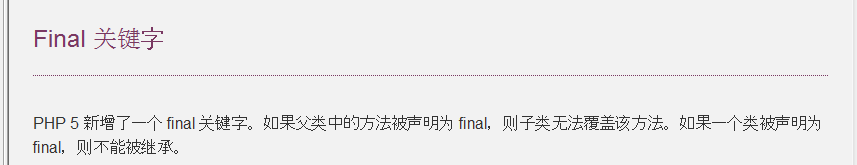
* 封装
  + 重载：在当前环境下，不可见的或者不存在的属性的使用，检测，销毁，赋值会自动调用重载方法
    - 4个魔术方法
      * \_\_set()
      * \_\_get()
      * \_\_isset()
      * \_\_unset()
    - 方法重载
      * \_\_call()
      * \_\_callStatic()
* 继承
  + 如何继承？继承的特点
    - 使用关键词extends
    - 特点：
      * 1.父类的属性和方法，继承之后，子类都会获得
      * 2，php是单继承
    - 继承是可以拿到所有的属性和方法，只不过，私有的，能继承，不能用
  + 重写
    - 为什么需要重写？
      * 继承到的方法，不适合子类使用，所以需要重写。
    - 重新的要求：
      * 1.修饰符的访问范围，子类至少要和父类相同
        + 父：public ,子：public
        + 父：protected, 子：public protected
        + 父：private 子：任意
      * 2.参数的数目必须一致，不能多，也不能少。构造方法除外
  + 构造方法的使用
    - 1.如果父类有，子类没有，默认子类调用父类的构造
    - 2.如果都有，默认子类调用自己的构造
    - 3.如果都有，子类想调用父类的构造，需要使用关键词：parent::\_\_construct()
    - 事实上，子类需要调用父类的方法的时候都可以使用parent::xx()
  + 访问控制
    - Public:所有
    - Protected:子类，类内
    - Private:类内

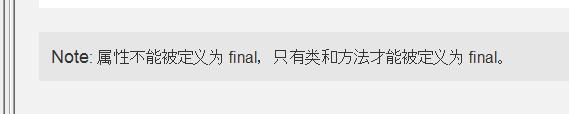
# OOP

## 继承

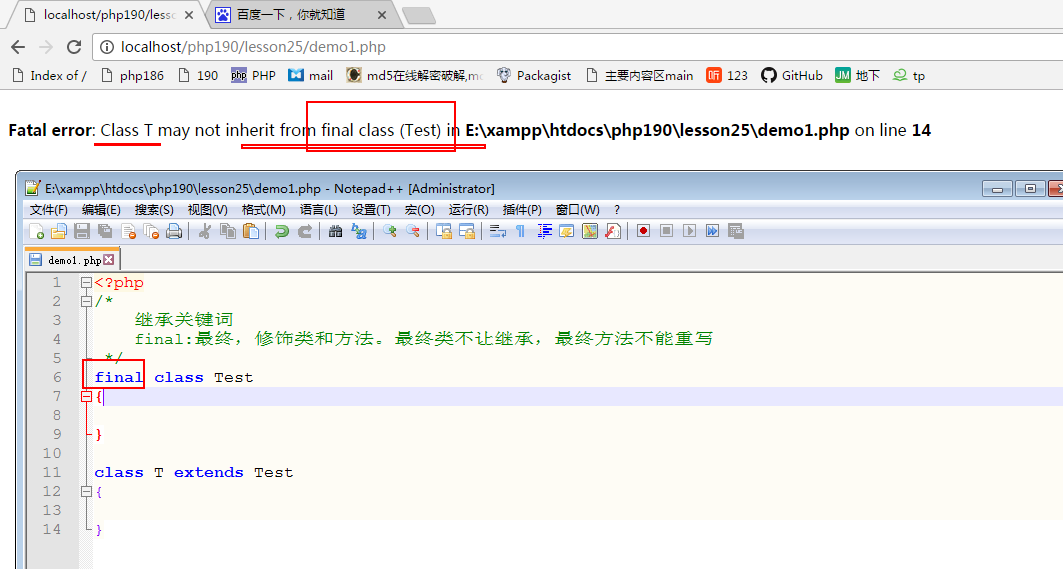
### Final

* 最后的，最终的
* **如果类不加以限制，是不是可以无限的继承，继承到最后，孙孙孙里面的属性和方法是不是好多都是没用的。那么能否对类进行控制,让他不能继承。可以。怎么玩？就是final**

