**学习笔记**

**人物简介中用到的html标签详解**

|  |  |  |
| --- | --- | --- |
| 1. | <!doctype html> | <!DOCTYPE> 声明必须是 HTML 文档的第一行，位于 <html> 标签之前。  <!DOCTYPE> 声明不是 HTML 标签；它是指示 web 浏览器关于页面使用哪个 HTML 版本进行编写的指令。  在 HTML 4.01 中有三种 <!DOCTYPE> 声明。在 HTML5 中只有一种：<!DOCTYPE html> |
| 2. | <html> | 定义 HTML 文档。 |
| 3. | <head> | 定义关于文档的信息 |
| 4. | <meta> | 1. 定义关于 HTML 文档的元信息。在 HTML 中，<meta> 标签没有结束标签。但是被词用到的其它标签都是成对出现的。 2. <meta> 标签永远位于 head 元素内部。   3. 元数据总是以名称/值的形式被成对传递的。 |
| 5. | <title> | <title> 元素可定义文档的标题。  浏览器会以特殊的方式来使用标题，并且通常把它放置在浏览器窗口的标题栏或状态栏上。同样，当把文档加入用户的链接列表或者收藏夹或书签列表时，标题将成为该文档链接的默认名称。    延伸阅读：标题里是什么？  一定要选择一个正确的标题，这对于定义文档并确保它能够在 Web 上有效利用来说是十分重要的。  请记住，用户可以用任何顺序、独立地访问文档集中的每一个文档。所以，文档的标题不仅应当根据其他文档的上下文定义，而且还要显示其自身的特点。  含有文档引用排序的标题通常不是什么好标题。举个例子，像“第十六章”或“第五部分”这样的标题，对读者理解其内容方面毫无用处。描述性更强的标题，像“第十六章：HTML 标题”，或者“第五部分：如何使用标题”，这样的标题不仅表达了它在一个大型文档集中的位置，还说明了文档的具体内容，吸引读者更有兴趣读下去。  自我引用的标题也没有什么用处。像“主页”这样的标题和内容毫无关系，类似的还有“反馈页”或“常用链接”等。你应该设计一个能够传达一定内容和目的的标题，令读者凭这个标题就可以判断是否由必要访问这个页面。“HTML <title> 标签的详细信息”，这就是一个描述性的标题，类似的还有“HTML <title> 标签的反馈页”等等。  人们常常会花费大量时间去创建 Web 文档，但却经常只是因为一个不吸引人或无意义的标题，而把这些努力全都浪费掉了。当自动为用户搜集链接的特殊软件在 Web 上越来越流行时，只有网页的标题才会作为与页面相关的描述性词语，被插入到庞大的链接数据库中。因此，我们怎么强调这一点都不过分：请为自己的每个文档都认真地选择一个描述性的、实用的并与上下文独立的标题。  在人物简介的html文件中输入:    在页面上显示为 |
| 6. | <body> | 1. 定义文档的主体。  2. body 元素包含文档的所有内容（比如文本、超链接、图像、表格和列表等等。） |
| 7. | <div> | <div> 可定义文档中的分区或节（division/section）。  <div> 标签可以把文档分割为独立的、不同的部分。它可以用作严格的组织工具，并且不使用任何格式与其关联。  如果用 id 或 class 来标记 <div>，那么该标签的作用会变得更加有效。  用法  <div> 是一个块级元素。这意味着它的内容自动地开始一个新行。实际上，换行是 <div> 固有的唯一格式表现。可以通过 <div> 的 class 或 id 应用额外的样式。  不必为每一个 <div> 都加上类或 id，虽然这样做也有一定的好处。  可以对同一个 <div> 元素应用 class 或 id 属性，但是更常见的情况是只应用其中一种。这两者的主要差异是，class 用于元素组（类似的元素，或者可以理解为某一类元素），而 id 用于标识单独的唯一的元素。    在人物简介的html文件中输入:    在浏览器中显示的效果为： |
| 8. | <b> |  |
| 9. | table | 定义和用法  <table> 标签定义 HTML 表格。  简单的 HTML 表格由 table 元素以及一个或多个 tr、th 或 td 元素组成。  tr 元素定义表格行，th 元素定义表头，td 元素定义表格单元。  更复杂的 HTML 表格也可能包括 caption、col、colgroup、thead、tfoot 以及 tbody 元素。 |
