `index()` 函数。

为了安全和可读性的理由，函数只能清楚的定义在 `ks` 脚本里，并且它的定义在源代码中要以第一列开始才能够被调用。

如果我们需要在脚本里定义一个函数但又不想它通过一个 url 来被调用，可以在函数名字前加上一个下划线。

```
def _private(value):
    """Private function - can't be called from the outside"""
    return value+1
```

为了从一个函数跳转到另外一个函数，可以为一个链接或一个表单的 `action` 属性指定一个函数名：

## Karrigell 官方文档(中文版)

---

```
def index():
    print '<a href="foo?name=bar">go to foo</a>'
def foo(name):
    print '<IMG SRC="../picture.jpg">'
    print name
```

注意 `foo()` 函数的第一行：因为这个 URL 决定的这个方法与文件或脚本相关连的 URL 同 `ks` 脚本是在同一个目录下，所以必须加上前缀“`.. /`”。

所有的 HTTP 环境变量，表单字段，自定义异常，身份认证函数，`session` 控制等都在 Python 脚本中以相同的方式处理。

下面是一个简单的 Karrigell Service 例子：

```
so = Session()
if not hasattr(so, 'x'):
    so.x = 0
def index():
    print "x = %s" % so.x
    print '<br><a href="increment">Increment</a>'
    print '<br><a href="decrement">Decrement</a>'
    print '<br><a href="reset">Reset</a>'

def increment():
    so.x = _private(so.x)
    raise HTTP_REDIRECTION, "index"
def decrement():
    so.x -= 1
    raise HTTP_REDIRECTION, "index"
def reset():
    so.x = 0
    raise HTTP_REDIRECTION, "index"
def _private(x):
    """The function name begins with _ : internal function,
    can't be call by a url"""
    return x+1
```

## 10.2 Smart urls

在访问 Karrigell Services 时 url 上带有附加的参数有的时候是非常有用的。例如，地址 `http://path/service.ks/function/foo/bar?name=smith`

将会使用 '`foo`' 和 '`bar`' 作为参数来调用 `function`。

它们可以在脚本中调用 THIS 的 `subpath` 属性。在下面的例子中将会列出 `['foo', 'bar']`

```
def function(name):
```

## Karrigell 官方文档(中文版)

```
print 'subpath ',THIS.subpath
print name
```

"smart urls"的一个问题是当我们要写一个链接或使用 `Include()` 或重定向到一个相关的 url, 我们就需要给出一个正确的 url,

是的, 要预先考虑好在目标 url 前有几个'../'。

有几个'../'可以由 `subpath` 属性的长度来决定, 如果我们想在下面的一个函数里写一个链接就需要这样写:

```
def function(name):
    print 'subpath ',THIS.subpath
    print name
    print '<a href=%slogin>Login</a>' %('../'*len(THIS.subpath))
```

或者使用 `THIS` 的 `up` 属性:

```
def function(name):
    print 'subpath ',THIS.subpath
    print name
    print '<a href=%slogin>Login</a>' %THIS.up
```

## 11. Python Inside HTML

Python Inside HTML 很像微软的 ASP 及 Sun 的 JSP 和 PHP; 它们基本上是 HTML 文档, 使用编程语言在其中插入部分代码, 当然了在这里指的是 Python。

在 Python Inside HTML 中, HTML 代码使用特殊的标签`<%`和`%>`来分开 Python 代码。

假设我们想显示当前日期, 我们将像这样混合 Python 和 HTML 代码:

```
The current date is
<% import time
print time.strftime("%d:%m:%y",time.localtime(time.time()))
%>
```

使用文本编辑器, 输入上面的代码并且把它保存到 Root Directory 下的 `time.pih` 里。在浏览器中输入 <http://localhost/time.pih> 然后看发生了什么。

我们会注意到在`<%`和`%>`中的代码是普通的 Python 代码, 在其中我们可以导入模块, 创建和实例化类, 使用变量, 读写文件系统等。可以像在 Python 脚本中一样访问 HTTP 变量, 表单字段, 及 Karrigell 定义的异常。

### **11.1 Python variables**

当我们只需要打印变量的值时, 可以使用`<%=var%>`来代替`<%print var%>`:

## Karrigell 官方文档(中文版)

Current directory <%= os.getcwd() %>

在这个例子中我们会注意到我们不需要显示地导入 os 模块：为了方便，在我们执行这个脚本的时候它已经在当前名字空间里了。还有 string 和 Cookie 模块也被导入了，因为它们可能会被大多数的脚本使用(当然，如果我们显式的使用 import string 我们的脚本也会正常的工作)。

