登录

2019年05月04日 阅读 576

关注

Flask框架从入门到精通之消息闪现和日志记录（二十一）

> 知识点：1、闪现 2、日志

一、概况

Flask提供了一个功能方便向前端反馈消息，这个功能叫闪现。闪现的功能是基于session实现，所以我们在使用闪现的时候需要配置secret_key。

二、使用

我们用一个上传图片的例子来实现闪现，当我们上传图片成功后，给前端反馈一个消息。

复制代码

```
from flask import Flask, render_template, flash, request, url_for, redirect
import os

app = Flask(__name__)

app.config["SECRET_KEY"] = "python is good"

UPLOAD_FOLDER = os.path.join(os.getcwd(), 'media')  # 图片上传路径 = 当前工作目录+media文件夹


@app.route('/')
def index():
    return render_template('upload.html')


@app.route('/upload', methods=['GET', 'POST'])  # 支持get、post请求
def upload():  # 视图函数
    file = request.files.get('file')  # files获取多媒体资源
    filename = file.filename
    file.save(os.path.join(UPLOAD_FOLDER, filename))  # 保存
    flash('上传成功')  # 添加闪现信息
    return redirect(url_for('index'))


if __name__ == '__main__':
    # 0.0.0.0代表任何能代表这台机器的地址都可以访问
    app.run(host='0.0.0.0', port=5000, debug=True）# 运行程序
```

前端代码

复制代码

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>

{#获取闪现#}
{% for msg in get_flashed_messages() %}
    <h1>{{ msg }}</h1>
{% endfor %}

<form action="/upload" method="post" enctype="multipart/form-data">
    <input type="file" name="file">
    <input type="submit" value="上传">
</form>

</body>
</html>
```

选择一张图片，点击上传：

当刷新浏览器的时候，闪现消息消失。

三、分类闪现

我们可以针对不同的场景，提供不同的分类。从而达到不同的反馈消息的样式不同。添加闪现消息的时候加个分类：

复制代码

```
flash('上传成功', 'success') # 添加闪现信息
```

前端获取的时候可以获取分类：

复制代码

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>

{#获取闪现#}
{% for msg in get_flashed_messages(category_filter=['success']) %}
    <h1 style="color: green">{{ msg }}</h1>
{% endfor %}

<form action="/upload" method="post" enctype="multipart/form-data">
    <input type="file" name="file">
    <input type="submit" value="上传">
</form>

</body>
</html>
```

我们在浏览器调试一下：

四、日志

- ERROR：这个级别的日志意味着系统中发生了非常严重的问题，必须有人马上处理，比如数据库不可用了，系统的关键业务流程走不下去了等等。很多人在实际开发的时候，不会去区分问题的重要程度，只要有问题就error记录下来，其实这样是非常不负责任的，因为对于成熟的系统，都会有一套完整的报错机制，那这个错误信息什么时候需要发出来，很多都是依据单位时间内error日志的数量来确定的。因此如果我们不分轻重缓急，一律error对待，就会徒增报错的频率，久而久之，我们的救火队员对错误警报就不会那么在意，这个警报也就失去了原始的意义。

- WARN：发生这个级别的问题时，处理过程可以继续，但必须要对这个问题给予额外的关注。假设我们现在有一个系统，希望用户每一个月更换一次密码，而到期后，如果用户没有更新密码我们还要让用户可以继续登录，这种情况下，我们在记录日志时就需要使用WARN级别了，也就是允许这种情况存在，但必须及时做跟踪检查。

- INFO：这个级别的日志我们用的也是比较多，它一般的使用场景是重要的业务处理已经结束，我们通过这些INFO级别的日志信息，可以很快的了解应用正在做什么。我们以在12306上买火车票为例，对每一张票对应一个INFO信息描述“[who] booked ticket from [where] to [where]”。

- DEBUG和TRACE：我们把这两个级别放在一起说，是应为这两个级别的日志是只限于开发人员使用的，用来在开发过程中进行调试，但是其实我们有时候很难将DEBUG和TRACE区分开来，一般情况下，我们使用DEBUG足以。

复制代码

```
app.logger.debug('A value for debugging')
```

```
app.logger.warning('A warning occurred (%d apples)', 42)
app.logger.error('An error occurred')
```

如，放在我们上面的代码中，我们希望在上传成功之后，把这条信息记录一下：

```
app.logger.info(filename + '上传成功了.....')
```
复制代码

欢迎关注我的公众号：

文章分类

后端

文章标签

Flask

相关推荐