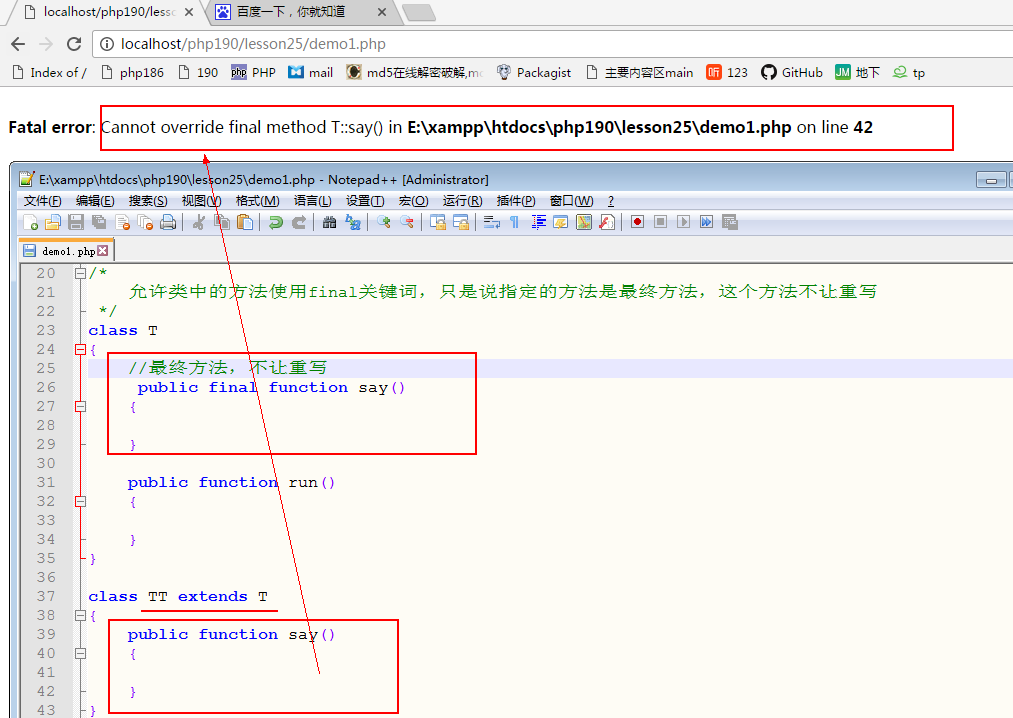




* **Final修饰类，方法**
  + **最终类，不让继承**

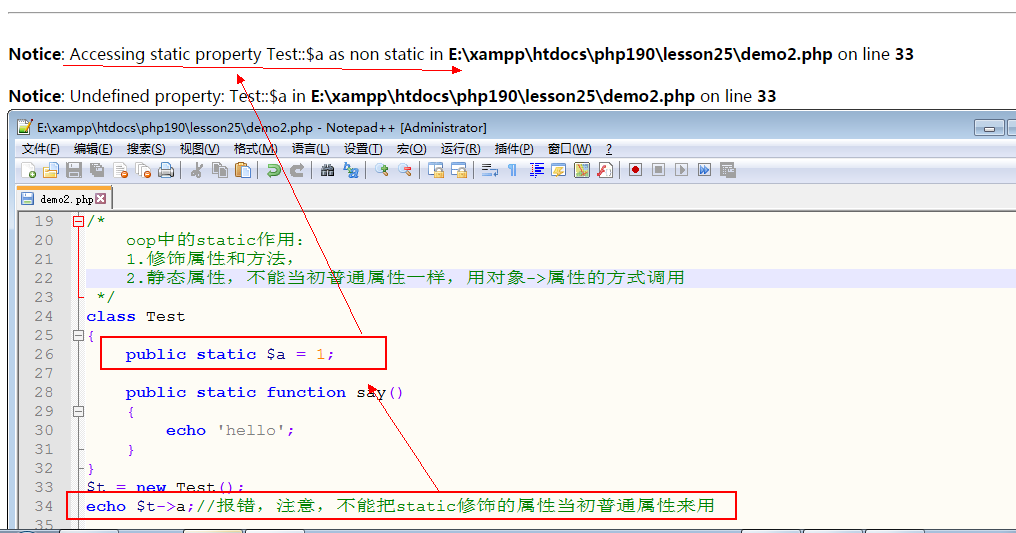


* + **最终方法，不让重写**

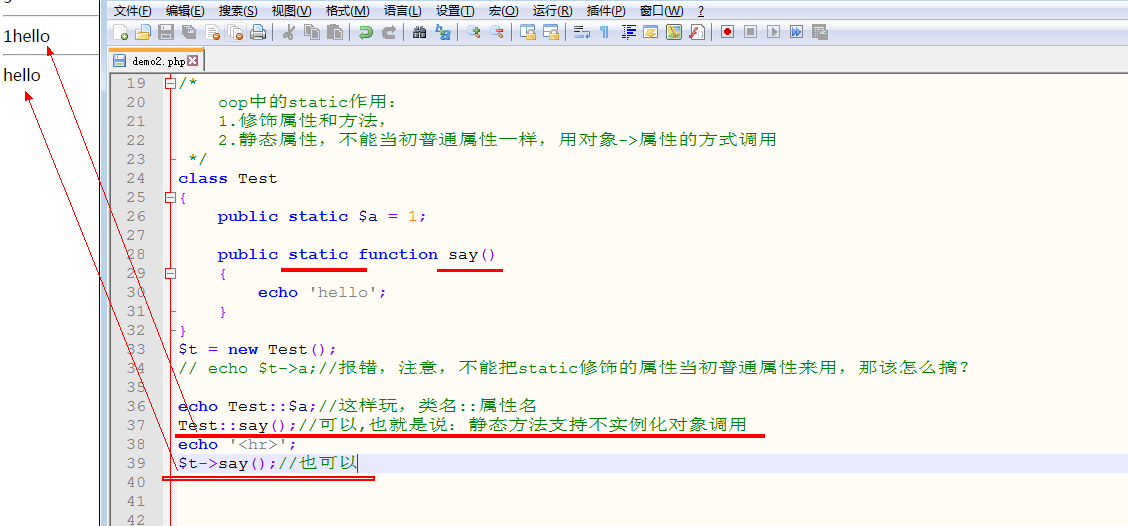


