**UiPath手把手教程六-工作流之状态机**

大家好这里是Python知识学堂。

作者在RPA领域工作已经两年多，看到UiPath产品也在不断的完善，有必要分享一下学习和使用UiPath的一些经验，帮助小伙伴踩坑。文章中有什么不对的地方还请指出，一起学习一起进步。

上期文章我们介绍了UiPath工作流中的Flowchart,并使用一个数字猜测的例子来说明了如何使用Flowchart来构建流程，大家可以温习一下。本次文章我们来学习一下UiPath的状态机（State Machine）。

关于状态机的官方文档：

https://docs.uipath.com/studio/docs/state-machines

**一、什么是状态机**

大家一致对于状态机的认识为：在UiPath中状态机是**一种自动化的类型**，它在执行过程中使用有限的状态，可以在活动触发时进入状态，在另一个活动被触发时退出状态。

当然了状态机也是一种工作流类型，当是在一定程度上是独立的。我们在进行流程构建的时候，可以在状态机中使用Sequence和Flowchart。这样一来，状态机就可以实现更加复杂的流程了。

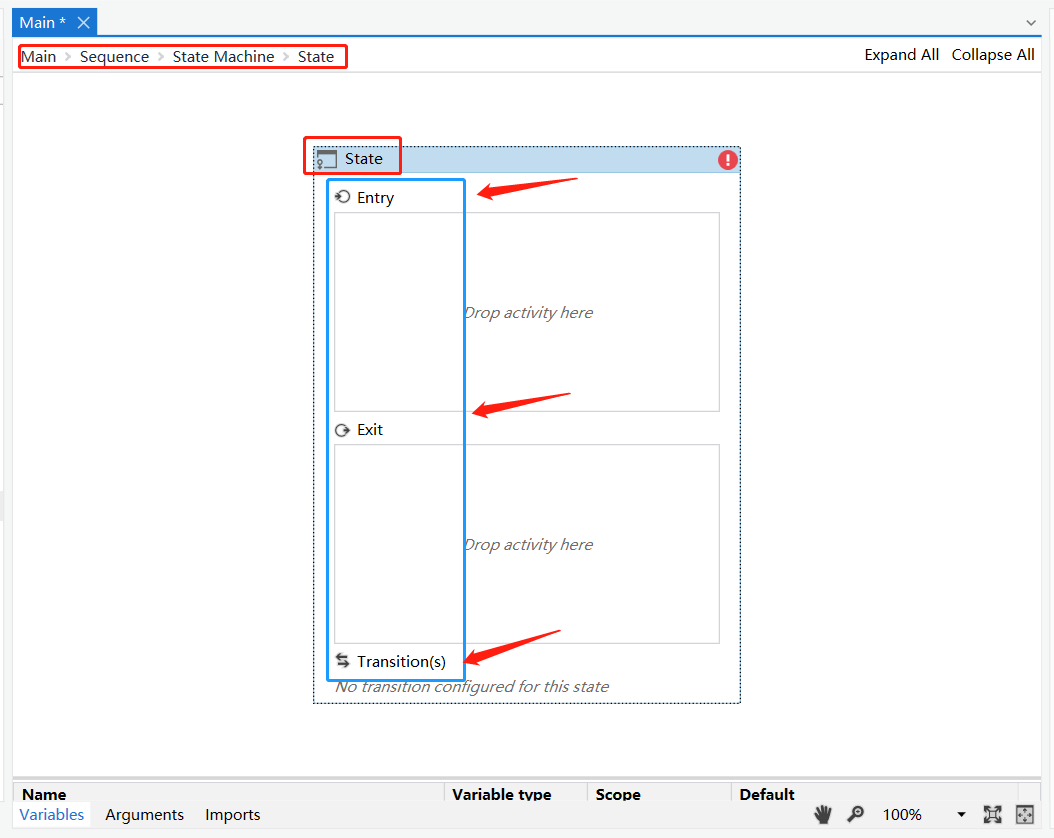
状态机的另一个重要方面是Transition（转换），而且其可以使我们添加从一个状态跳转到另一个状态的条件，这些条件由箭头或状态之间的分支表示。

State、Final State是属于状态机的两个特定的活动，官网文档中说道：我们只能创建一个初始状态，但是有可能有多个Final State（最终状态）。这个很好理解，毕竟初始开始状态是需要唯一的。