| 10. | <tr> | <tr> 标签定义 HTML 表格中的行。  tr 元素包含一个或多个 th 或 td 元素。 |
| 11. | <th> | 定义表格内的表头单元格。  HTML 表单中有两种类型的单元格：  表头单元格 - 包含表头信息（由 th 元素创建）  标准单元格 - 包含数据（由 td 元素创建）  th 元素内部的文本通常会呈现为居中的粗体文本，而 td 元素内的文本通常是左对齐的普通文本。 |
| 12. | <td> | <td> 标签定义 HTML 表格中的标准单元格。  HTML 表格有两类单元格：  表头单元 - 包含头部信息（由 th 元素创建）  标准单元 - 包含数据（由 td 元素创建）  td 元素中的文本一般显示为正常字体且左对齐。  使用 colspan 和 rowspan 属性来实现内容横跨多个行或列。 |
| 13. | <br> | <br> 可插入一个简单的换行符。  <br> 标签是空标签（意味着它没有结束标签，因此这是错误的：<br></br>）。在 XHTML 中，把结束标签放在开始标签中，也就是 <br />。  请注意，<br> 标签只是简单地开始新的一行，而当浏览器遇到 <p> 标签时，通常会在相邻的段落之间插入一些垂直的间距。  clear 属性  如果您希望文本流在内联表格或图像的下一行继续输出，请使用 clear 属性，该属性有三个可选的值：left、right 或者 all，每个值都代表一个边界或两边的边界。  HTML <br> 标签的 clear 属性  通常情况下，<br> 标签会告诉浏览器立即停止当前的文本流，并在下一行的左边，或者在左对齐的内联图形或表格的右边开始继续输出文本流。  但是有时候，您也许希望当前的文本流在当前左边或右边的内联表格或图像的下面一行继续输出。  HTML 4 和 XHTML 通过 <br> 标签提供了这样的功能。它可以具有三个值：left、right 或者 all，每个值都代表一个边界或两边的边界。当指定的边界没有图像时，浏览器才会继续输出文本。  一般情况下，文本流中的其他行会在内联图像的底部显示，除非图形的 <img> 标签被设置为左对齐或右对齐（对于 <table> 标签也一样）。因此，<br> 标签的 clear 属性只对这些左对齐或右对齐的图像或表格起作用。  在人物介绍的html文件中使用<br>标签：    浏览器中显示的效果： |
| 14. | <i> | <i> 标签显示斜体文本效果。  <i> 标签和基于内容的样式标签 <em> 类似。它告诉浏览器将包含其中的文本以斜体字（italic）或者倾斜（oblique）字体显示。如果这种斜体字对该浏览器不可用的话，可以使用高亮、反白或加下划线等样式。  提示：<i> 标签一定要和结束标签 </i> 结合起来使用。  在人物介绍的html文件中使用<br>标签：    浏览器中显示的效果： |
| 15. | <font> | <font> 规定文本的字体、字体尺寸、字体颜色。    在任务介绍中使用font标签：    在浏览器中呈现的效果： |
| 16. | <img> | img 元素向网页中嵌入一幅图像。  请注意，从技术上讲，<img> 标签并不会在网页中插入图像，而是从网页上链接图像。<img> 标签创建的是被引用图像的占位空间。  <img> 标签有两个必需的属性：src 属性 和 alt 属性。  HTML 与 XHTML 之间的差异  在 HTML 中，<img> 标签没有结束标签。  在 XHTML 中，<img> 标签必须被正确地关闭。  在 HTML 4.01 中，不推荐使用 image 元素的 "align"、"border"、"hspace" 以及 "vspace" 属性。  在 XHTML 1.0 Strict DTD 中，不支持 image 元素的 "align"、"border"、"hspace" 以及 "vspace" 属性。    人物介绍的html文件中使用<img>标签:    在浏览器中显示的效果 |
|  |  |  |