### Static

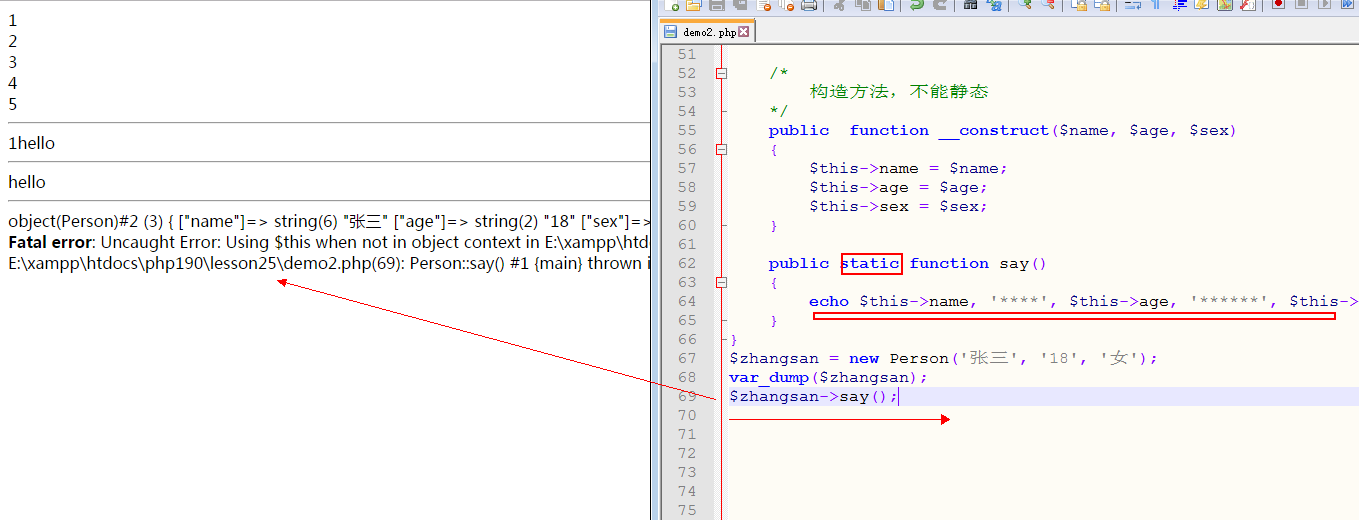
* 静态变量：
  + 特点：函数执行完成时，静态变量不会销毁，而是保留下来
* OOP中的static
* **1.只能修饰属性和方法，不能修饰类.**
* **2.静态属性，不能当初普通属性一样，使用对象->属性的方式调用**



