第二章 PHP5 面向对象进阶

10 月 27 日，家里添了一个男孩，生活变的忙碌起来。照顾小宝宝是件辛苦的事情，也更明白父母的辛苦。工作之余，继续写起来时间越来越紧张。

祝老婆早日康复，祝我的小宝健康成长，祝父母身体健康。

如果看到代码有什么问题。可以到 phpchina.com
我的博客留言给我。http://www.phpchina.com/?10 。

再次感谢 www.phpchina.com

gaoxiangming@hotmail.com
刀客羽朋
石家庄 2006-11-19

目录

2.1 类的继承

什么是继承

前面说过，面向对象的思想和我们的生活是息息相通的。
我们先分析一个生活中的例子：
自行车、折叠车、电动车的关系。
例 1：

自行车有什么特征（属性）？
- 两个轱辘
- 一个车座
- 两个脚蹬子
- 有颜色

自行车有什么动作（方法）？
- 骑行
- 刹车

折叠自行车有什么特征（属性）？
- 两个轱辘
- 一个车座
- 两个脚蹬子
- 有颜色

折叠自行车有什么动作（方法）？
- 骑行
- 刹车
- **折叠**

电动自行车有什么特征（属性）？
- 两个轱辘
- 一个车座
- 两个脚蹬子
- 有颜色
- 电池一块

电动自行车有什么动作（方法）？
- 骑行
- 刹车
- **电动行驶**

上面的三个表格，说明了自行车、折叠自行车、电动自行车特性。
我们描述折叠自行车和电动自行车时，除红色标注的部分，都和自行车一样。

我们尝试用另外一种方式，建立模型的方式来描述一次。

例 2：

自行车有什么特征（属性）？

- 两个轱辘
- 一个车座
- 两个脚蹬子
- 有颜色

自行车有什么动作（方法）？

- 骑行
- 刹车

折叠自行车有什么特征（属性）

- **折叠自行车和自行车有相同的属性**

折叠自行车有什么动作（方法）

- **折叠自行车具有自行车的所有方法。**
- **增加了折叠方法。**

电动自行车和自行车有相同的属性和方法。

- 增加了电池一块
- **增加了电动行驶的方法。**

这次的描述变简单了，只需要将增加的内容填写上去。关于自行车的描述被**复用**了。
仔细再观察对自行车的描述，我们发现上面三个自行车都缺少了一个重要的属性”车主架”。
在例 1 中，我们要在三个描述中分别添加”车铃铛”，这个属性。
在例 2 中，我们只要在自行车的描述中加入属性“车铃铛”，另外两个描述不用变化就完成内容的添加。同样，动作（方法）的变化也很容易。

感觉到了什么了么？**它让我们的描述更容易“扩充和维护”。**

PHP5 中的继承

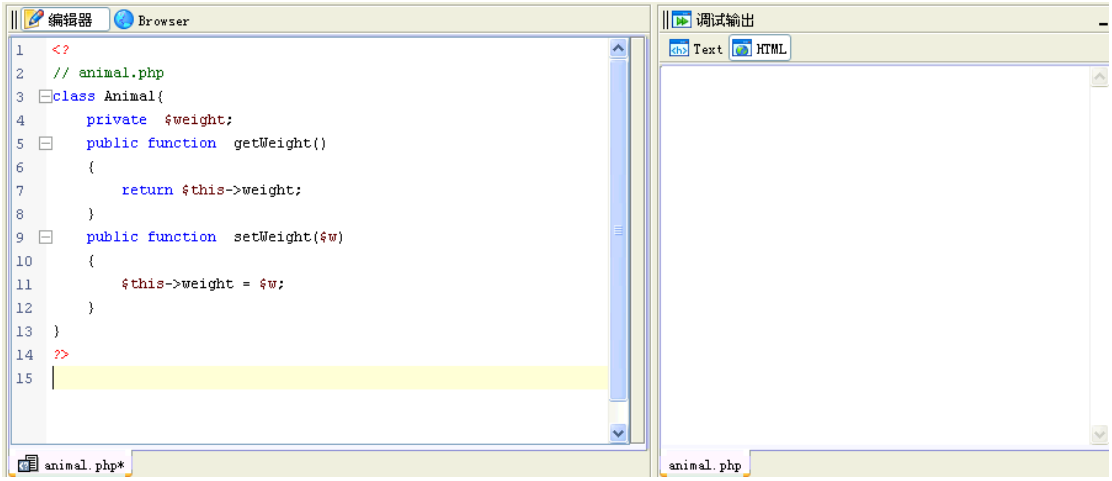
- 继承是面向对象最重要的特点之一，就是可以实现对类的复用。
- 通过“继承”一个现有的类，可以使用已经定义的类中的方法和属性。
- 继承而产生的类叫做子类。
- 被继承的类，叫做父类，也被成为超类。

- PHP 是单继承的，一个类只可以继承一个父类，但一个父类却可以被多个子类所继承。
- 从子类的角度看，它“ 继承（inherit ， extends）”自父类；而从父类的角度看，它“派生（derive）”子类。它们指的都是同一个动作，只是角度不同而已。
- 子类不能继承父类的私有属性和私有方法。
- 在 PHP5 中类的方法可以被继承，类的构造函数也能被继承。

继承的简单例子

我们分析自然界中的关系，动物类与犬类的关系。

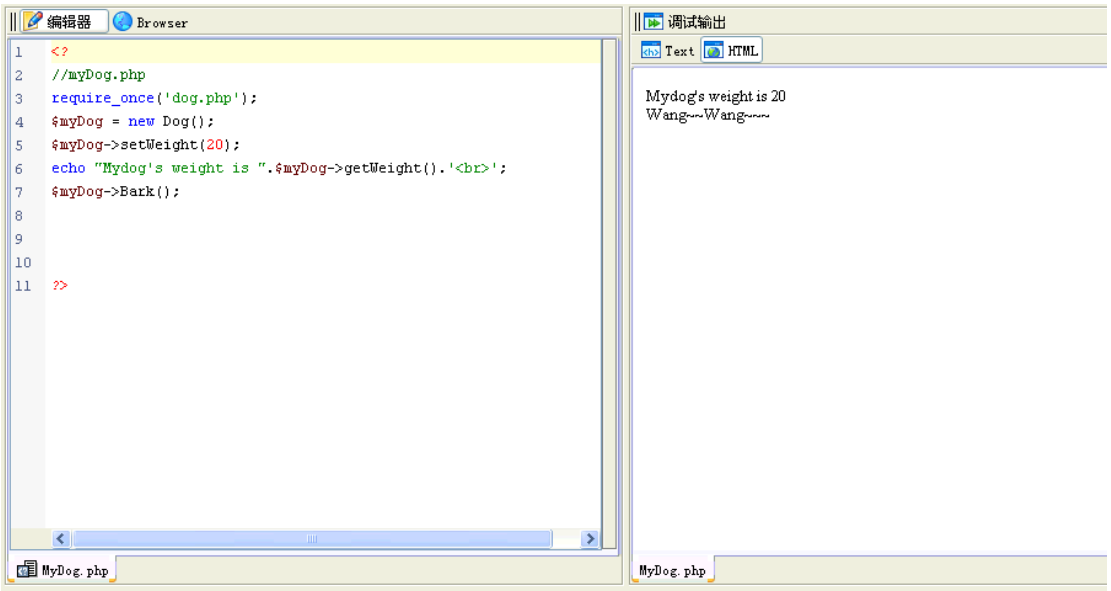
例 2-1　animal.php



Dog 类继承自 animal 类。
Dog.php

当我们实例化 animal 类的子类 Dog 类时， 父类的方法 setWeight（） 和 getWeight() 被继承。
我们可以直接调用父类的方法设置其属性$weight，取得其属性$weight 。

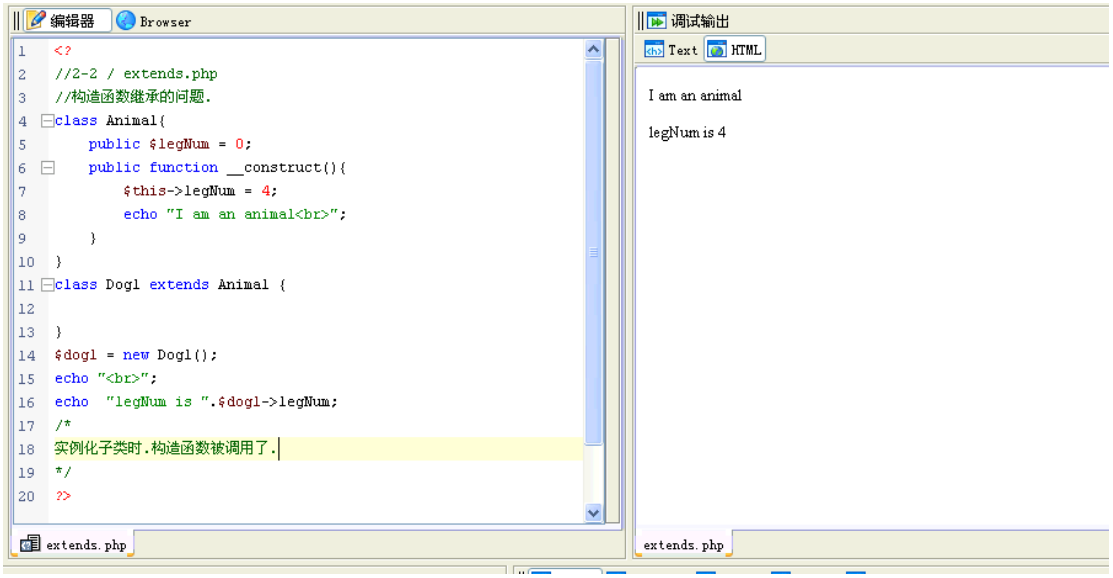
dog 类的实例。



构造函数的继承

有些资料上说 PHP5 的构造函数不被继承。
演示的结果证明，PHP5 的构造函数被继承了。

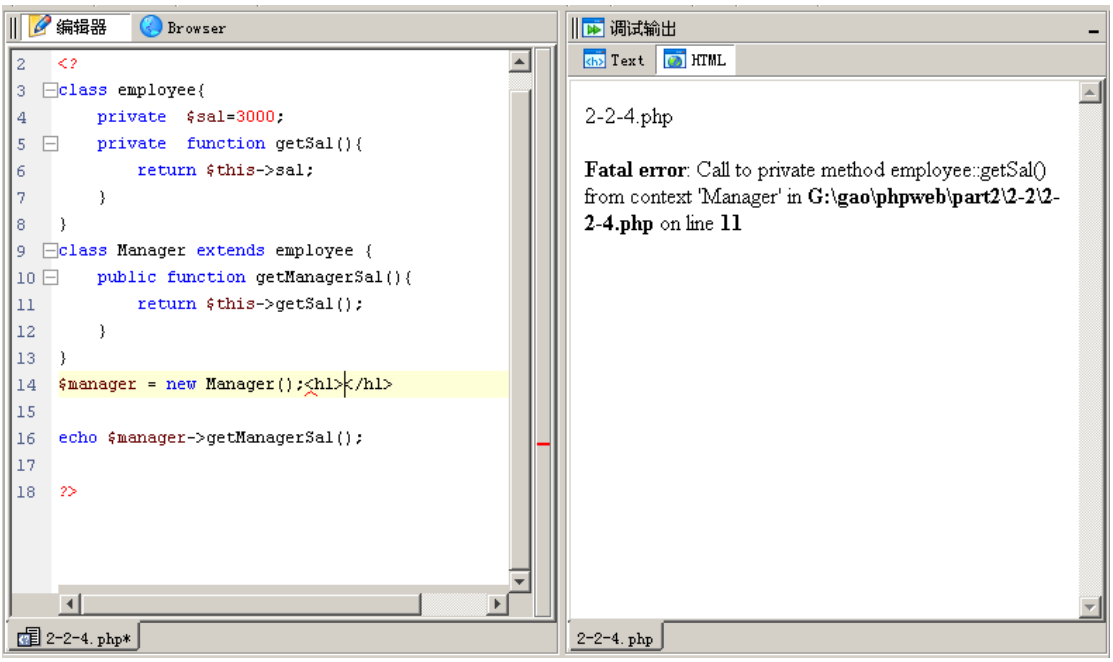
当子类 Dog1 被实例化时，继承的构造函数被调用了。
屏幕上显示了一句　" I am an Animal. " .