### 11.2 Strings for translation

一直以来在 Karrigell 中国际化都是非常重要的部分，当我们想根据用户的参数选择来翻译字符串时可以使用<%\_string%>

```
<% import time %>
<%_ "Current directory is " %><%= os.getcwd() %>
<%_ "Current day " %><%_ time.strftime("%a",time.localtime()) %>
```

如果我们已经为字符串 Current directory is 准备好了译文，当用户调用这个脚本时并且它的语言选择的是法语，那么他的浏览器将会显示 Le répertoire courant est.

参见 Karrigell support for [internationalization](#)

### 11.3 Indentation

因为 PIH 文件里有 Python 代码的导致这些代码必须缩进。因为 PIH 脚本混合了 HTML，为了更好的可读性并且把 Python 代码分块所以要使用缩进，使用恰当的缩进可以使提取代码与可读性变得不是很困难。

#### 11.3.1 Basics

Python Inside HTML 遵循以下简单的规则：

- 在脚本的开始处缩进为 0
- 每一部分的缩进都要根据当前缩排
- 当前缩排可以两种意思：
  1. 当 Python 代码以冒号结束时，下一行的缩进将自增 1
  2. 为了结束缩排要使用<%end%>

一个简单的条件例子：

```
<% if hour<12: %>      Good morning      <% end><% elif hour<18: %>      Good afternoon      <% en
```

一个 for 循环的例子：

## Karrigell 官方文档(中文版)

---

```
<table border=1> <tr> <th>Number</th> <th>Square</th> </tr> <% for i in range(10)
```

如果没有`<%end%>`标签，`</table>`标签将会被插入到当前循环中

最后一个是有两个级别缩排的例子：

```
# indentation

<% for i in range(10): %> <% if i % 2: %> # 0 - next one : 1# 1 - next
<td class="odd"><%= i %></td> <% end %> one : 2# 2# 2 - next one : 1#
<% else: %> <td class="even"><%= i %></td> 1 - next one : 2# 2# 2 - next
<% end %> <% end %>End of table one : 1# 1 - next one : 0# 0
```

注意在第一行必须使用`%>`关闭标签，否则在第二行时缩排仍是 1。

### 11.3.3 The `<indent>` tag

在比较长的或更复杂的代码中重复使用`<%end%>`将会使代码变得冗长。如果想让 PIH 中的一些代码像在 Python 中一样缩进的话，可以嵌入`<indent>`标签。

第一个例子：

```
<indent><% if hour<12: %> Good morning<% elif hour<18: %> Good afternoon<% else: %>
```

第二个例子：

```
<table border=1> <tr> <th>Number</th> <th>Square</th> </tr> <indent> <% for i in
```

在注意事项(A)那一行上面我们看到了使用了`<indent>`标签来表示开始缩排

我们也注意到了在缩进部分结束的时候缩排返回了 0

这是一个嵌套循环的例子：

```
<indent>
<table border=1>
<% for i in ['h']+range(10): %>
<tr>
<% for j in ['h']+range(10): %>
<% if i!='h' and j!='h': %>
<td><%= i*j %></td>
<% elif i=='h': %>
<th><%= i %></th>
<% elif j=='h': %>
<th><%= j %></th>
```

## Karrigell 官方文档(中文版)

---

```
<% else: %>
<td>*</td>
</tr>
</table>
</indent>
```

### 11.4 PIH as a templating system

PIH 脚本可以在 Karrigell server 外用于创建 HTML 文件，把它当作"模板系统"。

例如，下面的这个 HTML 文档是一个 PIH 脚本。chapter 号码不是固定的但是可以像这样插入到 HTML 中：

```
<h1><%= chapter %>. Python Inside HTML</h1>
```

为了从一个 PIH 脚本中处理 HTML，可以使用 Template 模块，可以使用 getScript(fileName) 函数创建一个脚本的实例，然后应用它的 render(nameSpace) 方法，传递当前名字空间(通常是 globals() 或 locals()) 它将会发现需要的变量。render 方法返回一个 Output 实例，它的 value 属性就是返回的 HTML 结果：

```
import Template
pihIn=Template.getScript("pythoninsidehtml.pih")
chapter=5
htmlOutput=pihIn.render(globals()).value
```