* **3.静态方法，可以不实例化对象，直接调用，当然也可以实例化之后，拿对象去调用。直接用的方式为：类名::方法名（）**



* **4.静态方法内部，不能使用$this.**

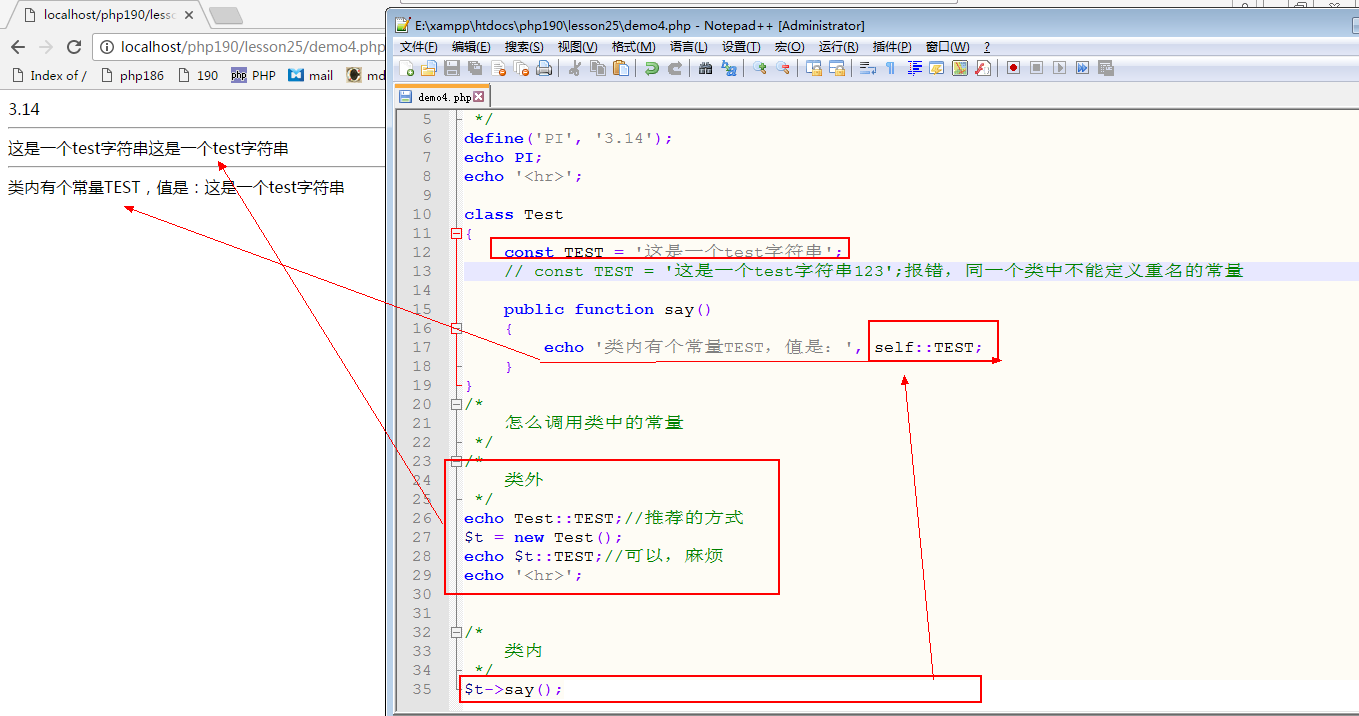


**而是用self来表示当前类**



### 类常量

* 类内部定义的常量。
* 注意：
  + **1.同一个类中，不能定义重名的常量**
  + **2.命名：全大写**
  + **3.规则：const 常量名 = ‘常量值’**