私有变量和方法不被继承

- 私有变量不被继承，私有方法也不被继承。
- 另外一种说法，私有变量和属性被继承了，但不能被调用。
- 无论怎么说，都不能调用父类的私有属性和私有方法。

例 2-2-4



2.2 访问控制

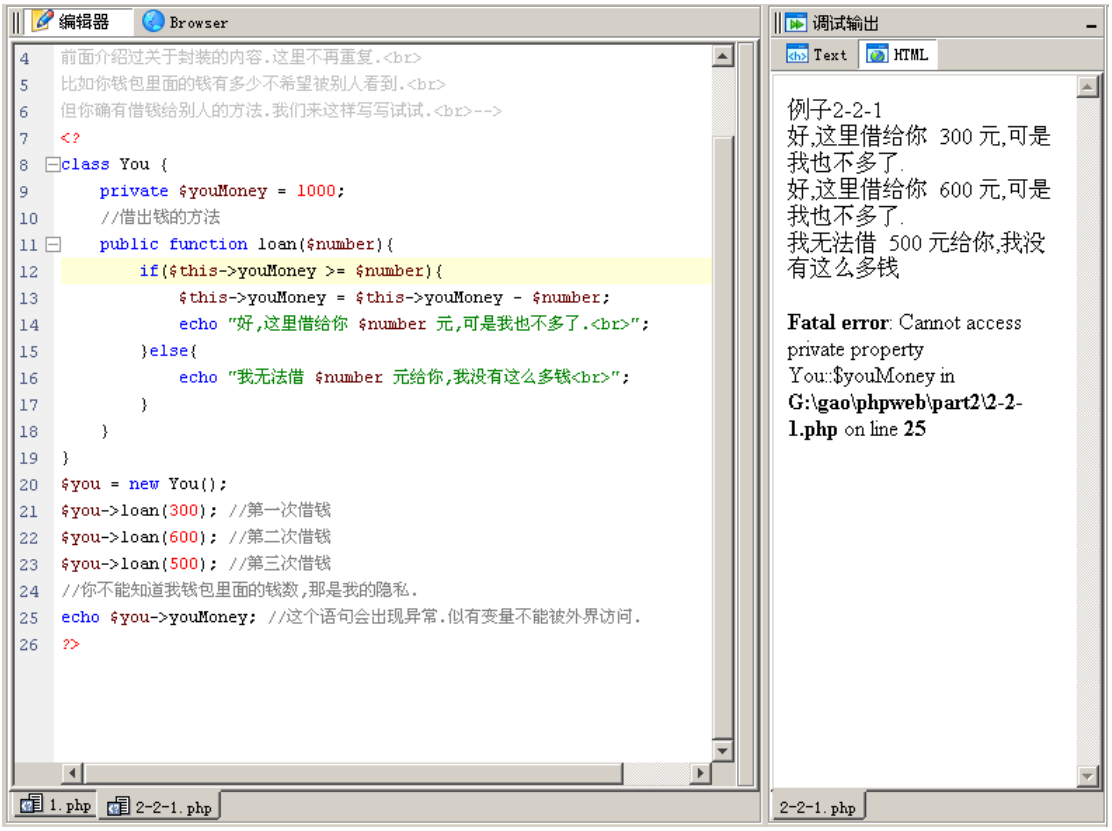
在 PHP5 中，可以在类的属性和方法前面加上一个修饰符（modifier），来对类进行一些访问上的控制。

下面表格显示了访问的权限。

修饰符	同一个类中	子类中	全局
private	Yes		
protected	Yes	Yes	
public	Yes	Yes	Yes(默认)

Private 的访问权限

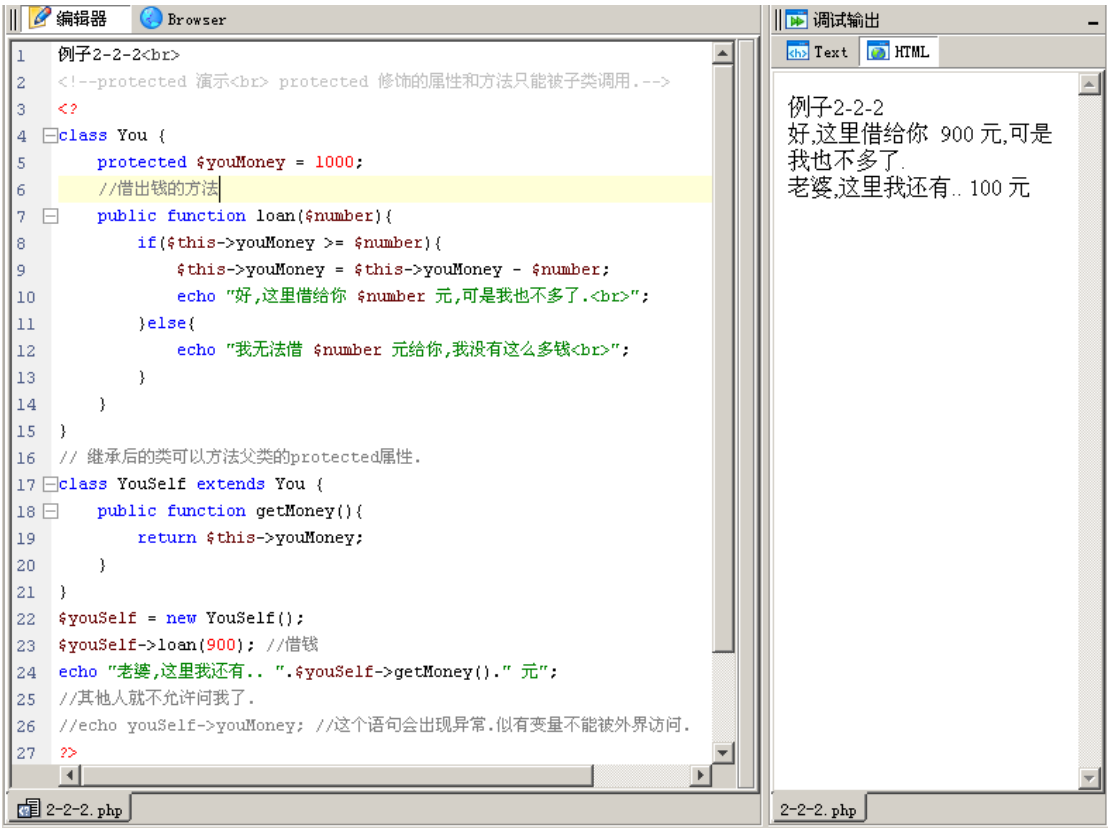
例 2-2-1
private 不能直接被外部调用，只能由当前对象调用。
前面介绍过关于封装的内容.这里不再重复。
比如你可以借钱给别人，但不希望别人知道你钱包里面有多少钱。
我们把它用 private 隐藏起来。



protected 的访问权限

例 2-2-2 演示

protected 修饰的属性和方法只能被子类调用。外界无法调用。



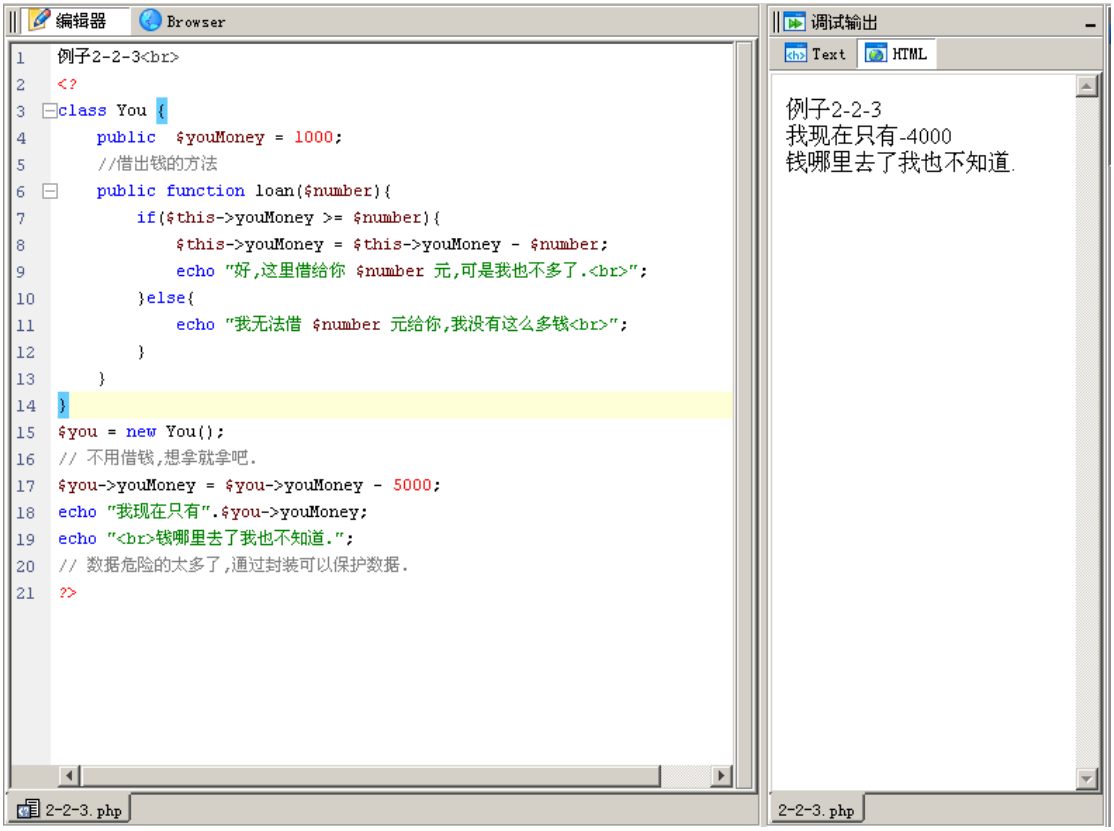
public 的访问权限

例子 2-2-3

数据的隐藏和封装是能够帮助我们保护数据的安全性。

Public 修饰的属性和方法，可以被无限制的调用。

嘿。。你的钱，不安全了。



2.3 重写（override）

•如果从父类继承的方法不能满足子类的需求，可以对其进行改写，这个过程叫方法的覆盖（**override**），也称为方法的重写。

•当对父类的方法进行重写时，子类中的方法必须和父类中对应的方法具有相同的方法名称，在 PHP5 中不限制输入参数类型、参数数量和返回值类型。（这点和 JAVA 不同）