### 11.5 Debugging

为了调试可以启动 PIHapp.py 脚本。这是一个小型的 GUI 应用程序它展示 PIH 文件如何转换成 Python 脚本。结果可以被保存成一个 HTML 文件并且显示在浏览器上，如果没有使用变量调用它的语法是高亮显示的。

## 12. HTML Inside Python

HTML Inside Python(PIH) 同 Python Inside HTML 是相反的；它在 Python 代码中包含 HTML 代码这要比使用 print 语句容易一些。我们可以使用两种方式：

- 短的 HTML 块不使用 print 语句，而使用一个字符串行；在执行的时候 HIP 将会自动添加 print 语句。

```
import os
currentDir=os.getcwd()
"Current directory is <b>" + currentDir + "</b>"
```

- 比较长的 HTML 块，使用 Python 的多行字符串语法：

```
the_smiths={'vocals':'Morrissey',
```

## Karrigell 官方文档(中文版)

---

```

'guitar':'Johnny Marr',
'the bass guitar':'Andy Rourke',
'the drums':'Mike Joyce'}
"""

<table border=1>
<tr backgroundcolor=green>
<td>One of the best pop bands ever</td>
</tr>
</table>
<table>
"""

for item in the_smiths.keys():
    print "<tr><td>%s</td><td>%s</td></tr>" %(item,the_smiths[item])
"</table>"

```

## 13. HTMLTags - generate HTML in Python

HTMLTags 模块为每个有效的标签使用大写字母定义了一个类。为了创建一个 HTML 块，一般的语法是：

```
t = TAG(innerHTML, key1=val1,key2=val2,...)
```

因此输出的结果是：

```
<TAG key1="val1" key2="val2" ...>innerHTML</TAG>
```

例如：

```
print A('bar', href="foo") ==> <A href="foo">bar</A>
```

与 Python 关键字(class,type)同名的属性必须首字母大写：

```
print DIV('bar', Class="title") ==> <DIV class="title">bar</A>
```

为了不使用值来生成 HTML 属性，要把它设为 True：

```
print OPTION('foo',SELECTED=True,value=5) ==> <OPTION value="5" SELECTED>
```

一些没有关标签的标签如<IMG>或<BR>， print 语句不会生成关标签。

innerHTML 参数可以是一个 HTML 类的实例，因为我们可以这样建造标签：

```
print B(I('foo')) ==> <B><I>foo</I></B>
```

HTML 的类支持加操作：

```
print B('bar')+INPUT(name="bar") ==> <B>bar</B><INPUT name="bar">
```

并且也支持重复操作：

```
print TH(' ')*3 ==> <TD>&nbsp;</TD><TD>&nbsp;</TD><TD>&nbsp;</TD>
```

## Karrigell 官方文档(中文版)

---

如果我们有一个实例列表，我们不能使用 `sum(instanceList)` 来连接 `items` 因为 `sum` 只能带几个参数。因此我们像这样调用 `Sum` 来完成这个工作：

```
Sum([ TR(TD(i)+TD(i*i)) for i in range(100) ])
```

上面的代码会生成一个有 0-99 个正方形列的行。

`innerHTML` 可以是一个字符串，但是我们不能连接一个字符串和一个 `HTML` 类实例，像这样：

```
H1(To be or ' + B('not to be'))
```

为了实现上面的功能，可以使用 `TEXT` 类，它不会生成任何标签：

```
H1(TEXT('To be or ') + B('not to be'))
```

一个简单的文档可以这样写：

```
print HTML( HEAD(TITLE('Test document')) +
    BODY(H1('This is a test document')+
        TEXT('First line')+BR()+
        TEXT('Second line')))
```

将会产生为：

```
<HTML>
<HEAD>
<TITLE>Test document</TITLE>
</HEAD>
<BODY>
<H1>This is a test document</H1>
First line
<BR>
Second line
</BODY>
</HTML>
```

如果文档比较复杂，为了更好的可读性可以先一个一个的创建元素类，然后把它们整个的打印出来。例如：

```
stylesheet = LINK(rel="Stylesheet",href="doc.css")
header = TITLE('Record collection')+stylesheet
title = H1('My record collection')
rows = Sum ([TR(TD(rec.title,Class="title")+TD(rec.artist,Class="Artist"))
    for rec in records])
table = TABLE(TR(TH('Title')+TH('Artist')) + rows)
print HTML(HEAD(header) + BODY(title + table))
```

## [14. Including other documents or scripts](#)

在一个脚本中可以通过 Karrigell 使用 `Include(url)` 函数来包含一个静态文件的内容或一个脚本文件的输出，`url` 是那个文件或脚本的地址(也可以使用普通文件的别名)。