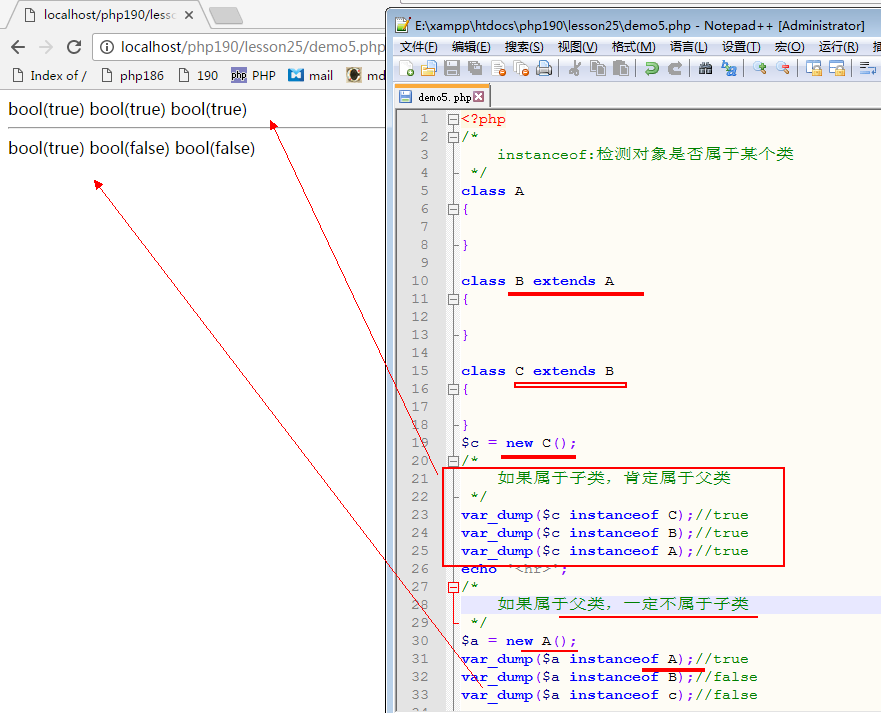


* **类常量的调用**
  + **类内：self::常量名**
  + **类外：类名：：常量名**
  + **和静态属性|方法是一样的**

### 检测

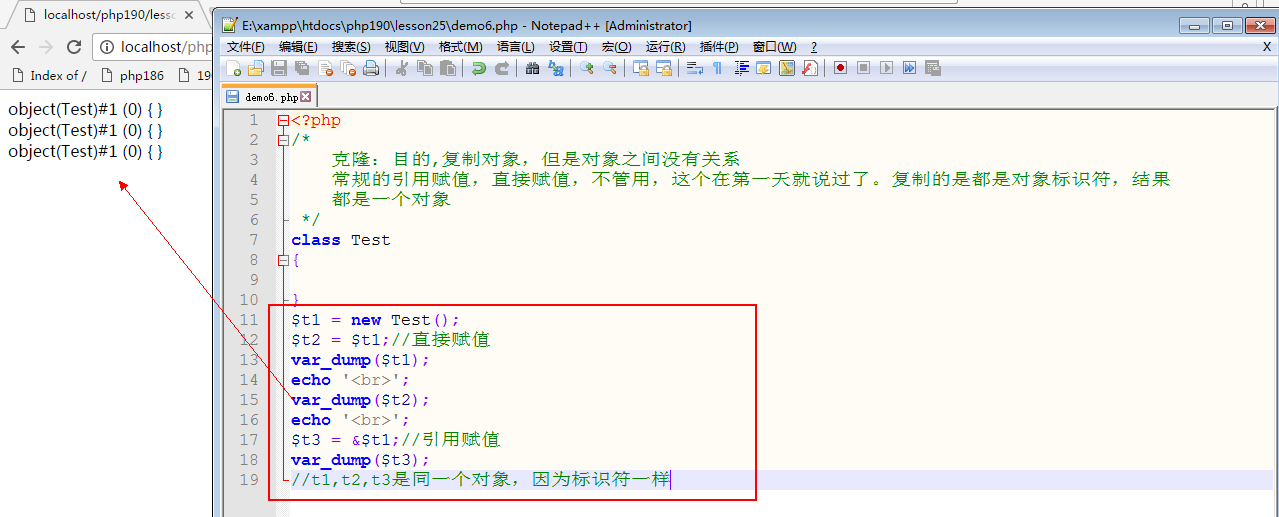
#### Instanceof

* 结果：bool
* **作用：检测对象是否属于某个类**
  + 举例：如果有3个类,A,B,C, A继承自B，B继承自C，如果对象b属于B，那么他肯定属于A，但是肯定不属于C
* 用法：
  + **$对象 instanceof 类名**
  + 结果是bool