一、httpd的安装

演示环境及版本

# cat /etc/issue

CentOS release 6.5 (Final)

Kernel \r on an \m

# uname -r

2.6.32-431.el6.x86\_64

查看httpd是否已安装

# rpm -qa httpd

使用yum列出相关httpd安装文件，此处为本地yum源

# yum list |grep httpd

httpd.x86\_64 2.2.15-29.el6.centos local\_server

httpd-devel.i686 2.2.15-29.el6.centos local\_server

httpd-devel.x86\_64 2.2.15-29.el6.centos local\_server

httpd-manual.noarch 2.2.15-29.el6.centos local\_server

httpd-tools.x86\_64 2.2.15-29.el6.centos local\_server

安装及验证httpd

# yum -y install httpd

# rpm -qa |grep httpd

httpd-tools-2.2.15-29.el6.centos.x86\_64

httpd-2.2.15-29.el6.centos.x86\_64

使用rpm方式寻找配置文件

# rpm -qc httpd

常用的配置文件：

/etc/httpd/conf.d/\*.conf ###辅助配置文件

/etc/httpd/conf/httpd.conf ###主配置文件

/etc/sysconfig/httpd ###httpd工作模式配置文件

使用rpm方式查看安装位置及生成的二进制文件

# rpm -ql httpd

主程序：

/usr/sbin/httpd MPM模式默认

/usr/sbin/httpd.event

/usr/sbin/httpd.worker

使用rpm方式查看包的帮助文件

# rpm -qd httpd

启动脚本：/etc/rc.d/init.d/httpd

日志文件目录：

/var/log/httpd

access\_log：访问日志

error\_log: 错误日志

站点文档目录：（站点根目录）

/var/www/html

httpd的工作目录：/var/www

二、httpd的工作模式

1、MPM： Multipath Processing Module（多路处理模块）

prefork: 多进程模型，每个进程响应一个请求；稳定性好，但并发能力有限；预先生成多个空闲进程；

由于prefork使用select()系统调用，所以最大并发不能超过1024；

worker：多进程模型，每个进程可生成多个线程，每个线程响应一个请求；预先生成多个空闲线程；

event：一个进程直接响应n个请求；可同时启动多个进程；

httpd-2.2: 测试使用； ### Author : Leshami

httpd-2.4: 可生产使用； ### Blog : http://blog.csdn.net/leshami

2、几种工作方式的切换

prefork模式下

# service httpd start

Starting httpd: [ OK ]

# ps -ef|grep httpd |grep -v grep ###一个主进程，生成了8个空闲进程

root 6413 1 0 09:40 ? 00:00:00 /usr/sbin/httpd

apache 6416 6413 0 09:40 ? 00:00:00 /usr/sbin/httpd

apache 6417 6413 0 09:40 ? 00:00:00 /usr/sbin/httpd

apache 6418 6413 0 09:40 ? 00:00:00 /usr/sbin/httpd

apache 6419 6413 0 09:40 ? 00:00:00 /usr/sbin/httpd

apache 6420 6413 0 09:40 ? 00:00:00 /usr/sbin/httpd

apache 6421 6413 0 09:40 ? 00:00:00 /usr/sbin/httpd

apache 6422 6413 0 09:40 ? 00:00:00 /usr/sbin/httpd

apache 6423 6413 0 09:40 ? 00:00:00 /usr/sbin/httpd

# ss -tulpn |grep httpd

tcp LISTEN 0 128 :::80 :::\* users:(("httpd",6413,4),("httpd",6416,4),("httpd",6417,4),("httpd",6418,4),

("httpd",6419,4),("httpd",6420,4),("httpd",6421,4),("httpd",6422,4),("httpd",6423,4)

# netstat -nltp|grep 80

tcp 0 0 :::80 :::\* LISTEN 6413/httpd

worker工作方式

# cat /etc/sysconfig/httpd|grep -v ^#

HTTPD=/usr/sbin/httpd.worker

# service httpd restart

Stopping httpd: [ OK ]

Starting httpd: [ OK ]

[root@orasrv1 ~]# ps -ef|grep httpd|grep -v grep

root 2261 1 0 17:47 ? 00:00:00 /usr/sbin/httpd.worker

apache 2264 2261 0 17:47 ? 00:00:00 /usr/sbin/httpd.worker

apache 2265 2261 0 17:47 ? 00:00:00 /usr/sbin/httpd.worker

apache 2266 2261 0 17:47 ? 00:00:00 /usr/sbin/httpd.worker

event工作方式

注，尽管2.2版本可以设置工作方式为httpd.event，生产环境不建议使用

# cat /etc/sysconfig/httpd|grep -v ^#

HTTPD=/usr/sbin/httpd.event

[root@orasrv1 ~]# service httpd restart

Stopping httpd: [ OK ]

