# 理解Apex测试

Apex测试应用程序的方式有两种。一种是通过Salesforce用户接口，这种方式虽然很重要，但仅仅通过用户接口不能捕获应用程序的所有用例。另一种方式是测试批量功能：如果使用SOAP API或Visualforce标准集控制器调用，最多可以传递200条记录。

An application is seldom finished. You will have additional releases of it, where you change and extend functionality. If you have written comprehensive tests, you can ensure that a regression is not introduced with any new functionality.

部署或者打包代码到Force.com AppExchange之前，以下条件必须满足：

1.至少有75%的Apex代码被单元测试覆盖，所有的这些单元测试必须完全成功。

注意：

* + - 当部署Apex代码到生产组织时，组织名称空间中每个单元测试都会默认执行。
    - 调用System.debug不算作Apex代码覆盖率的一部分。
    - 测试类和测试方法不算作Apex代码覆盖率的一部分。
    - 当75%的Apex代码被覆盖时，应该注意的焦点就不应该是覆盖率了。这时应该确认每一个应用程序中的用例都被覆盖了，包括正面和反面用例，多个和单个记录。这样做可以使75%或以上的代码被单元测试覆盖。

2.每个触发器必须有一些测试覆盖。

3.所有的类和触发器必须编译成功。

Salesforce在具有Apex代码的所有组织中运行所有测试，以验证由于任何服务升级而没有任何行为被更改

# 测试内容

Salesforce建议为以下内容编写测试：

单一操作

测试以验证单个记录是否产生正确的，预期的结果。

批量操作

任何Apex代码，不论触发器、类或者是扩展代码，可以调用1到200条记录。最好不要只测试单独一条记录，而是批量测试。

Positive behavior

测试以验证预期的行为是通过每个预期排列发生的，也就是说，用户正确填写一切并且没有超出极限。

Negative behavior

应用程序可能有一些限制，例如无法添加未来日期，无法指定负数等。必须测试负面情况，并验证错误消息是否正确生成以及在正常情况下为正数。

受限制用户

测试在代码中使用的对sObjects的访问受限的用户是否看到预期的行为。也就是说，他们是否可以运行代码或接收错误消息。

注意：条件运算和三元运算只有当Positive分支和Negative分支都被执行时才会算作执行。

# Apex单元测试

为了促进开发健壮、没有错误的代码，Apex支持创建和执行单元测试。单元测试是用来验证某个特定代码段是否能够正确运行的类。单元测试方法不带参数、不向数据库提交任何数据、不发送邮件，并在定义时被测试关键字testMethod标记或者用@isTest注解。而且，测试方法必须被定义在测试类中，也就是被@isTest注解的类。

# Test Class for a Apex Schedule Class

public static @isTest void methodName(){

Test.startTest();

ScheduleClassName scn = new ScheduleClassName();

String str = ‘0 0 23 \* \* ?’;

System.schedule(‘Test’, scn, str);

Test.stopTest();

}

# Visualforce开发工具

开启Visualforce开发模式：

1.在设置里的设置搜索框输入**高级用户详细信息（Advanced User Detials）**，然后选择高级用户详细信息**。**没有结果？在搜索框输入**个人信息（Personal Information）**，选择个人信息。

2.点击编辑。

3.选中开发模式复选框。

4.选中在开发模式中显示查看状态复选框（非必选），会在开发页脚上显示查看状态标签。此标签可以用来监控Visualforce页面的执行。

5.点击保存。

Salesforce user interface

[The Force.com IDE](https://developer.salesforce.com/page/Force.com_IDE)

# 使用开发模式页脚

开启开发模式，在地址栏输入URL后就可以查看和编辑页面。例如，Page名为HelloWord，Salesforce instance为na3.salesforce.com ，就在地址栏输入

<https://na3.salesforce.com/apex/HelloWorld>。

View State

Component Reference

Disable Development Mode

注意：使用View State标签的开发者必须理解页面请求的步骤，使用之前需要先学习order of execution in a Visualforce page。

因为View State是和form数据相关联的，所以只有当页面中包含<apex:form>标签时View State才会出现。而且，View State只显示在使用Custom Controllers 或者Controller Extensions的页面中。

View State由文件夹节点组成。点击其中任何一个文件夹，就会出现一个带内容标签的饼状图。这个图展示了文件夹的子类VF自定义Controller，Apex对象，或者字段。可以通过鼠标悬停在某个块上来查看元素对父类大小的影响。这是和文本节点一样的信息。这个功能要求浏览器激活了version6或者更高版本的Flash。

Salesforce允许VF page的View State的大小最大是135KB。View State标签展示了页面中的那一个元素占用了这些内存。通常，view state大小越小，加载速度越快。如果要缩小页面的view state的大小，可以通过优化ApexController的代码和移除多余的在使用的VF组件。例如：

1.如果view state中有比较大比例的部分来源于Controllers或者Controller Extensions,可以考虑优化SOQL，只返回和VF page有关联的数据。