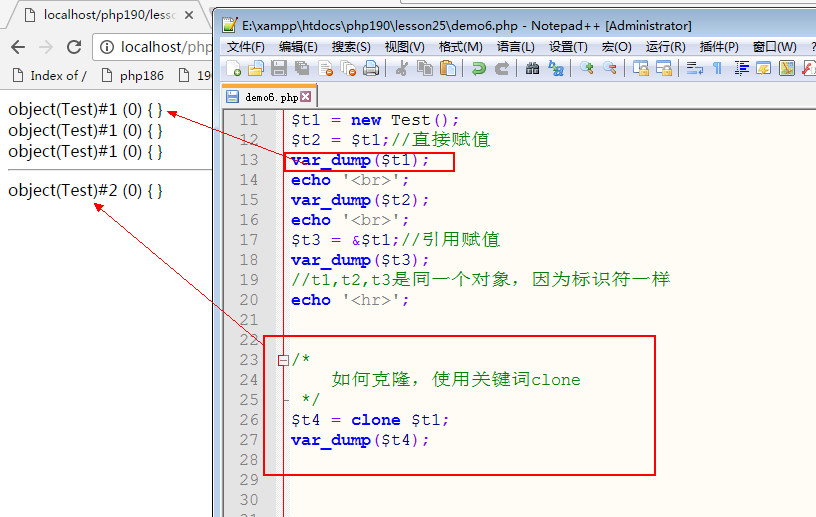


## 对象的复制|对象克隆

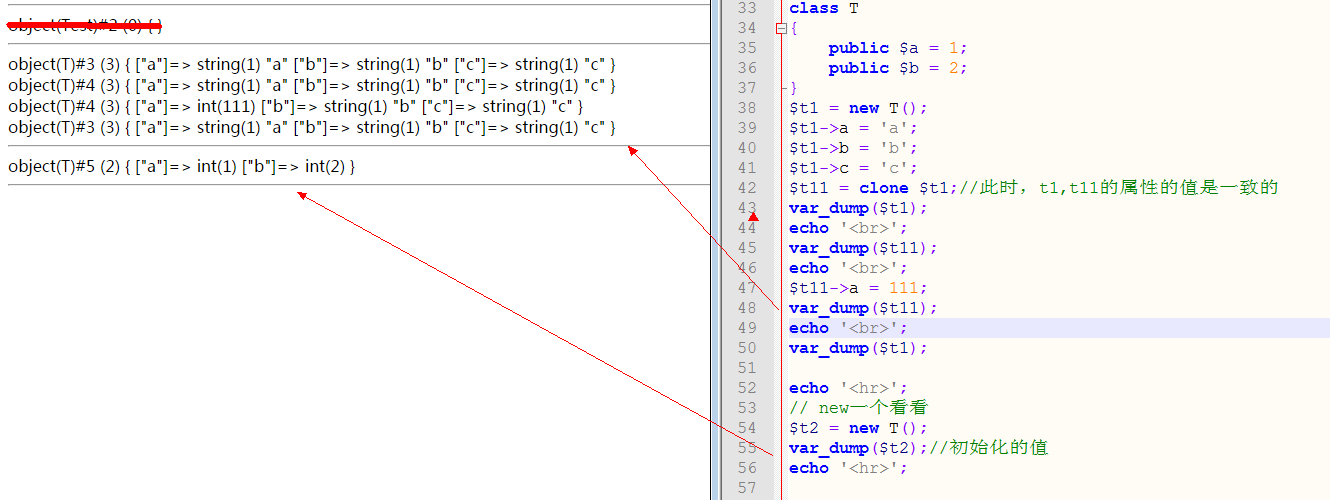
* 克隆：目的,复制对象，但是对象之间没有关系
* **注意：常规的引用赋值，直接赋值，不管用，这个在第一天就说过了。复制的是都是对象标识符，结果都是一个对象**



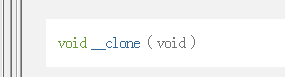