•子类中的覆盖方法不能使用比父类中被覆盖方法更严格的访问权限。

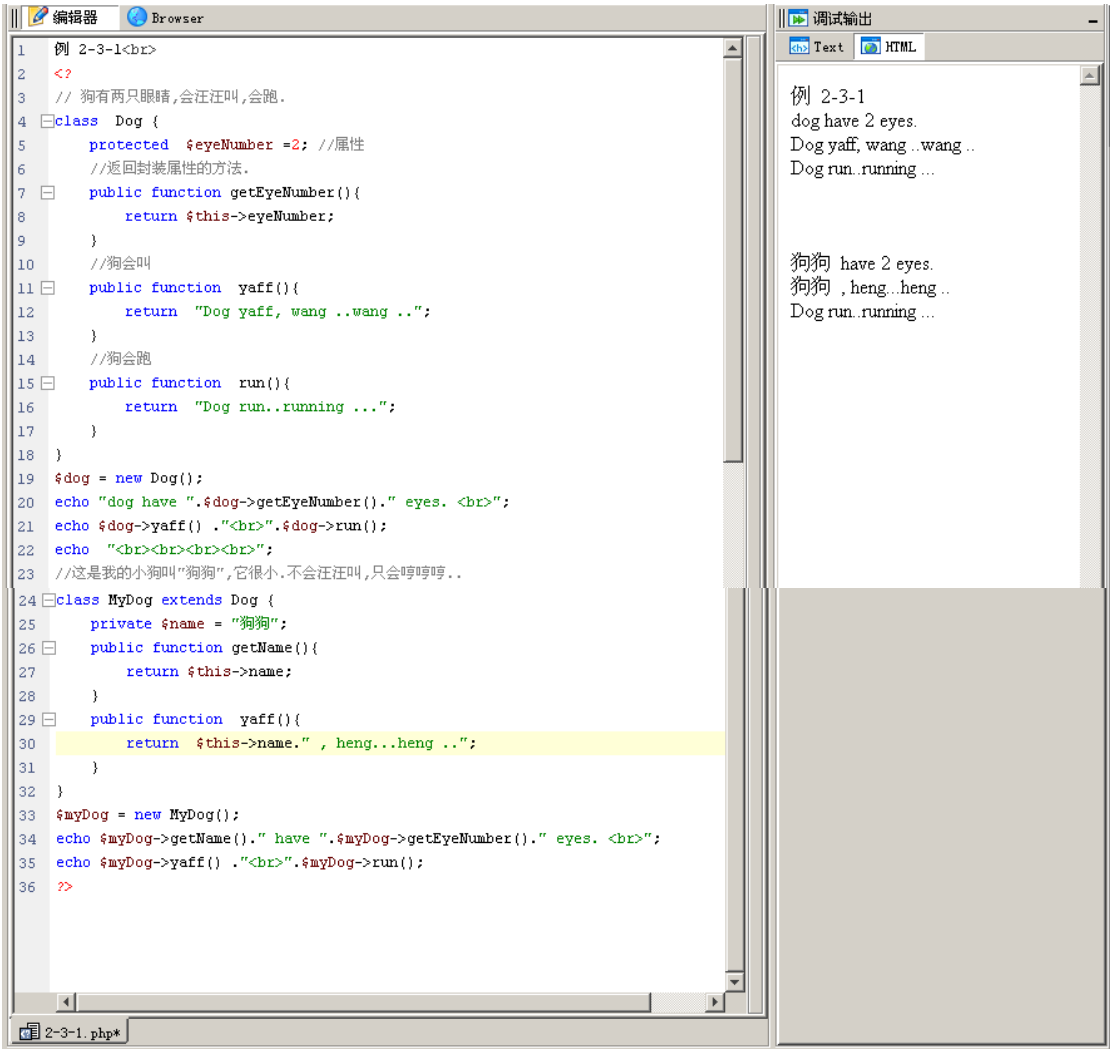
•声明方法时，如果不定义访问权限。默认权限为 public。

PHP5 重写方法

先设置一个父类，这个父类是 “Dog”类，这个类描述了 dog 的特性。
Dog 有 2 个眼睛，会跑，会叫。就这样描述先。

我养了一直狗，是只小狗，符合 Dog 类的特性，但有所不同。
我的小狗有名字，我的小狗太小了，不会大声的叫，只会哼哼。

我们用继承的概念去实现这个设计。
例子 2-3-1

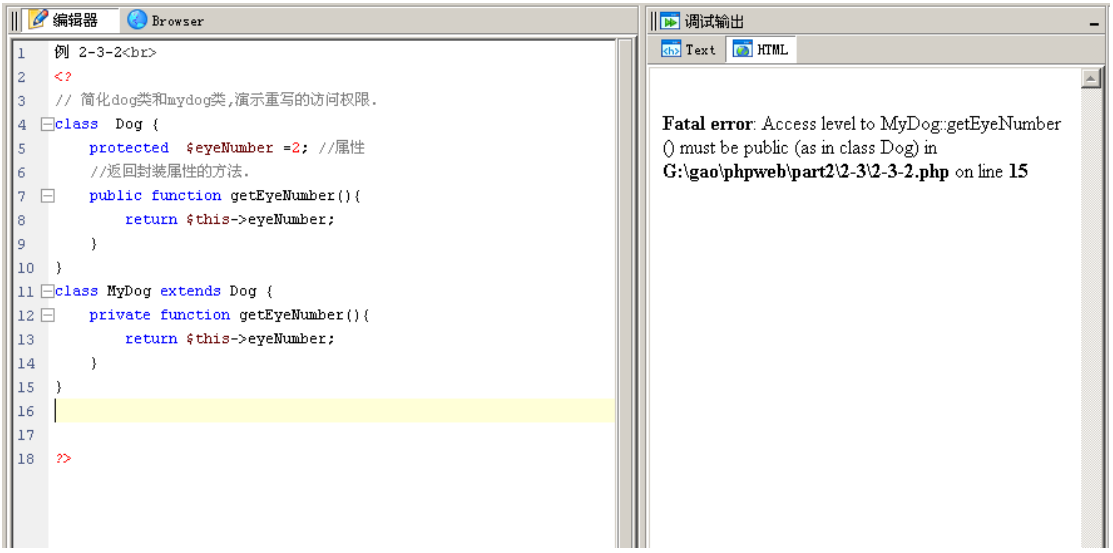


重写方法与访问权限

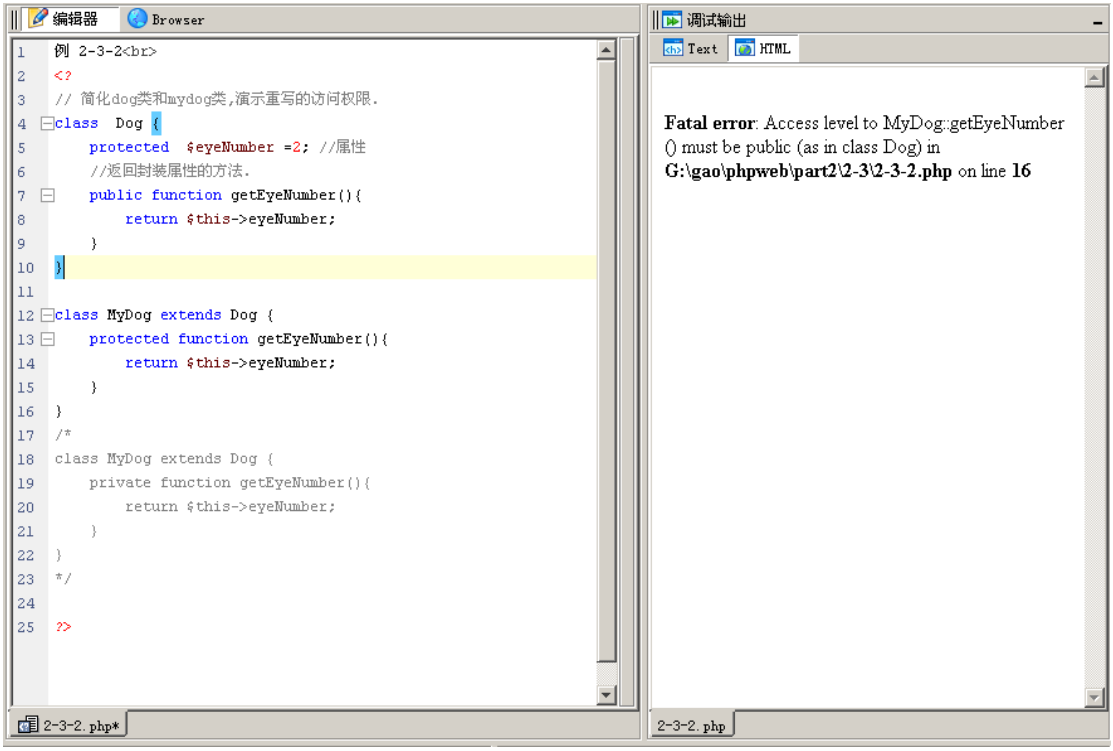
子类中的覆盖方法不能使用比父类中被覆盖方法更严格的访问权限。

例 2-3-2

父类为 public 子类为 private 时。



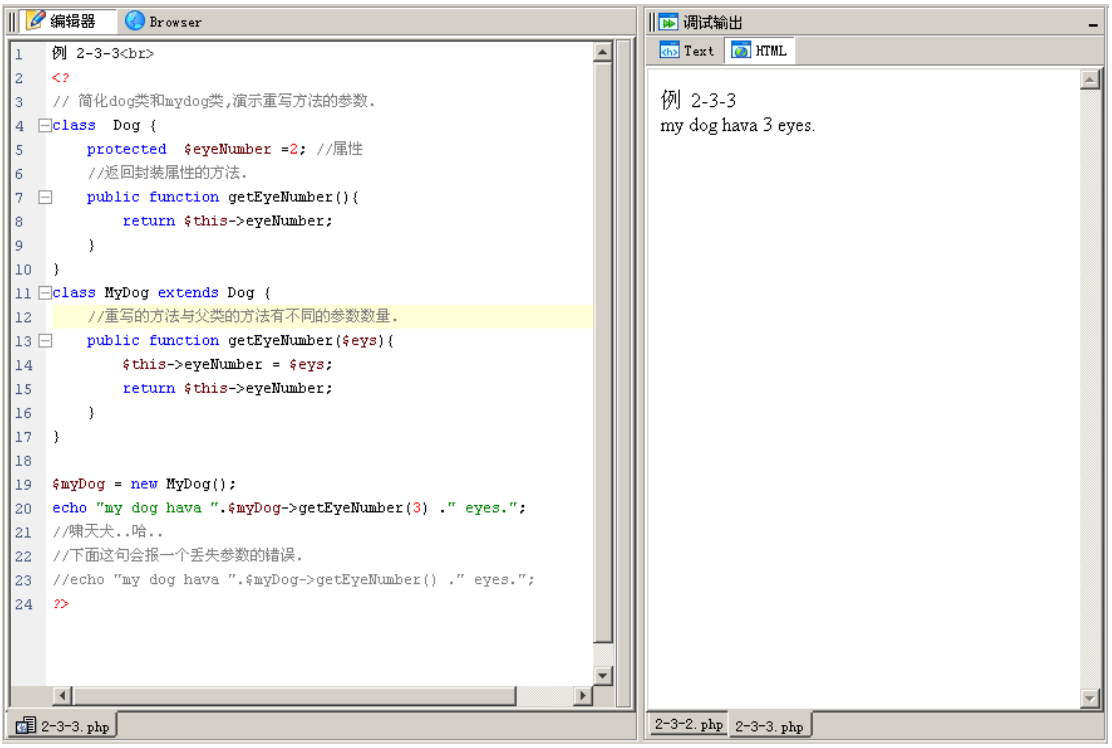
父类为 public 子类为 protected 时。



重写时的参数数量

例 2-3-3

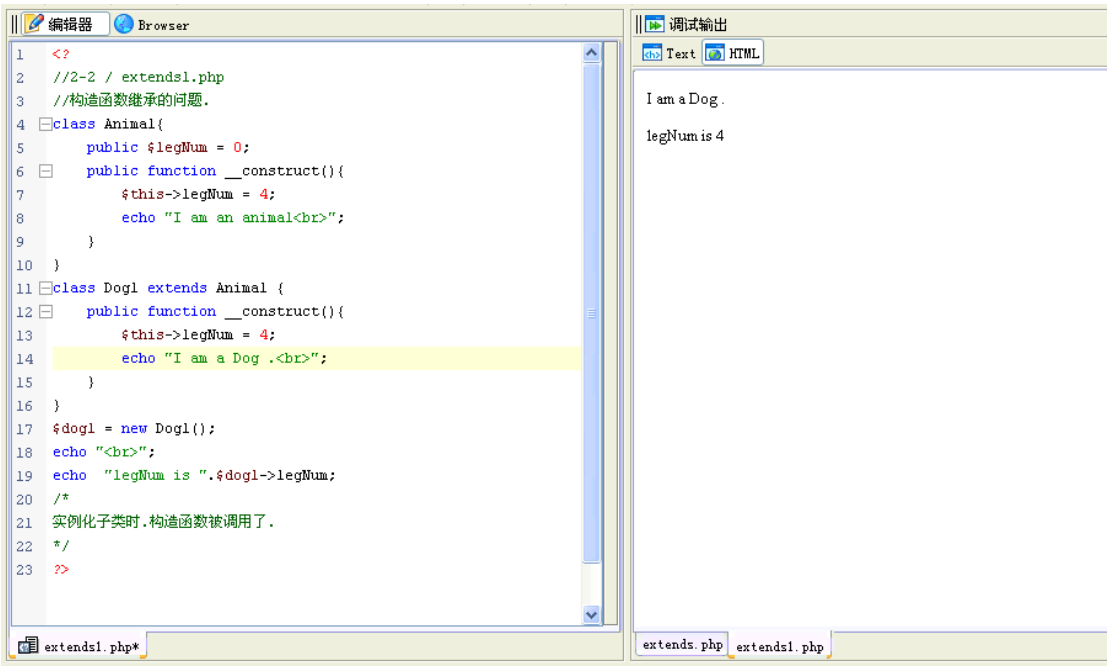
子类可以拥有与父类不同的参数数量。(这点与 java 不同，PHP 是弱类型语言。)



构造函数重写

下面这个例子中，父类和子类都有自己的构造函数，当子类被实例化时，子类的构造函数被调用，而父类的构造函数没有被调用，请对比第一节的构造函数继承。

例子：2-2/ extends1.php



（注：这点和 Java 不同，在 java 中构造函数是不能被继承的，而且子类实例化时，子类的构造函数被调用，父类的构造函数也会调用。）

2.4 this 关键字

- PHP5 中为解决变量的命名冲突和不确定性问题，引入关键字“$this”代表其所在当前对象。
- $this 在构造函数中指该构造函数所创建的新对象
- 在类中使用当前对象的属性和方法，必须使用$this->取值。
- 方法内的局部变量，不属于对象，不使用$this 关键字取值。