当我们想利用已有的页面创建一组页面时这非常有用，在不同的页面中显示不同的内容(例如一个页头和/或页脚)。我们的脚本可以通过这样构建：

```
Include("authenticationTest.py") # a script testing user authentication
Include("header.htm") # a static file with a title, style sheet etc
(... your script body ...)
Include("footer.py") # a script which will print current date for instance
```

在运行时，当出现递归引用(脚本包含自己或脚本 1 包含脚本 2 而脚本 2 又包含脚本 1)Karrigell 会触发 `RecursionError` 异常。

### **14.1 Namespace**

在被调用的脚本中定义的变量与调用它的脚本在运行在同一名字空间里。如果调用脚用是：

```
name="Brian"
print "The life of "
Include("whoseName.py")
```

`whoseName.py` 的代码是：

```
print name
```

我们将看到 `name` 变量在调用脚本里是可以使用的。

### **14.2 Passing parameters**

也可以传递参数到一个被调用的脚本：可以参见个人 portal 事例的 `menu` 脚本。这个脚本带有一个菜单项列表参数和一个匹配的 `url` 的列表参数。

它将通过这种方式被包含：

```
Labels=...
Urls=...
Include("k_menu.hip",
    labels=Labels,
    urls=Urls,
    targetUrl="index.pih")
```

参数也可以通过追加在 `url` 后的查询字符串来传递：使用 `Include("/mypath/aScript.pih?data=dummy")` 中的变量 `data` 在当前名字空间里将是可用的。

## 15. Sessions

Session 是用来在服务器端内存中保存一个用户从一个页面到另一个页面相关信息的机制。

假设我们在一个网站上发现并想购买两张 CD，然后我们在另一个页面上发现了一本书。然后我们会被询问关于住址信息，如果是一个礼物还会发送一个信息，然后可能会问及我们的信用卡号。如果我们确认购买这时就会出现一个页面显示所有的信息。

如果没有 sessions 这是很难实现的，那我们每次都要传递上一次的信息到一个隐藏的表单字段，这是很难实现的。

有了 sessions 一切变得非常容易；每个用户都带有一个"session identifier"标识每次向服务发送请求时都会在表单下附加一个 cookie。这个标识符匹配服务器端的一个对象可以设置或得到它的属性值。

在 Karrigell 中，在页面中如果要想修改或访问 session 信息，首先要创建一个 session 对象：

```
sessionObj=Session()
```

如果我们开始了一个会话，Karrigell 就会生成一个叫做 sessionId 的 cookie 并被发送到 web 浏览器。在后面的请求浏览器会自动的发送这个 cookie 并且服务器端也会寻找与之相关的对象。

session 对象是一个普通的 Python 对象，我们可以设置它的属性：

```
sessionObj.name="John Doe"
```

在另一个脚本我们可以访问它的值：

```
name=sessionObj.name
```

Session 对象支持一个 close() 方法，它会导致 session 信息丢失。

我们不必显示的关闭一个 session，Karrigell 会确保不会多于 1000 个 session 当第 1000 个 session 被创建时它就会清除时间最久的那个 session.

### 15.1 Example

在一个 HTML 文件里创建一个表单然后发送到一个 PIH 脚本：

```
<h3>Who are you ?</h3>
<form action="sessionTestBegin.pih">
Name <input name="myname">
<br>First name <input name="firstname">
<br><input type="submit" value="Ok">
</form>
```

下面的脚本将会通过 QUERY 或者表单字段变量来得到输入的值，创建一个 session 对象并且把 name 和 firstname 作为这个对象的属性保存起来。

## Karrigell 官方文档(中文版 )

---

```
<h3>Begin session</h3>
<%
sessionObj=Session()
sessionObj.name=_myname
sessionObj.firstname=_firstname
print sessionObj.name
%>
Next...</a>
```

下一个脚本没有使用查询字符串，但它仍然可以通过 session 对象来得到信息：

```
<h3>Session test goes on</h3>
<%
session=Session()
print session.firstname
session.close()
%>
```

因为上面的脚本关闭了 session，如果我们回到上一页然后再一次点击 Next...链接我们将会得到一个 Python 的调用栈告诉我们 session 没有 firstname 属性。修改上面的脚本删除或注释 session.close() 这一行看看会发生什么。