* 克隆方法
  + **使用clone关键词**
  + **就是在赋值的时候，多个clone关键词**



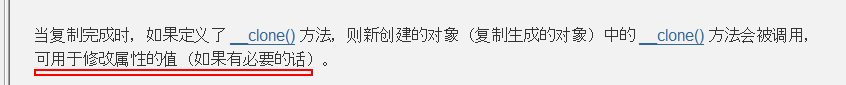
* **克隆的意义**
  + **可以复制某一时刻的对象的所有属性，而不需要初始化**



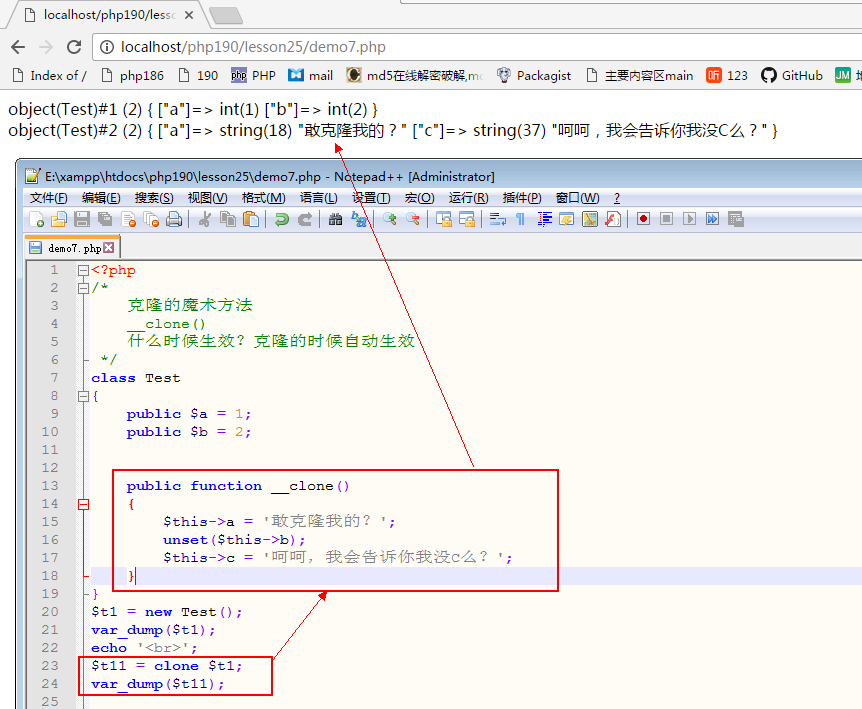
### 克隆的魔术方法\_\_clone()