2.如果view state被component tree影响，就需要尽量减少 页面所依赖的组件的数量。

# VF控件库

Standard Component Reference

**apex:actionFunction**

一个为使用ajax请求直接从JavaScript代码调用controller中的方法提供支持的组件。该组件必须包含在**apex:form**组件之内。

与**apex:actionSupport**只支持从其它VF组件调用controller中的方法不同的是，该组件定义了一个新的可以被javascript代码段调用的javascript功能。

**apex:actionPoller**

一个根据指定的interval（时间间隔）定时向服务器发送ajax请求的定时器。每次请求都会部分或者整体刷新页面。

该组件必须位于它作用的区域内。

使用该组件的注意事项：

被该组件调用的方法应该是尽量简单的。最好避免在该组件调用的方法内使用DML、外部服务调用和其它资源密集型操作。认真考虑根据指定的时间间隔重复执行所调用的方法所带来的影响，特别是用在分布比较广或者开放周期比较长的页面时。

该组件会定时刷新connection，保持session的有效性。使用该组件的页面不会因为长时间不操作而超时。

不要在enhanced lists使用该组件。

# Sandbox

# with sharing Keyword or without sharing Keyword

apex类的这两个关键字用来指定是否执行sharing rules。

With sharing关键字用来为当前被class考虑在内的用户指定分享规则。必须为该类显式设置此关键字，因为apex代码在系统上下文中运行。在系统上下文中，apex代码有访问所有对象和字段——对象存取权限、字段等级安全、没有应用到当前用户的分享规则。这是为了确保不会因为对于某个用户隐藏的字段或对象导致代码执行失败。这个规则的例外是被executeAnonymous调用的代码和Apex中的Chatter。executeAnonymous总是使用当前用户的完整权限执行。

# ApexPages Namespace

## Action Class

使用ApexPages.Action可以在VS自定义controller和controller扩展中创建action方法。

Action Constructors

Action(action)

使用指定action创建ApexPages.Action实例

Signature

Public Action(String Action)

Action Methods

getExpression()

返回值类型：String

invoke()

返回值类型：System.PageReference

## Component Class

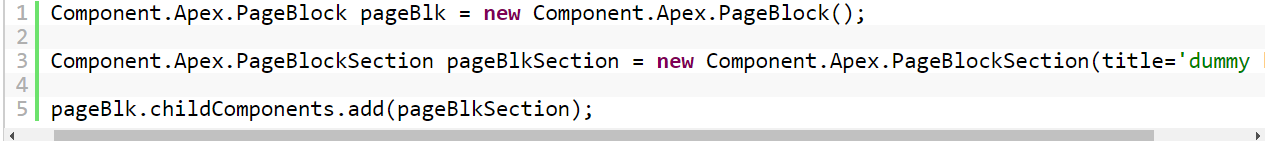
**动态组件属性**

**childrenComponents**

为组件返回一个子组件（可能是多个子组件）的引用

值类型：List<ApexPages.Component>

示例：

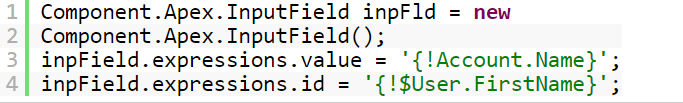


**expressions**

使用expression语言符号为一个属性设置值。

值类型：String

示例：



**facets**

Sets the content of a facet to a dynamic component. (将facet的内容设置为动态组件？)

值类型：String

示例：



## IdeaStandardController Class

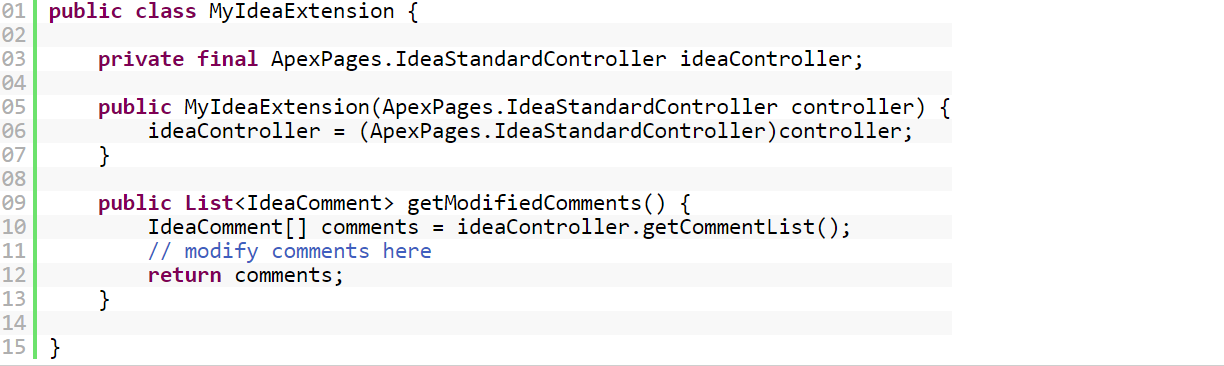
IdeaStandardController 对象提供除了StandardContraller提供的功能之外的特定功能。