Starting httpd: [ OK ]

[root@orasrv1 ~]# ps -ef|grep httpd|grep -v grep

root 2402 1 0 17:49 ? 00:00:00 /usr/sbin/httpd.event

apache 2405 2402 0 17:49 ? 00:00:00 /usr/sbin/httpd.event

apache 2406 2402 0 17:49 ? 00:00:00 /usr/sbin/httpd.event

apache 2407 2402 0 17:49 ? 00:00:00 /usr/sbin/httpd.event

三、httpd模块化特性

高度模块化

core + modules,

DSO: Dynamic Shared Object

模块目录：

/etc/httpd/modules: 符号链接文件

/usr/lib64/httpd/modules

模块的查看

httpd -M ###查看当前httpd进程的所有模块

httpd.event -M ###查看event工作模式下的所有模块 更正@20160712

httpd.worker -M ###worker工作模式下的所有模块 更正@20160712

httpd.worker -l ###worker工作模式下的核心模块 更正@20160712

模块的查看示例

# httpd -M

Loaded Modules:

core\_module (static)

mpm\_prefork\_module (static)

http\_module (static)

so\_module (static)

auth\_basic\_module (shared)

..............

# httpd.event -l

Compiled in modules:

core.c

event.c

http\_core.c

mod\_so.c

模块的动态装载与卸载

# cp /etc/httpd/conf/httpd.conf /etc/httpd/conf/httpd.conf.bak

# cat /etc/httpd/conf/httpd.conf |grep authn\_alias\_module

LoadModule authn\_alias\_module modules/mod\_authn\_alias.so

# vi /etc/httpd/conf/httpd.conf ###注释mod\_authn\_alias.so模块

# cat /etc/httpd/conf/httpd.conf |grep authn\_alias\_module

#LoadModule authn\_alias\_module modules/mod\_authn\_alias.so

# service httpd restart

# httpd -M ###使用该方式前后进行对比即可知道模块是否装载或卸载

四、验证httpd服务

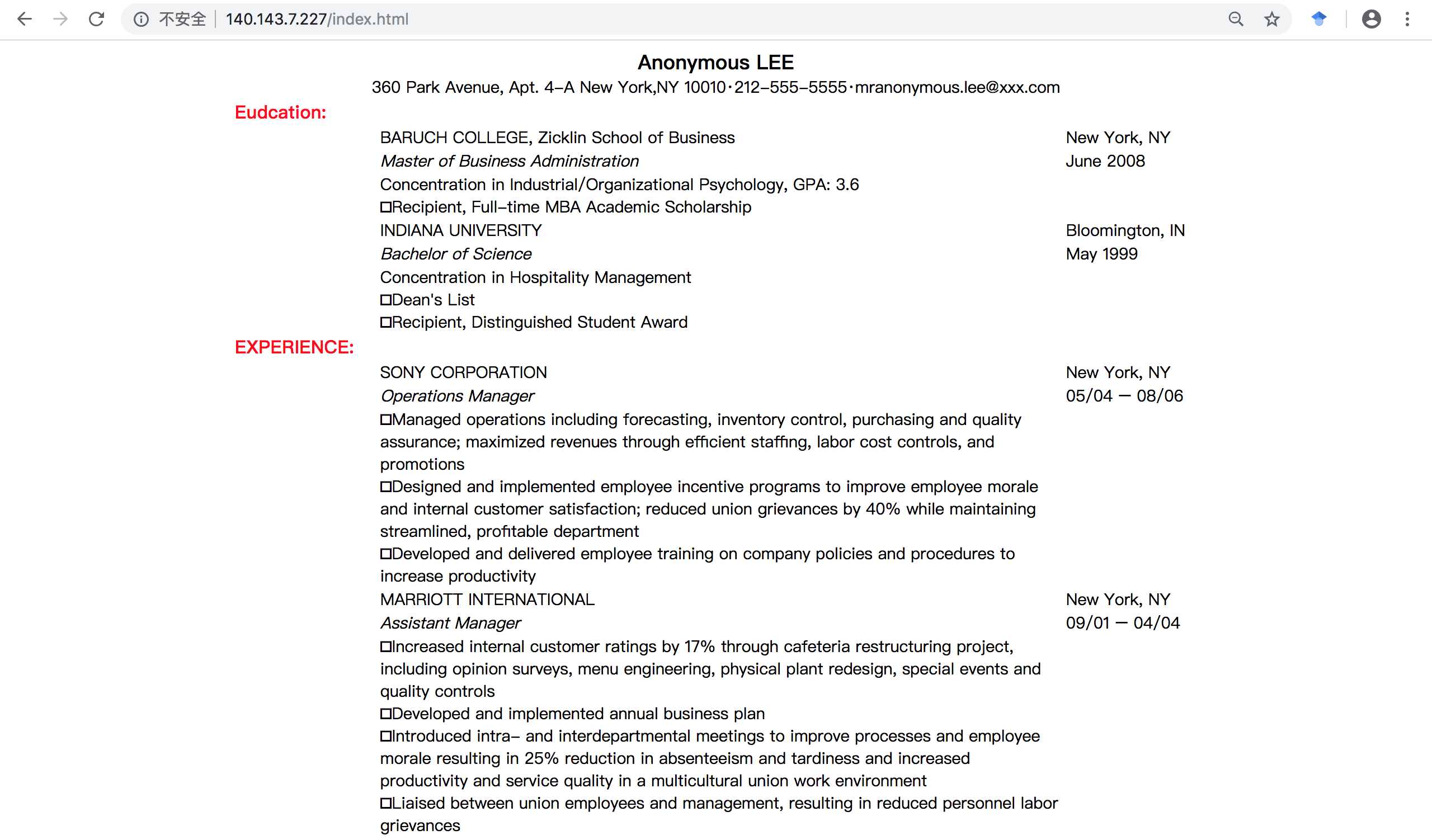
# echo "<h1>orasrv1.xlk.com</h1>" >/var/www/html/index.html