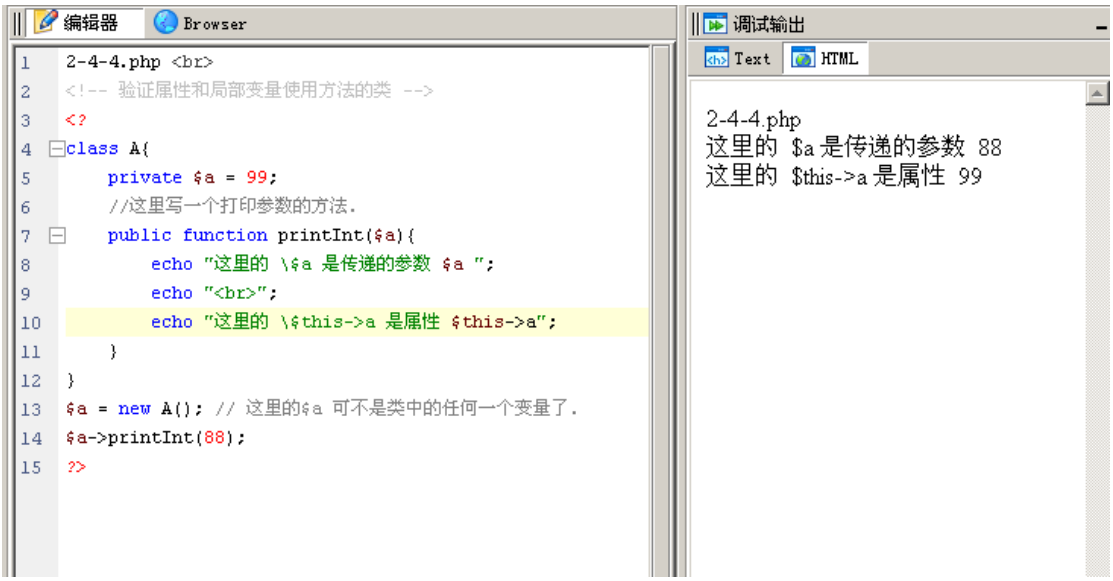
局部变量和全局变量与 $this 关键字

例 2-4-4.php

使用当前对象的属性必须使用$this 关键字。

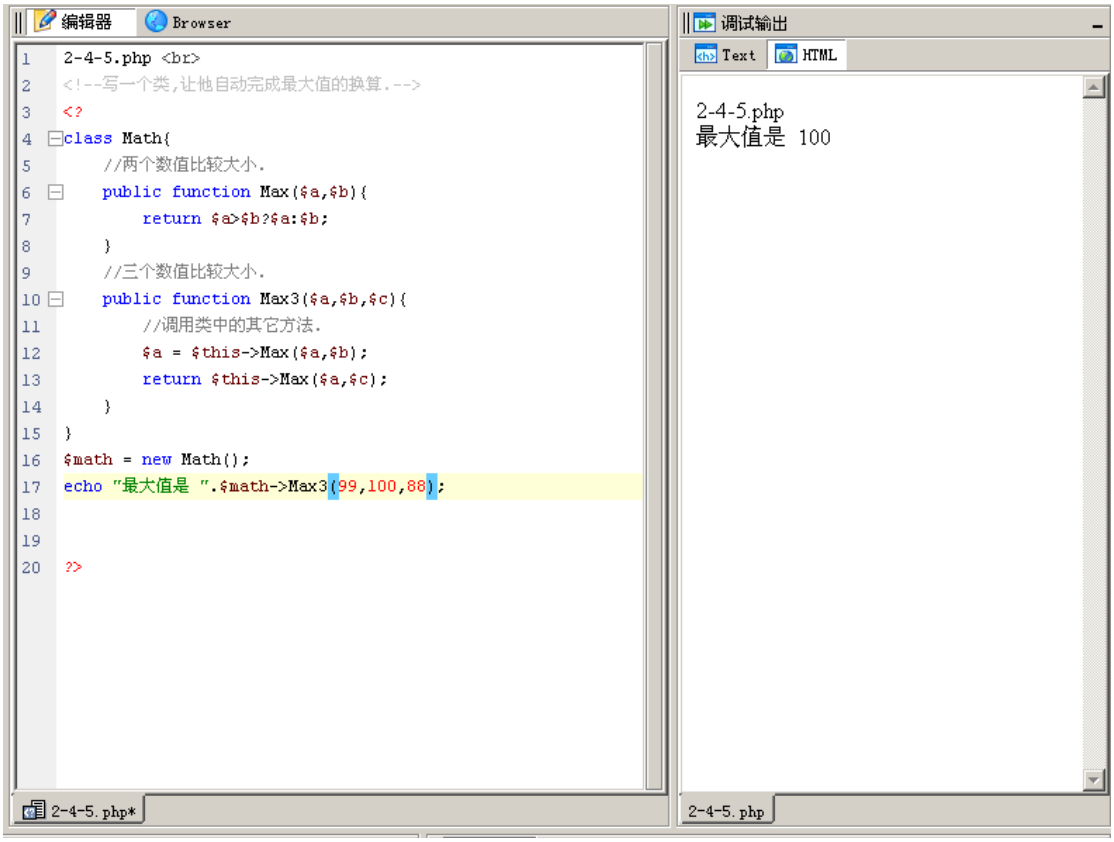
局部变量的只在当前对象的方法内有效，所以直接使用。

注意：局部变量和属性可以同名，但用法不一样。在使用中，要尽量避免这样使用，以免混淆。



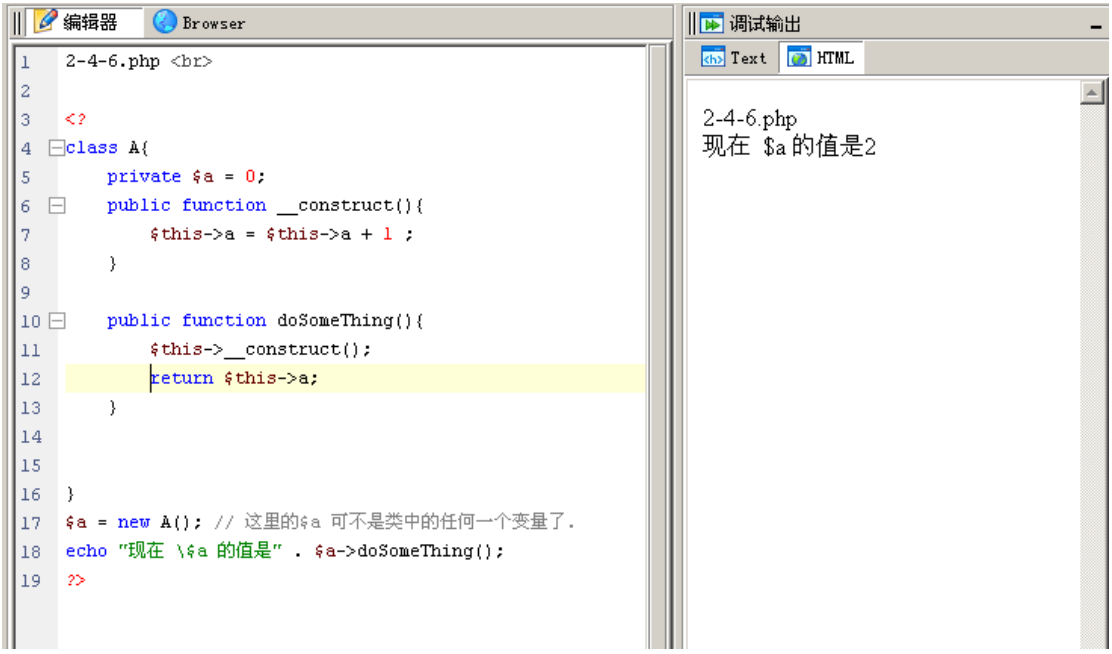
用$this 调用对象中的其它方法

例：2-4-5



使用$this 调用构造函数

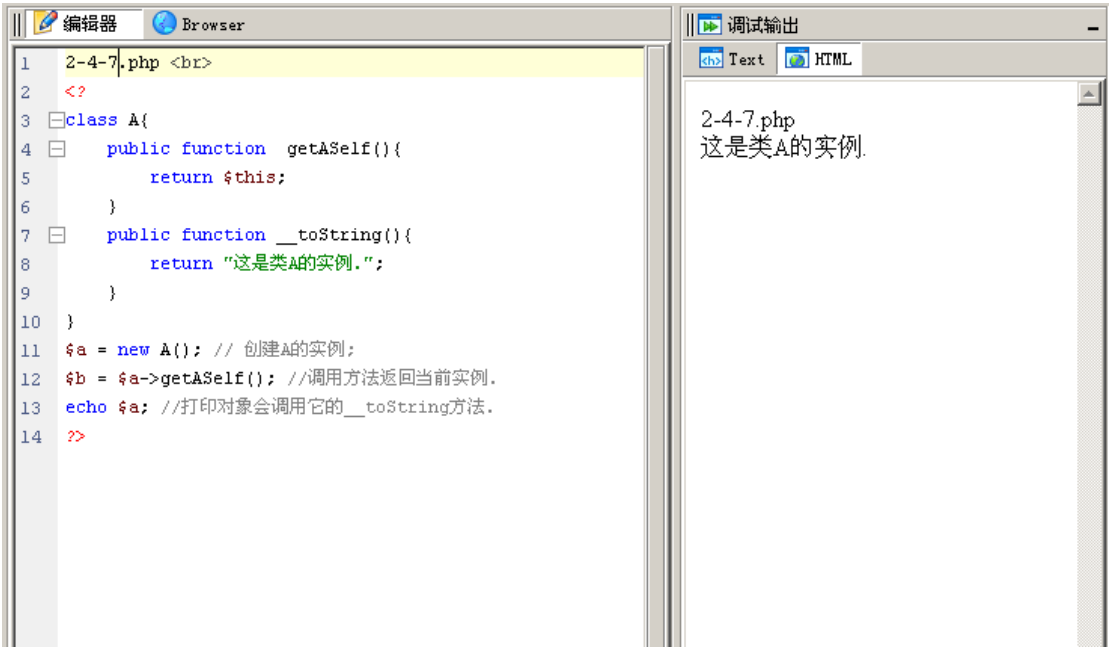
例：2-4-6　调用构造函数和析构函数的方法一致。



$this 到底指的什么？

$this 就是指当前对象，我们甚至可以返回这个对象使用 $this

例 2-4-7



通过 $this 传递对象

编辑器 Browser

```
2-4-8.php <br>
<!-- 通过$this 传递对象
在这个例子中,我们写一个根据不同的年龄发不同工资的类.
我们设置处理年龄和工资的业务模型为一个独立的类.
-->
<?
class User{
    private $age ;
    private $sal ;
    private $payoff ; //声明全局属性.

    //构造函数,中创建Payoff的对象.
    public function __construct(){
        $this->payoff = new Payoff();
    }
    public function getAge(){
        return $this->age;
    }
    public function setAge($age){
        $this->age = $age;
    }
    // 获得工资.
    public  function getSal(){
        $this->sal =  $this->payoff->figure($this);
        return $this->sal;
    }
//这是对应工资与年龄关系的类.
class Payoff{
    public function figure($a){
        $sal =0;
        $age = $a->getAge();
        if($age >80 || $age <16 ){
            $sal = 0;
        }elseif ($age > 50){
            $sal = 1000;
        }else{
            $sal = 800;
        }
        return $sal;
    }
}
//实例化User
$user = new User();

$user->setAge(55);
echo $user->getAge()."age ,his sal is " . $user->getSal();
echo "<br>";

$user->setAge(20);
echo $user->getAge()."age , his sal is " . $user->getSal();
echo "<br>";

$user->setAge(-20);
echo $user->getAge()."age , his sal is " . $user->getSal();
echo "<br>";

$user->setAge(150);
echo $user->getAge()."age , his sal is " . $user->getSal();


?>
```

调试输出

Text HTML

```
2-4-8.php
55age ,his sal is 1000
20age , his sal is 800
-20age , his sal is 0
150age , his sal is 0
```

2.5 parent::关键字

•PHP5 中使用 parent::来引用父类的方法。

parent:: 可用于调用父类中定义的成员方法。

parent::的追溯不仅于直接父类。

通过 parent::调用父类方法

例 2-5-1

编辑器 Browser

```
1  2-5-1.php<br>
2  <!-- 声明一个员工类,经理类继承自员工类 -->
3  <?
4  class employee{
5      protected  $sal=3000;
6      public function getSal(){
7          $this->sal = $this->sal + 1200;
8          return $this->sal ;
9      }
10 }
11
12 class Manager extends employee {
13     //如果想让经理在员工工资的基础上多发1500元.
14     //必须先调用父类的getSal()方法.
15     public function getSal(){
16         parent::getSal();// 这里调用了父类的方法.
17         $this->sal = $this->sal + 1500;
18         return $this->sal ;
19     }
20 }
21 $emp = new employee();
22 echo "普通员工的工资是 " . $emp->getSal();
23 echo "<br>";
24
25 $manager = new Manager();
26 echo "经理的工资是: " . $manager->getSal();
27 ?>
```