## 16. Authentication

### 16.1 Basic HTTP authentication

Karrigell 支持基本的 HTTP 身份验证，一种方式就是通过要求用户名和密码来保护一些文件的访问。服务器在接受到用户输入时会把它保存在全局变量 AUTH\_USER 和 AUTH\_PASSWORD 中。如果用户的输入被验证后服务器就会发送一个页面，如果没有被验证用户就会被要求重新输入用户名和密码，如果用户取消了输入就会看到错误的信息。在 Karrigell 中，身份验证是通过 Authentication 函数来处理的，调用

[Authentication\(testFunction\[,realm,errorMessage\]\) where testFunction is a user-defined function taking no argument, which returns true if the authentication test succeeds \(depending on the values of AUTH\\_USER and AUTH\\_PASSWORD\) and false otherwise, realm is a string with the name of the authentication domain \(the one that will appear on the popup window\) and errorMessage is a string displayed on the browser if the user cancels his authentication request. Both realm and errorMessage have default values](#)

## Karrigell 官方文档(中文版)

---

Here is an example with a very simple test function :

```
<%
def authTest():
    return (AUTH_USER=="proust" and AUTH_PASSWORD=="marcel")
Authentication(authTest,"Authentication test", \
    "Sorry, you are not allowed to access this page")
%>
Welcome, authenticated user !
```

With this test function, if a visitor finds a way to read the source code, he will easily discover a valid login/password couple. A better solution is to use md5 : it is a function which takes a string as argument, and returns a 16-bytes "digest". The digest is guaranteed to be different for two different strings, and it is impossible to find the string if you only know the digest

A good method is to compute the md5 digests of user and password and store them in a file. The authentication test will read this file, compute the digests of AUTH\_USER and AUTH\_PASSWORD, and return true if the digests match

```
<%
import md5
digests=open("digest.ini","rb").read()
userDigest=digests[:16]
passwordDigest=digests[16:]
def authTest():
    return (md5.new(AUTH_USER).digest()==userDigest and \
        md5.new(AUTH_PASSWORD).digest()==passwordDigest)
Authentication(authTest,"Authentication test", \
    "Sorry, you are not allowed to access this page")
%>
Welcome, authenticated user !
```

See the `k_password.py` script, in the `admin` directory, which generates a file with the md5 digests of administrator's login and password

## 16.2 The `RestrictToAdmin()` function

A shortcut is provided to restrict access to a page to the administrator whose login and password have been defined by the `k_password.py` script : a function called `RestrictToAdmin()`. Put it at the beginning of your script, like this :

```
RestrictTo Admin()
print "Hello !"
```

The browser will ask for the admin's information before showing the page. By default, this information is searched in the file `admin.ini` in the folder `admin`. If you want to use another file you can specify it as argument to the function `RestrictToAdmin()` :

```
RestrictTo Admin(admin_file_name)
print "Hello !"
```

The format of this file must be the same as the one generated by `k_password.py`

## 17. Internationalization and Unicode

As you'll have guessed by reading this documentation, I'm not from an English-speaking country (I'm French, and more precisely Breton - the name Karrigell is a Breton word). So I've included a program to facilitate internationalization of scripts

### 17.1 Translation

In a script, every time you want a message translated into a given language, instead of writing it as a normal string with quotes, it's written using a function called `_`, this way :

```
print _("Hello everybody")
```

In Python Inside HTML (PIH) you can use the shortcut `<%_>` :