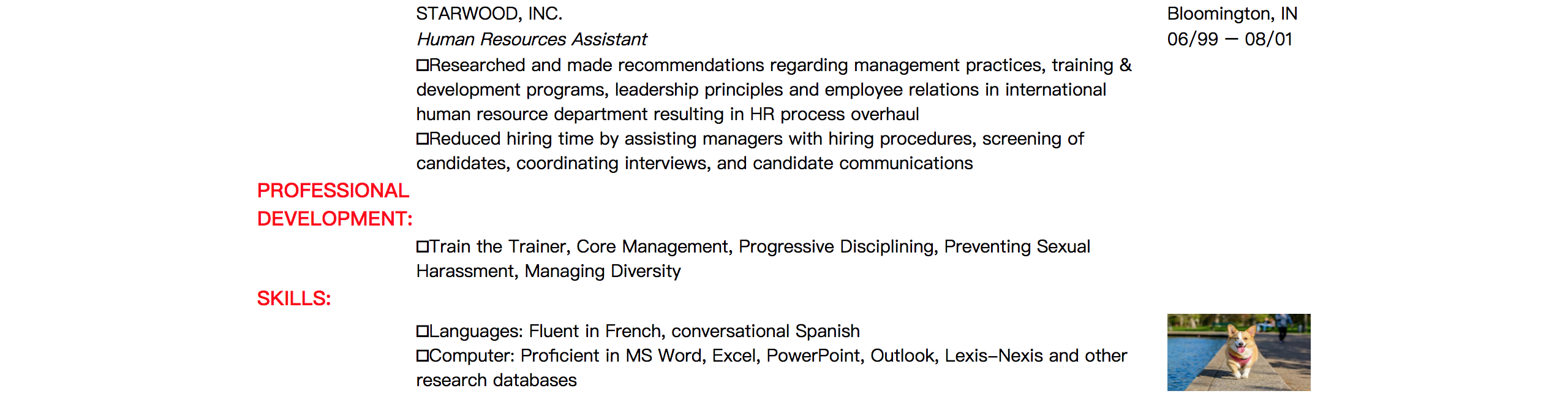
# curl http://192.168.21.10

<h1>orasrv1.xlk.com</h1>

From <https://cloud.tencent.com/developer/article/1183585>

Html成品展示：





#### 参考资料：

ssh和scp的简单使用：

#### SSH 客户端的简单使用:

**使用ssh客户端远程登录到指定的计算机**

$ ssh [-p port] user@remote

* user 是在远程机器上的用户名，如果不指定的话默认为当前用户
* remote 是远程机器的地址，可以是 **IP**／**域名**，或者是 **后面会提到的别名**
* port 是 **SSH Server 监听的端口**，SSH 服务器的默认端口号是 22，不指定则使用默认端口号

提示：

* 使用 exit 退出当前用户的登录
* 在工作中，SSH 服务器的端口号很有可能**不是 22**，如果遇到这种情况就需要使用 -p 选项，指定正确的端口号，否则无法正常连接到服务器

### scp（掌握）

* scp 就是 secure copy，是一个在 Linux 下用来进行 **远程拷贝文件** 的命令
* 既可以将本地内容拷贝到远程计算机,也可以将远程计算机内容拷贝到本地

| **命令** | **对应英文** | **作用** |
| --- | --- | --- |
| scp 用户名@ip:文件名或路径 用户名@ip:文件名或路径 | secure copy | 远程复制文件 |

| **选项** | **含义** |
| --- | --- |
| -r | 若给出的源文件是目录文件，则 scp 将递归复制该目录下的所有子目录和文件，目标文件必须为一个目录名 |
| -P | 若远程 SSH 服务器的端口不是 22，需要使用大写字母 -P 选项指定端口 |

* 它的**地址格式与 ssh 基本相同**，**需要注意的是**，在指定端口时用的是大写的 -P 而不是小写的
* IP地址的\*\*:\*\* 后面的路径如果不是绝对路径，则**以用户的家目录**作为参照路径

# 把本地当前目录下的 01.py 文件 复制到 远程 家目录下的 Desktop/01.py

scp -P port 01.py user@remote:Desktop/01.py