2-5-1.php

调试输出

Text HTML

```
2-5-1.php
普通员工的工资是 4200
经理的工资是: 5700
```

2-5-1.php

父类的 private 属性

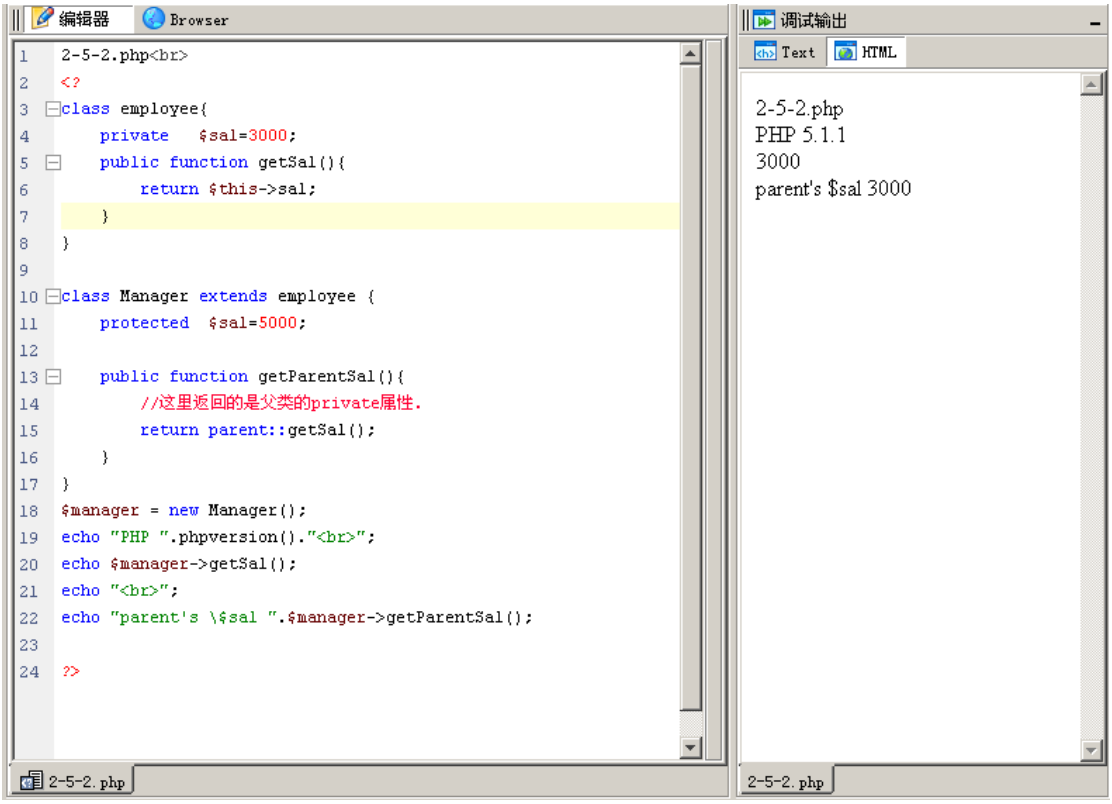
这个东西解释起来十分的不爽。

Private 属性是不能被继承的，如果父类有私有的属性。那么父类的方法只为父类的私有属性服务。

例：2-5-2

下面的例子看起来很奇怪，在子类中重新定义了一个属性$sal

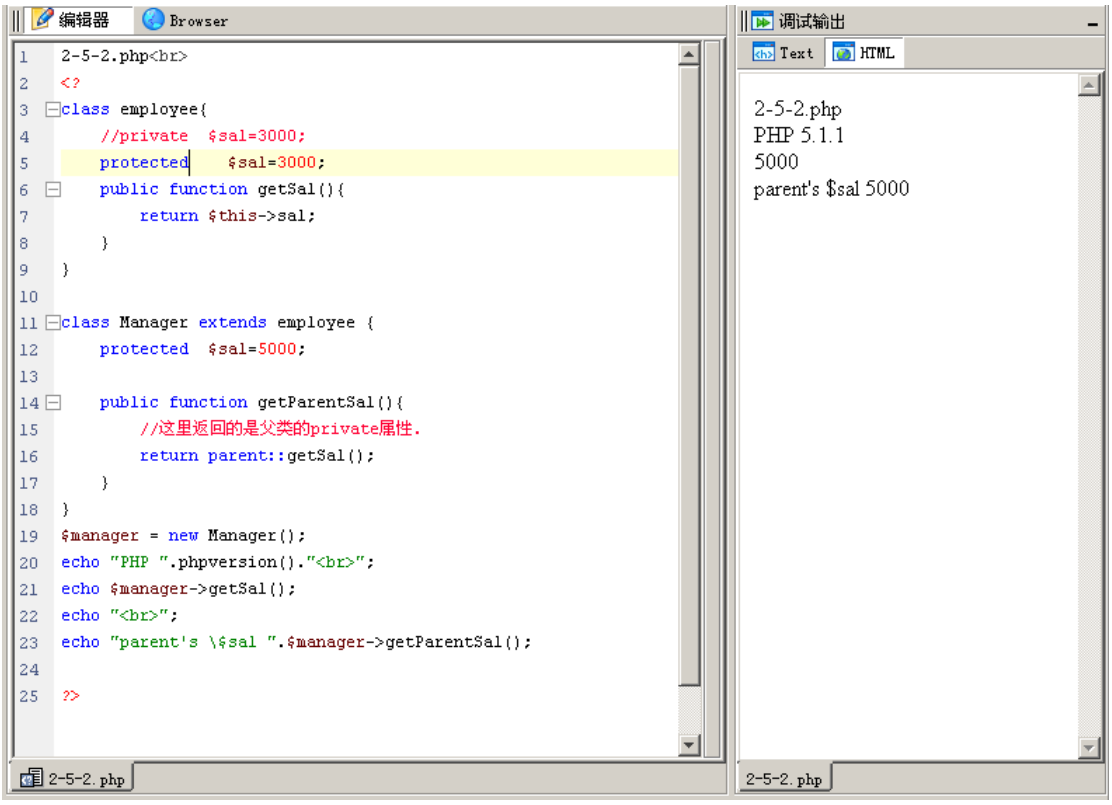
系统却返回了父类的属性。




```
1  2-5-2.php<br>
2  <?
3  class employee{
4      private   $sal=3000;
5      public function getSal(){
6          return $this->sal;
7      }
8  }
9
10 class Manager extends employee {
11     protected  $sal=5000;
12
13     public function getParentSal(){
14         //这里返回的是父类的private属性.
15         return parent::getSal();
16     }
17 }
18 $manager = new Manager();
19 echo "PHP ".phpversion()."<br>";
20 echo $manager->getSal();
21 echo "<br>";
22 echo "parent's \$sal ".$manager->getParentSal();
23
24 ?>
```

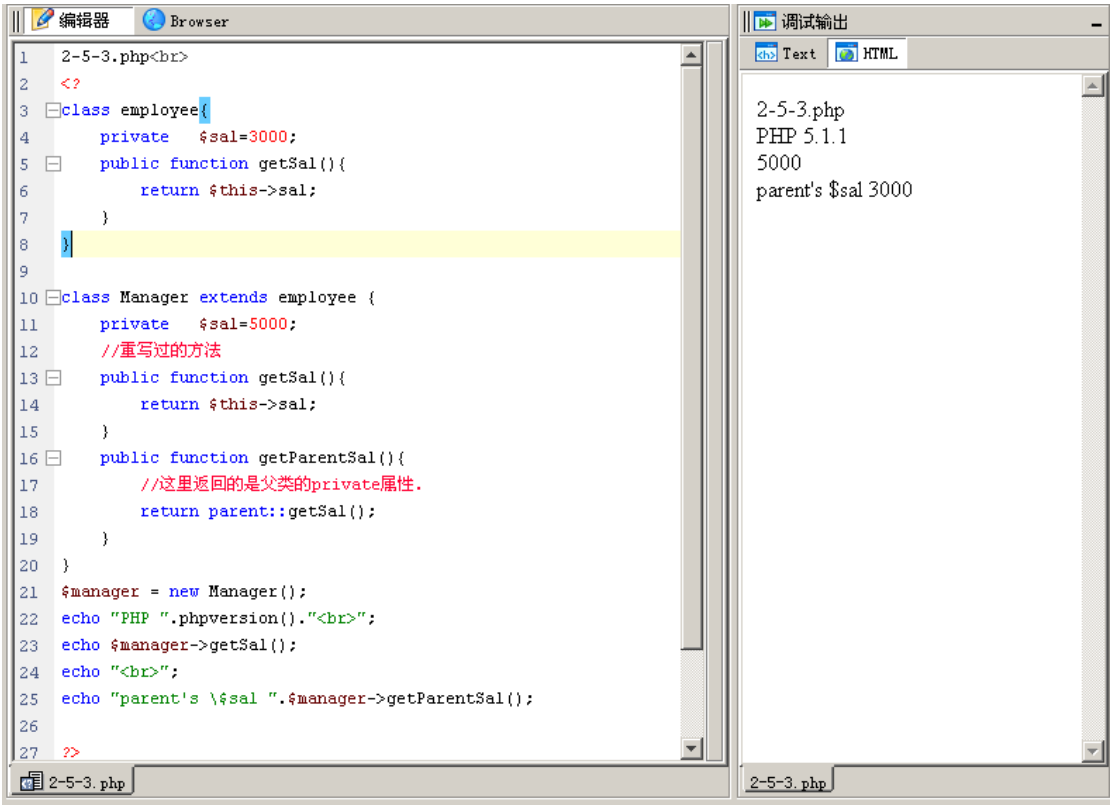
如果父类中的属性被子类重写了。结果是这样的。

注意 第 5 行的属性定义变成了 protected。



子类中重写的方法对当前 private 有效。

例：2-5-3.php



打开 zend 调试状态看看，内存中的情况。注意最下面，有两个$sal 。分别是 3000 和 5000 。
例：2-5-4.php

将父类的属性$sal 改成 protected ,子类重写了父类的属性。
在内存中只有一个 $sal .
例：2-5-5.php

如果你学过 java，你会觉得这一切都是很难理解的。
在 Java 中当子类被创建时，父类的属性和方法在内存中都被创建，甚至构造函数也要被调用。
PHP5 不是这样，PHP5 调用父类用的是 parent:: 而不是 parent-> ，这足以说明 PHP5 不想在内存中让父类也被创建。PHP5 想让继承变的比 Java 更简单。
适应下就好。