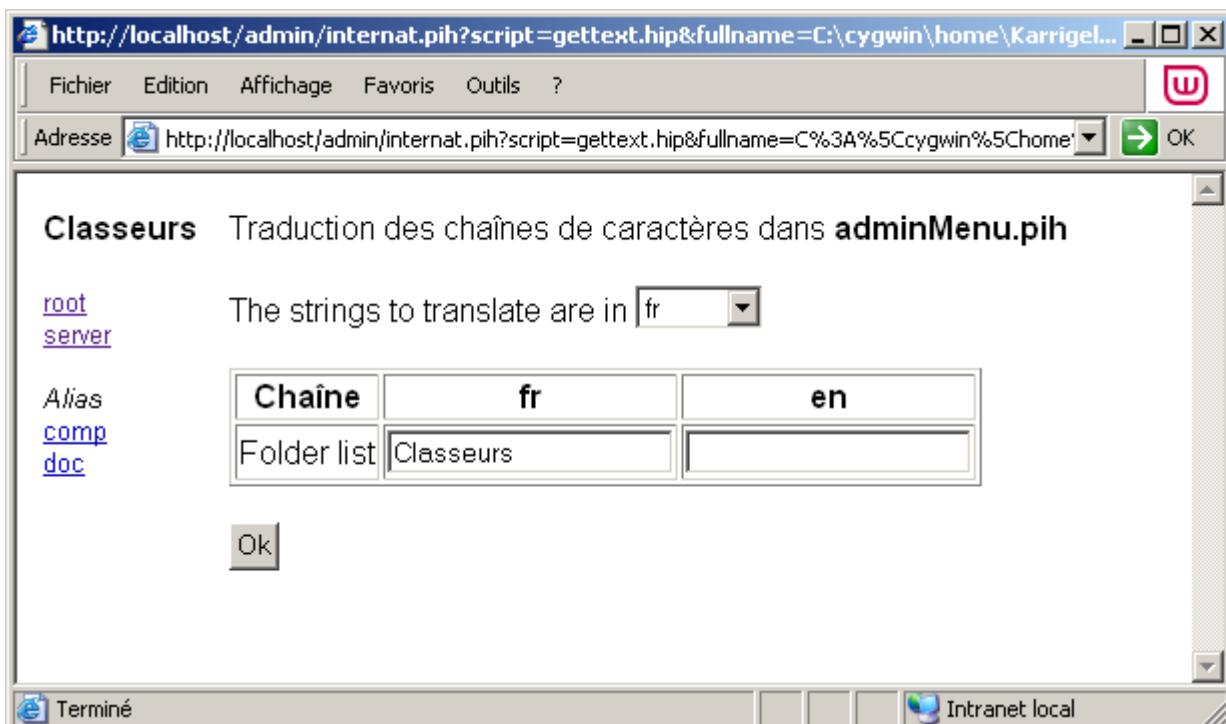
```
<%_ "Hello everybody" %>
```

Karrigell provides a simple web interface to create and modify translations of strings. For security reasons, the script that manages translation is reserved to the administrator. An authentication script is run, relying on md5 digests stored in a file called `admin.ini`, which the administrator **must** set by running the script `k_password.py` in the directory `admin`

With your browser, call the script `http://localhost/admin/internat.pih`. It opens a directory browser with which you can access all the files which may contain strings to translate (that is, all the files with an extension `.py`, `.pyw`, `.pih`, `.hip`). When clicking on a file name, a page appears with all the strings to translate (the arguments of the `_()` function) and asks for a translation in all the languages currently chosen in the browser language preferences[\(1\)](#). If translations have already been made they appear in the form fields

Fill in the fields and validate the form ; this creates or updates the translations

## Karrigell 官方文档(中文版)



You can check the effect by calling the script you've just modified and changing the language order in the preferences

Translation is held in a file which is common to all the files in the same directory. You can also edit the whole dictionary clicking on the first item in the script list

## 17.2 Unicode support

### New in version 2.2.2

*mostly written by Radovan Garabik*

Unicode is a normalized standard used to represent all the writing styles in the world. For each sign (a letter in any alphabet, an ideogram in an Asiatic language) Unicode defines a unique number, called a "code point". Since computers and networks can only manage bytes, a mapping between "code points" and one or several bytes must be defined ; these mappings are called "encodings"

Because there are many different encodings, when a program has to print a sign (a greek letter, a math symbol, a Chinese sign) it must receive **two** pieces of information : the string representing the sign (a sequence of bytes) **and** the encoding used. If it receives only a string, the program can try to guess an encoding (this is what a web browser usually does) but with no guarantee of success

Karrigell defines two parameters in the configuration file to handle Unicode :

- `encodeFormData` : if set to 1, Karrigell tries to transform form data into Unicode strings, trying different encodings one after the other. More precisely, if the received data only have ASCII characters, no conversion is made, the string is kept as is ; otherwise several encodings are tried and the first one that succeeds is used

## Karrigell 官方文档(中文版)

---

So, the form data available in QUERY are either bytestrings or Unicode strings  
 encodeFormData defaults to 0 : no Unicode conversion is made

- outputEncoding : defines the encoding that the browser will use to interpret the strings that it's asked to print. The default value is ISO-8859-1, an encoding used to represent the signs of western european languages based on the latin alphabet. You can define another encoding by setting outputEncoding in section [Server] of the configuration file to another value

You can see examples in the demo/unicode directory : set outputEncoding to utf-8 to see the expected result