# 把远程 家目录下的 Desktop/01.py 文件 复制到 本地当前目录下的 01.py

scp -P port user@remote:Desktop/01.py 01.py

# 加上 -r 选项可以传送文件夹# 把当前目录下的 demo 文件夹 复制到 远程 家目录下的 Desktop

scp -r demo user@remote:Desktop

# 把远程 家目录下的 Desktop 复制到 当前目录下的 demo 文件夹

scp -r user@remote:Desktop demo

注意：

* scp 这个终端命令只能在 Linux 或者 UNIX 系统下使用
* 如果在 Windows 系统中，可以安装 PuTTY，使用 pscp 命令行工具或者安装 FileZilla 使用 FTP 进行文件传输

### SSH 高级:

* 免密码登录
* 配置别名

提示：有关 SSH 配置信息都保存在用户家目录下的 .ssh 目录下

#### 1）免密码登录

##### 步骤

* 配置公钥
  + 执行 ssh-keygen 即可生成 SSH 钥匙，一路回车即可
* 上传公钥到服务器
  + 执行 ssh-copy-id -p port user@remote，可以让远程服务器记住我们的公钥

##### 示意图

非对称加密算法

* 使用 **公钥** 加密的数据，需要使用 **私钥** 解密
* 使用 **私钥** 加密的数据，需要使用 **公钥** 解密

#### 2) 配置别名

每次都输入 ssh -p port user@remote，时间久了会觉得很麻烦，特别是当 user, remote 和 port 都得输入，而且还不好记忆

而 **配置别名** 可以让我们进一步偷懒，譬如用：ssh ubuntu 来替代上面这么一长串，那么就在**本地计算机**的 ~/.ssh/config 里面**追加**以下内容：

Host ubuntu

HostName 服务器ip地址

User python

Port 22

**保存之后，即可用** ssh ubuntu**实现远程登录了，**scp **同样可以使用**

<http://www.cnblogs.com/wangchaowei/p/6852339.html>

其中用户名@服务器ip 现在可以用别名 centos 来代替

通过ssh操作的说明<https://www.jianshu.com/p/460b4ce4f8e1>