这样调用会让 PHP5.1.1 溢出。新版不知道有没有问题。
例：2-5-10.php

第 13 行改成这样就好了。注意比较。

```
return parent:: getSal();
```

这样的代码引起了递归操作，子类调用父类的方法，父类又调用子类方法。

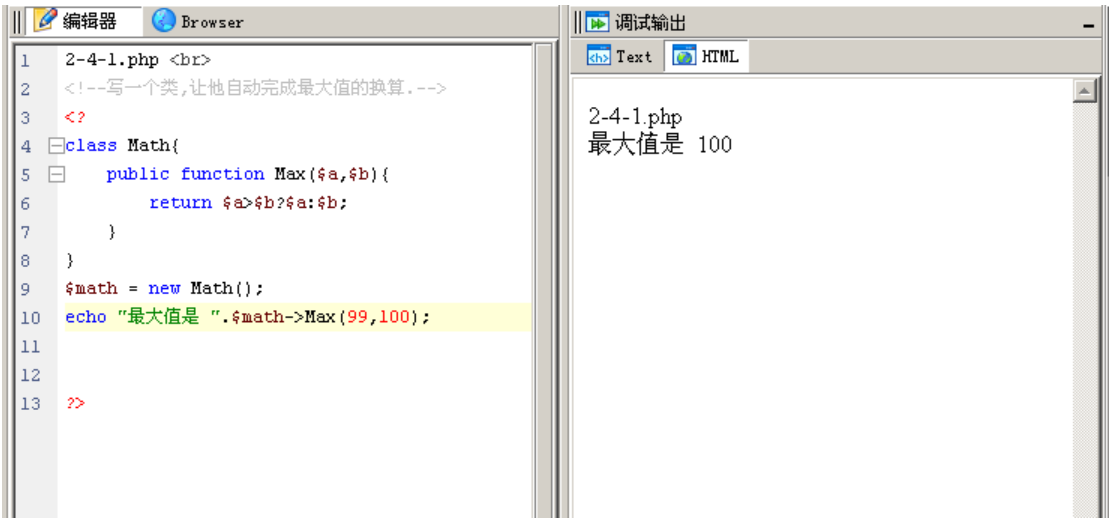
```
return parent::$this->getSal();
```

2.5 重载 Overload

- 当类中的方法名相同时，称为方法的重载(overload)
- 重载是 Java 等面向对象语言中重要的一部分。
- 在 PHP5 中不支持重载。

在 PHP5 中不支持重载。

例 2-4-1
先写一个取最大值的类。

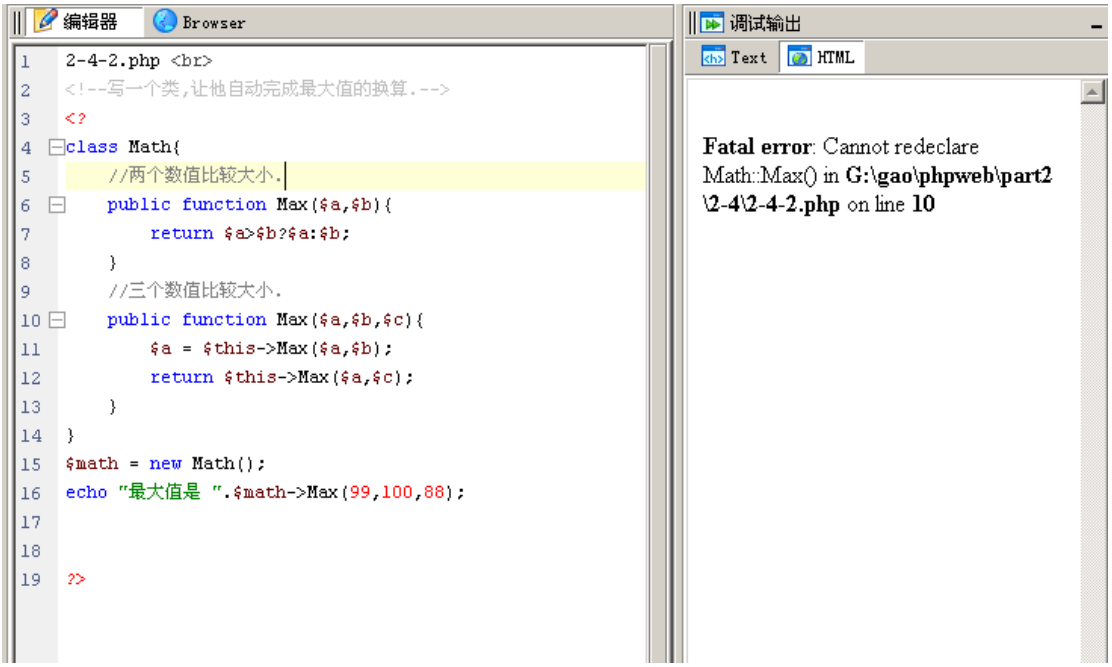


例：2-4-2

如果传递过来三个数值。如何计算？

下面的写法，在 Java 中是太平常不过了。但是在 PHP5 中，不能这样。

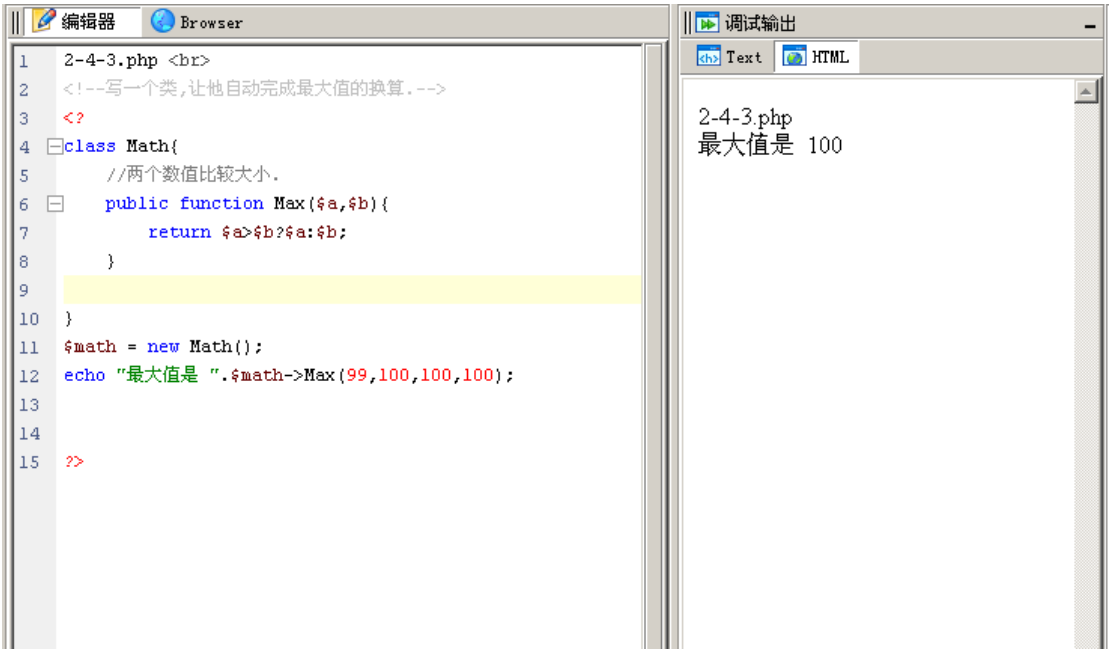
PHP5 不支持有多个相同名称的方法，也就是不支持重载。



例 2-4-3

对于一个方法，缺少参数时候会报错。

当参数太多的时候，PHP 就当什么都没有看到。程序可以正常运行。