---

- (1) On Microsoft Internet Explorer the language preference is set by Tools/Internet Options/General/Languages ; accepted languages are chosen from a list and ordered by preference. On Firefox use Edition/Preferences/Languages

## 18. Karrigell namespace

This page lists all the names available in Karrigell scripts

|                    |  |
|--------------------|--|
| ACCEPTED_LANGUAGES | The list of languages accepted by the client (they can be set in the browser)  |
| AUTH_USER          | User name if <a href="#">basic HTTP authentication</a> is used   |
| AUTH_PASSWORD      | User password if <a href="#">basic HTTP authentication</a> is used   |
| Authentication     | Function used for <a href="#">basic HTTP Authentication</a>  |
| CONFIG             | Configuration options, as set in the command line or in the configuration file<br><br>Attributes<br>serverDir<br>Directory where the Karrigell.py script stands<br>rootDir<br>Root directory, from where files and scripts are served<br>outputEncoding<br>The encoding used for output<br>COOKIE<br>The cookies sent by the client to the server. Instance of the SimpleCookie class in Python cookie module<br>HEADERS<br>The request headers, sent by the user agent (browser) to the server<br>HTTP_REDIRECTON |

## Karrigell 官方文档(中文版)

`raise HTTP_REDIRECTION, url` performs an HTTP redirection (302) to the specified url

### Include()

Function used to include a file or a script in another script

### Login([script\_url])

Function user to limit access to the script to authenticated users. By default, redirects to the script `login.php` in the root directory, and access is restricted to the administrator. Another script can be specified to customize the authentication test

### PATH

Script path

### QUERY

When a script receives form data, QUERY is a dictionary mapping field names to field values

### REQUEST

When a script receives form data, REQUEST is a dictionary mapping field names to field values (same as QUERY)

### REQUEST\_HANDLER

An object representing the current request handler

## Attributes

### client\_address

A tuple with client IP address and port, for example ('127.0.0.1', 1825)

### command

The HTTP command or method (usually GET or POST)

### encoding

The encoding as returned by function `guess_type` in built-in module `mimetypes`

### RESPONSE

The response headers sent by the server to the client, such as Content-type  
`RestrictToAdmin()`

Function used to restrict access of the script to the site administrator

### SCRIPT\_END

Exception to stop the script execution

### SCRIPT\_ERROR

`raise SCRIPT_ERROR, msg` to stop the execution of script and write msg  
`Session()`

Function that returns the [session object](#)

### SET\_COOKIE

The Cookies sent by the server to the client. Instance of the SimpleCookie

## Karrigell 官方文档(中文版)

---

|                              |  |
|------------------------------|--|
|                              | <p>class in Python cookie module</p> <p><b>THIS</b></p> <p>An object representing the current script. The examples suppose that the script path in the filesystem is<br/>C:\\cygwin\\home\\Karrigell\\webapps\\demo, and that it is called by the url demo/thisTest.pih</p> <p><b>Attributes</b></p> <p><b>basename</b><br/>Script base name (thisTest.pih)</p> <p><b>baseurl</b><br/>Part of the url before the script name (demo/)</p> <p><b>code</b><br/>The script source code (or the conversion into Python code for pih scripts)</p> <p><b>dirname</b><br/>Directory name (C:\\cygwin\\home\\Karrigell\\webapps\\demo)</p> <p><b>extension</b><br/>Script extention (pih)</p> <p><b>name</b><br/>Full name of the script in the file system<br/>(C:\\cygwin\\home\\Karrigell\\webapps\\demo\\thisTest.pih)</p> <p><b>parent</b><br/>The script's parent if it was included by the Include() function, or None</p> <p><b>path</b><br/>The path used by the client to request the script , including the query string if there was one (/demo/thisTest.pih)</p> <p><b>subpath</b><br/>If the script is requested with additional arguments separated by /, subpath is the list of these argements (see "<a href="#">smart urls</a>")</p> <p><b>url</b><br/>The url used to request the script, stripped from the query string if there was one (/demo/thisTest.pih)</p> |
| <b>COOKIE</b>                | The cookies sent by the client to the server. Instance of the SimpleCookie class in Python cookie module   |
| <b>HEADERS</b>               | The request headers, sent by the user agent (browser) to the server  |
| <b>HTTP_REDIRECT<br/>ION</b> | raise HTTP_REDIRECT, url performs an HTTP redirection (302) to the specified url   |
| <b>Include()</b>             | Function used to include a file or a script in another script  |
| <b>Login([script_url])</b>   | Function user to limit access to the script to authenticated users. By default, redirects to the script login.pih in the root directory, and access is   |