IdeaStandardController 对象的方法被IdeaStandardController 的特定实例调用和操作。

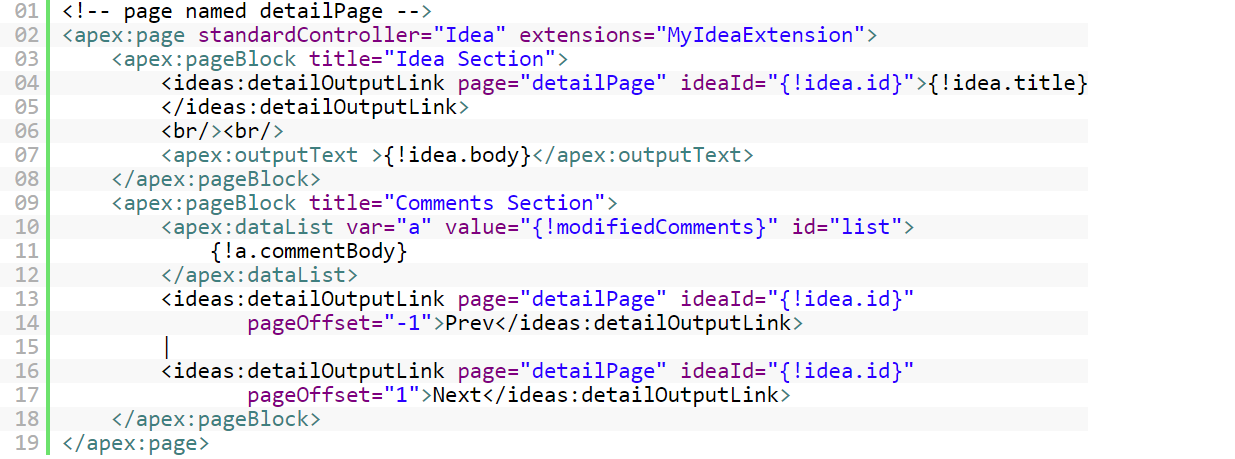
IdeaStandardController 类继承了所有StandardController类的方法。

IdeaStandardController 不能被实例化，使用ideas controller可以通过自定义扩展controller的构造函数获取实例。

示例：



下面的VS代码展示了怎么将上面的例子在一个页面中使用。页面必须命名为detailPage。



## StandardController Class

**Usage**

StandardController的对象引用Salesforce提供的预建VF Controller。唯一一个必须要引用StandardController对象的情况是为一个标准Controller定义extension的时候。StandardController是extension class构造函数的唯一参数的类型。

**Q：What is Controller Extension?**

**A：A controller extension is any Apex class containing a constructor that takes a single argument of type** *ApexPages.StandardController* **or CustomControllerName, where CustomControllerName is the name of a custom controller you want to extend.**

**StandardController Constructor**

参数类型：SObject（一个标准或自定义对象）

**StandardController Methods**

addFields(fieldNames)

参数类型：List<String>

用法：该方法应该在一条记录被加载前调用，它是通过controller的构造函数调用的。如果该方法要在构造函数之外调用，需在调用addField()之前先调用reset()函数。

FieldNames中的String可以是字段的API名称或者字段的明确关系。

此方法只通过动态VF绑定用于controllers。

cancel()

返回类型：System.PageReference

delete()

删除记录并返回到删除页面。

返回值类型：System.PageReference

edit()

返回值类型; System.PageReference

getId()

根据Visualforce页面URL中的id查询字符串参数的值，返回当前处于上下文中的记录的ID。

返回值类型：String

getRecord()

根据Visualforce页面URL中的id查询字符串参数的值，返回当前处于上下文中的记录。

返回值类型：sObject

Note：只有在相关联的VF标签中引用的字段才可用于查询此sObject。其他的字段包括任何其他对象中的关联字段，必须使用SOQL语句查询。

可以通过包含引用您要查询的任何其他字段的隐藏组件来解决此限制。通过将组件的rendered属性设置为false来隐藏组件的显示

示例：



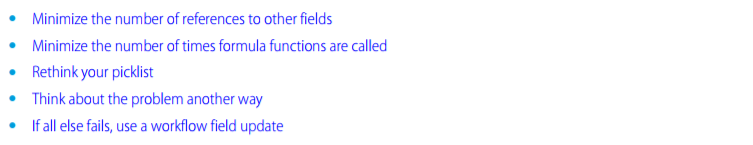
reset()

强制控制器重新获取对新引用字段的访问。在此方法调用之前对记录所做的任何更改都将被丢弃。

返回值类型：Void

**Usage**

# 减少公式编译长度的方法



# ProcessInstance

## ProcessInstance

Represents an instance of a single, end-to-end approval process. Use this and the node, step, and workitem process instance objects to create approval history reports.

## ProcessInstanceHistory

This read-only object shows all steps and pending approval requests associated with an approval process (ProcessInstance).

## ProcessInstanceStep

Represents one work item in an approval process (ProcessInstance).

## ProcessInstanceNode

Represents a step in an instance of an approval process. Compare to ProcessNode, which describes the step in a process definition. Use this object to retrieve approval history.

## ProcessInstanceWorkitem

Represents a user’s pending approval request.