2.7 实例

简单写一个实例，这个例子只是本章内容的表达。在了解更多面向对象内容后，相信你能写出更合适的表达代码。

在上一章，我们写了一个建立用户 user 类，直接使用 user 类读取用户信息的类。假设我们又有了新的需求。

- 任何用户都可以查看别的用户的信息，当然不能看到别人的密码。
- 任何用户都可以修改自己的密码。

于是我们对第一章的类做些改动，首先我们在 userInfo 类中，将获得密码的方法隐藏。

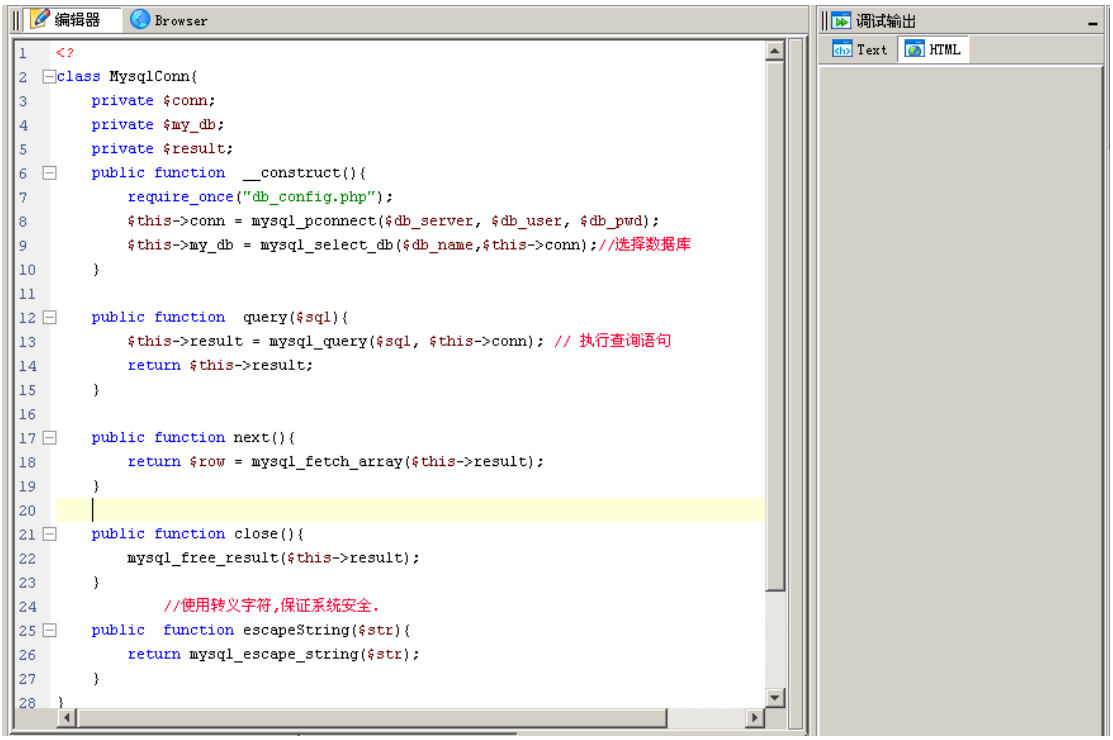
我们再写一个 UserChange 类继承自 userInfo，在 UserChange 中增加修改密码的方法。并将获取密码的方法重写为 public 权限。

这样，在你的页面中，就可以创建两种 user。一种是只能看到信息不能看到密码、不能修改密码的 userInfo 的实例。另外一种是比 userInfo 功能更强的 UserChange 类，这个实例可以修改密码，可以获得密码。

在合适的位置创建不同的 user，就是你的业务逻辑的内容了。

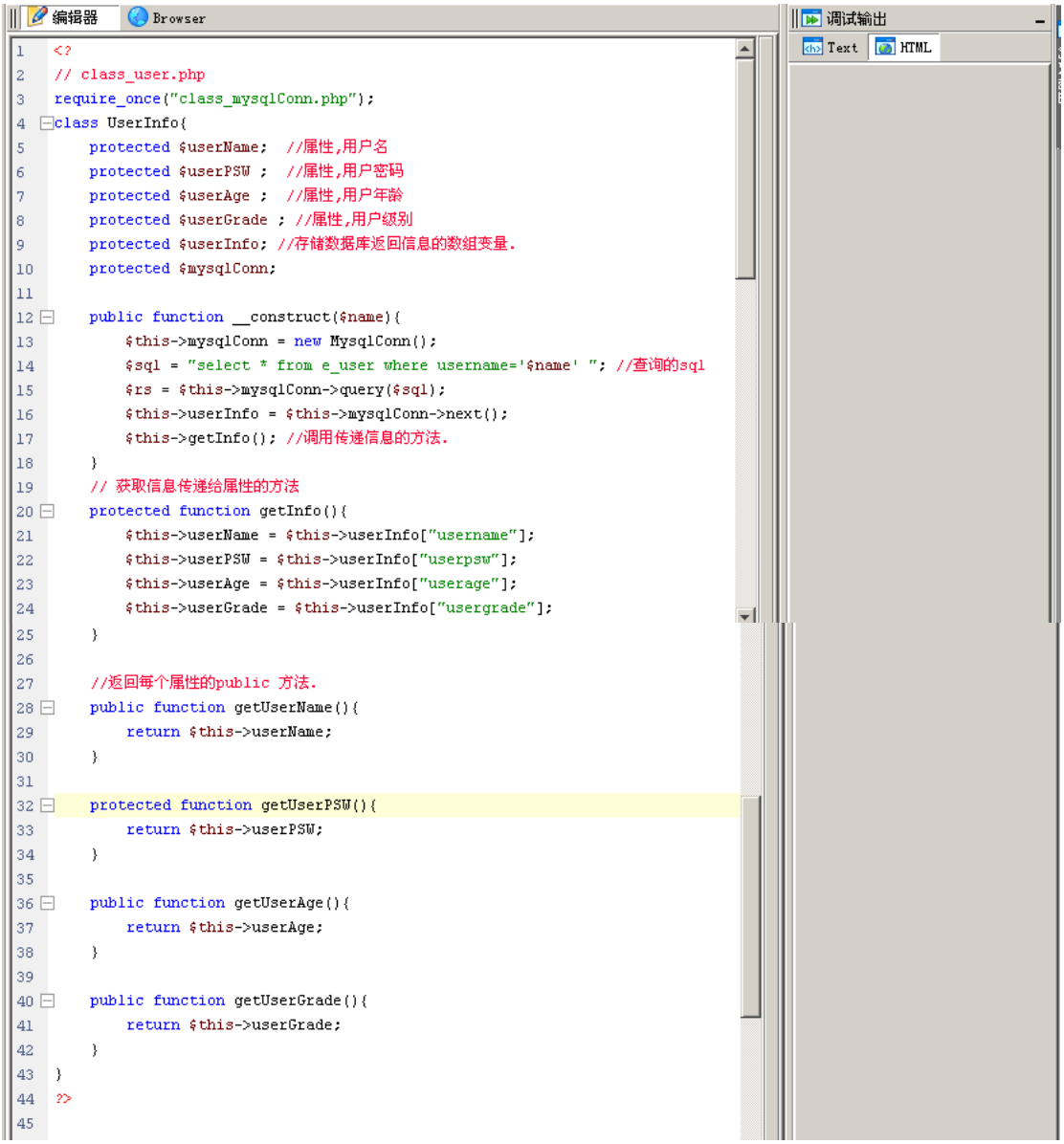
同时，我们独立出一个数据库连接类，数据库连接类比较完善的网上有很多，大家学习完毕面向对象后，自己也可以写出更完善的数据库类。