## Karrigell 官方文档(中文版)

---

|                 |   |
|-----------------|---|
|                 | restricted to the administrator. Another script can be specified to customize the authentication test   |
| PATH            | Script path   |
| QUERY           | When a script receives form data, QUERY is a dictionary mapping field names to field values   |
| REQUEST         | When a script receives form data, REQUEST is a dictionary mapping field names to field values (same as QUERY)   |
| REQUEST_HANDLER | <p>An object representing the current request handler</p> <p>Attributes</p> <p>client_address</p> <p>A tuple with client IP address and port, for example ('127.0.0.1', 1825)</p> <p>command</p> <p>The HTTP command or method (usually GET or POST)</p> <p>encoding</p> <p>The encoding as returned by function guess_type in built-in module mimetypes</p> <p>RESPONSE</p> <p>The response headers sent by the server to the client, such as Content-type</p> <p>RestrictToAdmin()</p> <p>Function used to restrict access of the script to the site administrator</p> <p>SCRIPT_END</p> <p>Exception to stop the script execution</p> <p>SCRIPT_ERROR</p> <pre>raise SCRIPT_ERROR, msg to stop the execution of script and write msg</pre> <p>Session()</p> <p>Function that returns the <a href="#">session object</a></p> <p>SET_COOKIE</p> <p>The Cookies sent by the server to the client. Instance of the SimpleCookie class in Python cookie module</p> <p>THIS</p> <p>An object representing the current script. The examples suppose that the script path in the filesystem is</p> <p>C:\cygwin\home\Karrigell\webapps\demo, and that it is called by the url demo/thisTest.pih</p> <p>Attributes</p> <p>basename</p> <p>Script base name (thisTest.pih)</p> |

## Karrigell 官方文档(中文版)

---

|                          |  |
|--------------------------|--|
|                          | <b>baseurl</b><br>Part of the url before the script name (demo/)<br><b>code</b><br>The script source code (or the conversion into Python code for pih scripts)<br><b>dirname</b><br>Directory name (C:\cygwin\home\Karrigell\webapps\demo)<br><b>extension</b><br>Script extention (pih)<br><b>name</b><br>Full name of the script in the file system<br>(C:\cygwin\home\Karrigell\webapps\demo>thisTest.pih)<br><b>parent</b><br>The script's parent if it was included by the Include() function, or None<br><b>path</b><br>The path used by the client to request the script , including the query string if there was one (/demo/thisTest.pih)<br><b>subpath</b><br>If the script is requested with additional arguments separated by /, subpath is the list of these argements (see " <a href="#">smart urls</a> ")<br><b>url</b><br>The url used to request the script, stripped from the query string if there was one (/demo/thisTest.pih) |
| <b>RESPONSE</b>          | The response headers sent by the server to the client, such as Content-type  |
| <b>RestrictToAdmin()</b> | Function used to restrict access of the script to the site administrator   |
| <b>SCRIPT_END</b>        | Exception to stop the script execution   |
| <b>SCRIPT_ERROR</b>      | <code>raise SCRIPT_ERROR, msg</code> to stop the execution of script and write msg   |
| <b>Session()</b>         | Function that returns the <a href="#">session object</a>   |
| <b>SET_COOKIE</b>        | The Cookies sent by the server to the client. Instance of the SimpleCookie class in Python cookie module   |
| <b>THIS</b>              | An object representing the current script. The examples suppose that the script path in the filesystem is<br>C:\cygwin\home\Karrigell\webapps\demo, and that it is called by the url demo/thisTest.pih<br><br>Attributes<br><br><b>basename</b><br>Script base name (thisTest.pih)<br><b>baseurl</b><br>Part of the url before the script name (demo/)<br><b>code</b>  |

## Karrigell 官方文档(中文版)

---

The script source code (or the conversion into Python code for pih scripts)  
dirname

Directory name (C:\cygwin\home\Karrigell\webapps\demo)  
extension

Script extention (pih)  
name

Full name of the script in the file system

(C:\cygwin\home\Karrigell\webapps\demo>thisTest.pih)  
parent

The script's parent if it was included by the Include() function, or None  
path

The path used by the client to request the script , including the query string if  
there was one (/demo>thisTest.pih)  
subpath

If the script is requested with additional arguments separated by /, subpath is  
the list of these argements (see "[smart urls](#)")

url

The url used to request the script, stripped from the query string if there was  
one (/demo>thisTest.pih)