* 没有返回值，也不需要参数



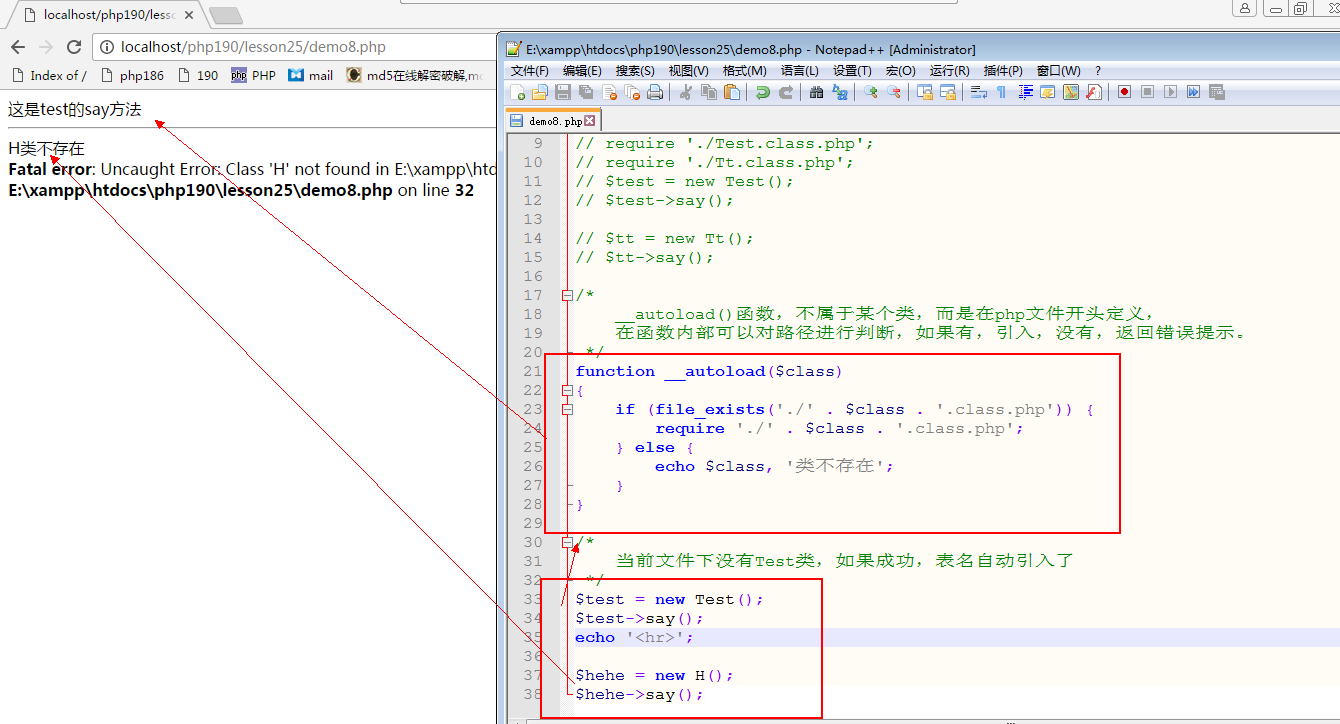
* 注意：\_\_clone()在clone对象的时候自动调用



* 注意：默认都是浅克隆

## 自动加载

* 优势：引入不存在的文件的时候，不需要挨个require，而是指定自动引入的方法，由方法去判断
* 方法1：
  + **\_\_autoload()**
  + **内部实现：判断类的目录，然后引入**



* 方法2：
  + **Spl\_autoload\_register()**