Mysql 连接类。




```php
<?
class MysqlConn{
    private $conn;
    private $my_db;
    private $result;
    public function  __construct(){
        require_once("db_config.php");
        $this->conn = mysql_pconnect($db_server, $db_user, $db_pwd);
        $this->my_db = mysql_select_db($db_name,$this->conn);//选择数据库
    }

    public function  query($sql){
        $this->result = mysql_query($sql, $this->conn); // 执行查询语句
        return $this->result;
    }

    public function next(){
        return $row = mysql_fetch_array($this->result);
    }

    public function close(){
        mysql_free_result($this->result);
    }
        //使用转义字符,保证系统安全.
    public  function escapeString($str){
        return mysql_escape_string($str);
    }
}
```

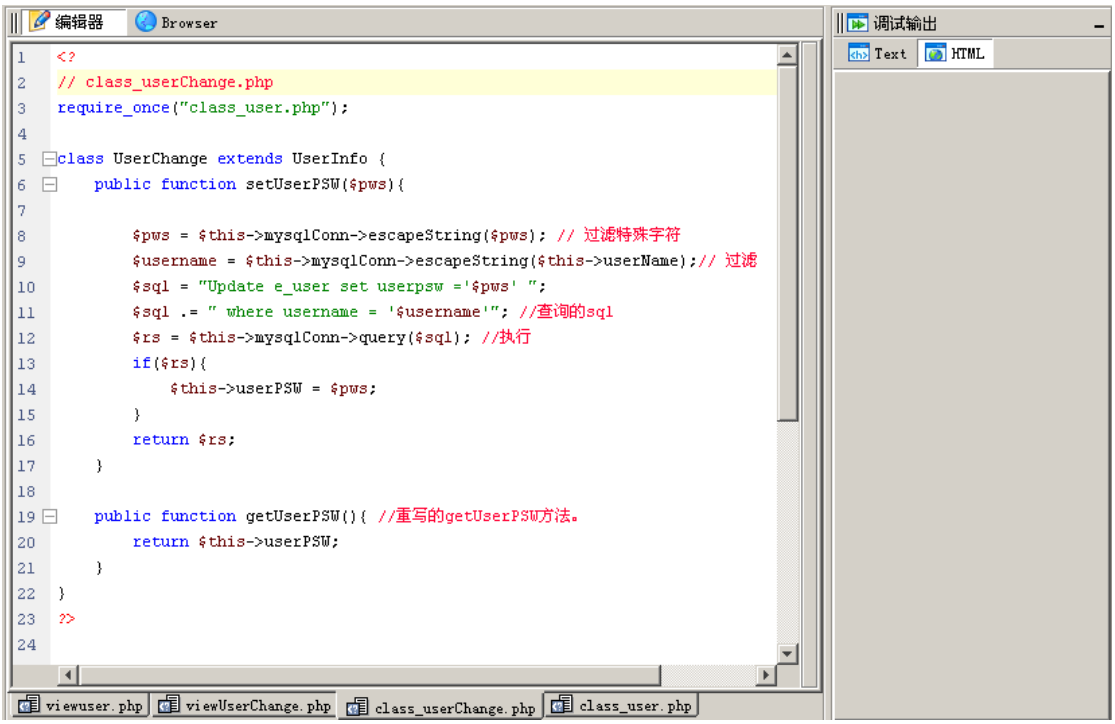
父类 User 类




```php
<?
// class_user.php
require_once("class_mysqlConn.php");
class UserInfo{
    protected $userName;  //属性,用户名
    protected $userPSW ;  //属性,用户密码
    protected $userAge ;  //属性,用户年龄
    protected $userGrade ; //属性,用户级别
    protected $userInfo; //存储数据库返回信息的数组变量.
    protected $mysqlConn;

    public function __construct($name){
        $this->mysqlConn = new MysqlConn();
        $sql = "select * from e_user where username='$name' "; //查询的sql
        $rs = $this->mysqlConn->query($sql);
        $this->userInfo = $this->mysqlConn->next();
        $this->getInfo(); //调用传递信息的方法.
    }
    // 获取信息传递给属性的方法
    protected function getInfo(){
        $this->userName = $this->userInfo["username"];
        $this->userPSW = $this->userInfo["userpsw"];
        $this->userAge = $this->userInfo["userage"];
        $this->userGrade = $this->userInfo["usergrade"];
    }

    //返回每个属性的public 方法.
    public function getUserName(){
        return $this->userName;
    }

    protected function getUserPSW(){
        return $this->userPSW;
    }

    public function getUserAge(){
        return $this->userAge;
    }

    public function getUserGrade(){
        return $this->userGrade;
    }
}
?>

```

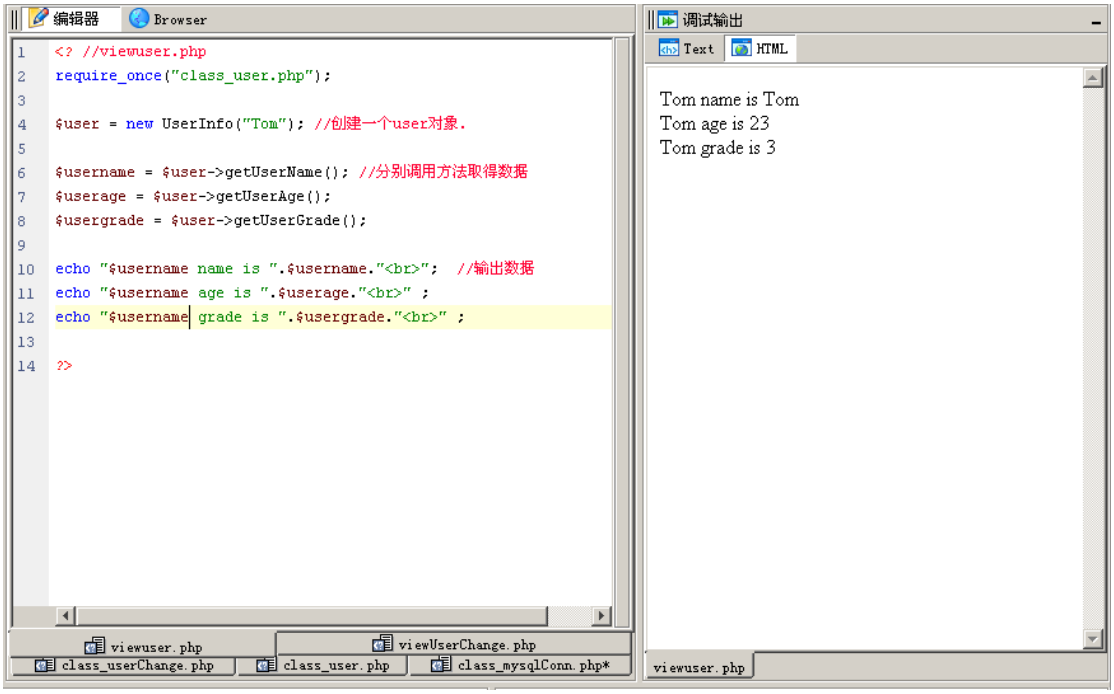
子类 UserInfo 类

添加了修改密码的方法，重写并公开了获取密码的方法。

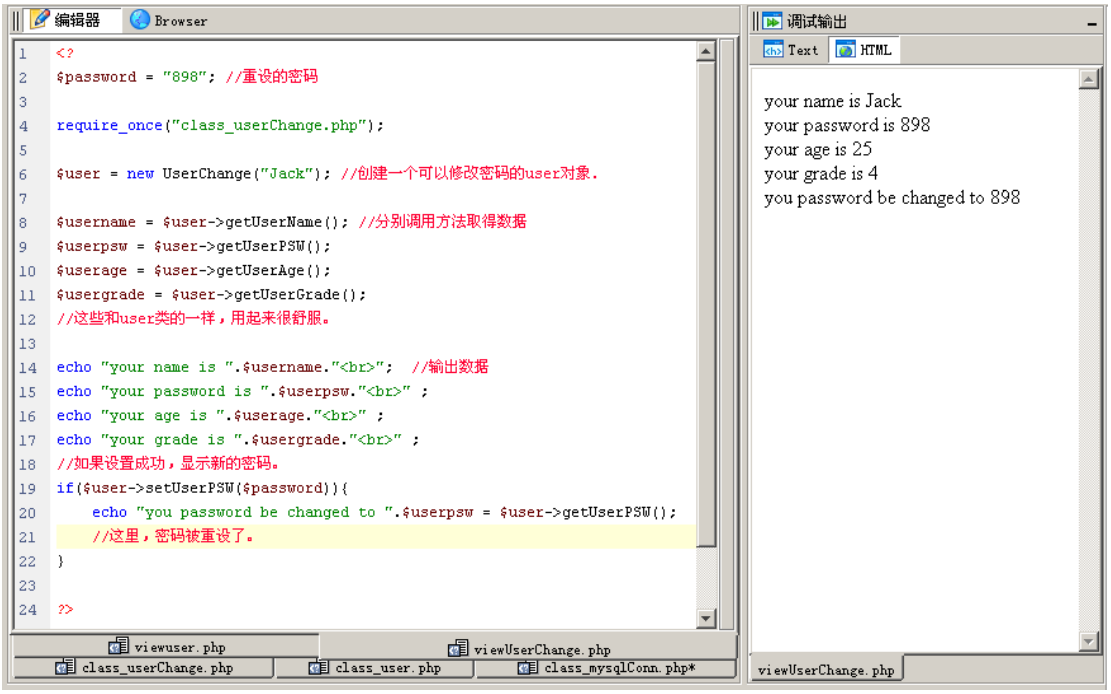



```php
<?
// class_userChange.php
require_once("class_user.php");

class UserChange extends UserInfo {
    public function setUserPSW($pws){

        $pws = $this->mysqlConn->escapeString($pws); // 过滤特殊字符
        $username = $this->mysqlConn->escapeString($this->userName);// 过滤
        $sql = "Update e_user set userpsw ='$pws' ";
        $sql .= " where username = '$username'"; //查询的sql
        $rs = $this->mysqlConn->query($sql); //执行
        if($rs){
            $this->userPSW = $pws;
        }
        return $rs;
    }

    public function getUserPSW(){ //重写的getUserPSW方法。
        return $this->userPSW;
    }
}
?>
```

在任何位置都可以放心使用 userInfo 类。



可以重设密码的 Userchange 类的实例。



本章介绍了，PHP5 面向对象关于继承、重载、重写、$this 关键字，通过 parent 调用父类方法等 PHP5 面向对象初步的知识。

下一章，面向对象设计进阶。希望大家支持。