# Local对象：
在Flask中，类似于request的对象，其实是绑定到了一个werkzeug.local.Local对象上。这样，即使是同一个对象，那么在多个线程中都是隔离的。类似的对象还有session以及g对象。
# Thread Local对象：
只要满足绑定到这个对象上的属性，在每个线程中都是隔离的，那么他就叫做Thread Local对象。

# 应用上下文和请求上下文：
应用上下文和请求上下文都是存放到一个LocalStack的栈中。和应用app相关的操作就必须要用到应用上下文，比如通过current_app获取当前的这个app。和请求相关的操作就必须用到请求上下文，比如使用url_for反转视图函数。
1. 在视图函数中，不用担心上下文的问题。因为视图函数要执行，那么肯定是通过访问url的方式执行的，那么这种情况下，Flask底层就已经自动的帮我们把请求上下文和应用上下文都推入到了相应的栈中。
2. 如果想要在视图函数外面执行相关的操作，比如获取当前的app(current_app)，或者是反转url，那么就必须要手动推入相关的上下文：
    * 手动推入app上下文：
        ```
        # 第一种方式：
        app_context = app.app_context()
        app_context.push()
        # 第二种方式：
        with app.app_context():
            print(current_app)
        ```

    * 手动推入请求上下文：推入请求上下文到栈中，会首先判断有没有应用上下文，如果没有那么就会先推入应用上下文到栈中，然后再推入请求上下文到栈中：
        ```
        with app.test_request_context():
            print(url_for('my_list'))
        ```

# 为什么上下文需要放在栈中：
1. 应用上下文：Flask底层是基于werkzeug，werkzeug是可以包含多个app的，所以这时候用一个栈来保存。如果你在使用app1，那么app1应该是要在栈的顶部，如果用完了

app1，那么app1应该从栈中删除。方便其他代码使用下面的app。
2. 如果在写测试代码，或者离线脚本的时候，我们有时候可能需要创建多个请求上下文，这时候就需要存放到一个栈中了。使用哪个请求上下文的时候，就把对应的请求上下文放到栈的顶部，用完了就要把这个请求上下文从栈中移除掉。

# 保存全局对象的g对象：
g对象是在整个Flask应用运行期间都是可以使用的。并且他也是跟request一样，是线程隔离的。这个对象是专门用来存储开发者自己定义的一些数据，方便在整个Flask程序中都可以使用。一般使用就是，将一些经常会用到的数据绑定到上面，以后就直接从g上面取就可以了，而不需要通过传参的形式，这样更加方便。

# 常用的钩子函数：
钩子函数是系统级的消息处理函数, 通常在某些系统消息前触发
在Flask中钩子函数是使用特定的装饰器装饰的函数。为什么叫做钩子函数呢，是因为钩子函数可以在正常执行的代码中，插入一段自己想要执行的代码。那么这种函数就叫做钩子函数。（hook）
1. before_first_request：Flask项目第一次部署后会执行的钩子函数。
2. before_request：请求已经到达了Flask，但是还没有进入到具体的视图函数之前调用。一般这个就是在视图函数之前，我们可以把一些后面需要用到的数据先处理好，方便视图函数使用。
3. context_processor：使用这个钩子函数，必须返回一个字典。这个字典中的值在所有模版中都可以使用。这个钩子函数的函数是，如果一些在很多模版中都要用到的变量，那么就可以使用这个钩子函数来返回，而不用在每个视图函数中的render_template中去写，这样可以让代码更加简洁和好维护。
4. errorhandler：在发生一些异常的时候，比如404错误，比如500错误。那么如果想要优雅的处理这些错误，就可以使用errorhandler来出来。需要注意几点：
    * 在errorhandler装饰的钩子函数下，记得要返回相应的状态码。
    * 在errorhandler装饰的钩子函数中，必须要写一个参数，来接收错误的信息，如果没有参数，就会直接报错。
    * 使用flask.abort可以手动的抛出相应的错误，比如开发者在发现参数不正确的时候可以自己手动的抛出一个400错误。
示例代码如下：
@app.errorhandler(404)

```
def page_not_found(error):
    return render_template('404.html'),404
```