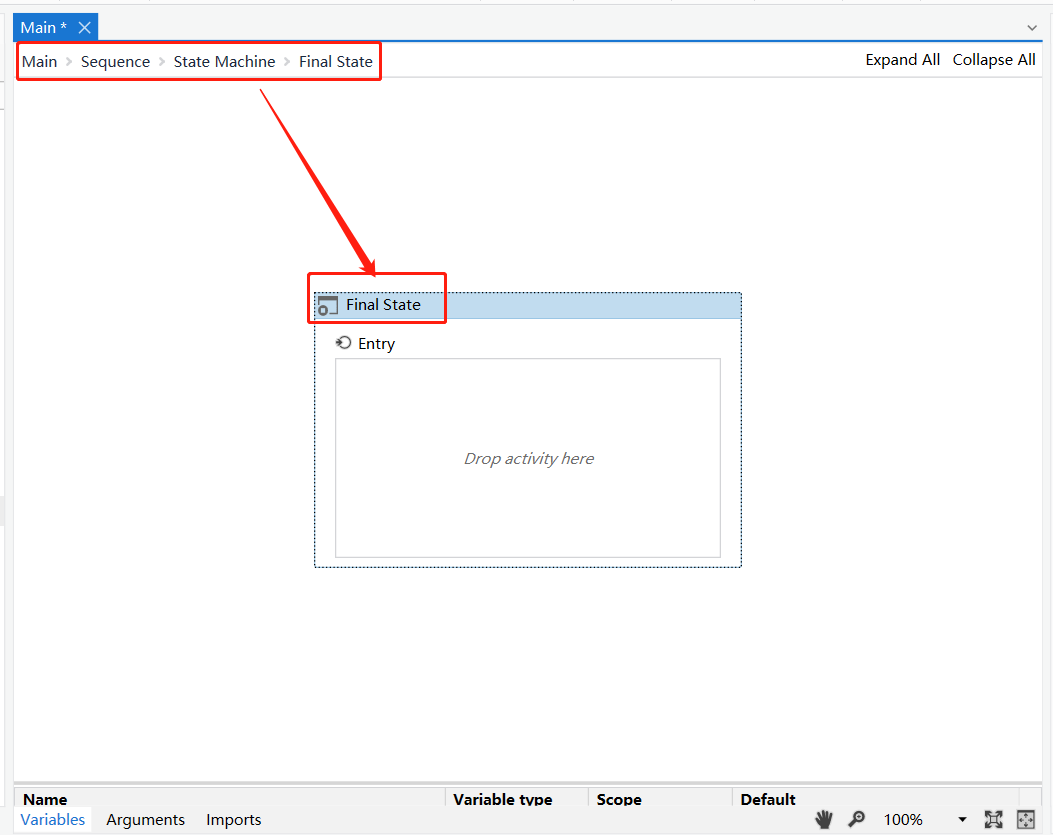
**二、State和Final State**

官网文档写到State 活动包含三个部分，**Entry**, **Exit** 和 **Transition**，而 Final State 只包含一个部分 Entry。拖拉这些活动到设计器中之后，可以通过双击来进行活动的展开，以查看更多的信息并编辑它们。

我们先拖动一个State到设计器中看看他长什么样子：



Final State：



可以看出State活动有三个部分，而Final State活动中只有Entry部分。

在State活动中，**Entry和Exit是用来添加哪些活动的呢？**为所选状态添加进入和退出时要执行的活动。而**Transition**用于显示当前连接到所选状态的所有转换。

在Final State活动中Entry为所选状态添加进入和退出时要执行的活动，详细信息我们将在后续案例中进一步阐述。

State活动包含的3个部分：

**Entry：主要为所进入的状态添加想要执行的活动。**

**Exit：主要为所退出的状态添加想要执行的活动。**

**Transition(s)：则会显示连接到当前所选状态的所有转换。**

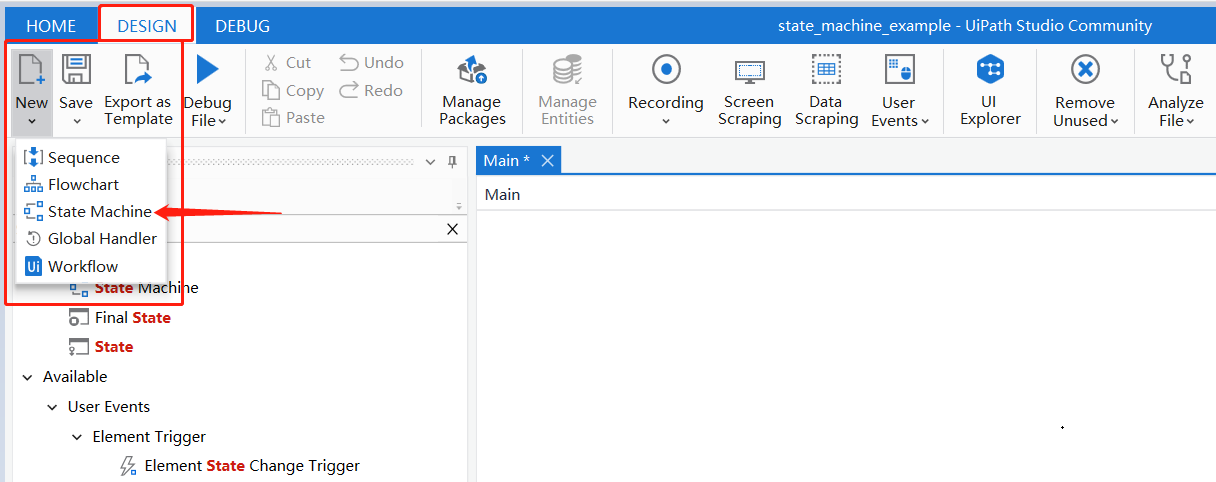
**三、状态机案例**

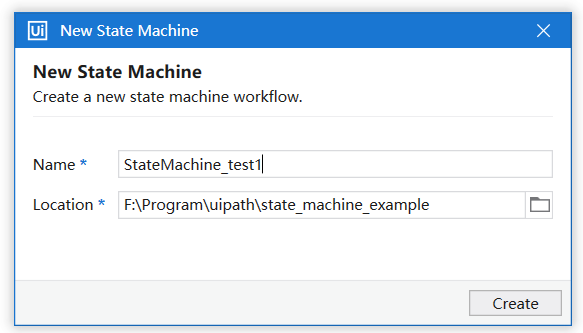
一般来说状态机适合某种情况发生执行另一种情况的流程，比如说这样的一个场景：我们的流程执行是使用一封邮件是否到达来触发，当邮件到达的时候我们就处理邮件，按照邮件附件Excel信息执行某一个RPA流程，执行成功后将成功信息反馈给业务人员；如果没有邮件（一般为指定邮件主题的邮件）到达，我们就反馈其他的信息。这样的一种RPA场景就比较适合使用状态机来执行。

为了让大家更加容易的来了解状态机流程，我们仍然以之前的**猜数字**的游戏来进行举例：

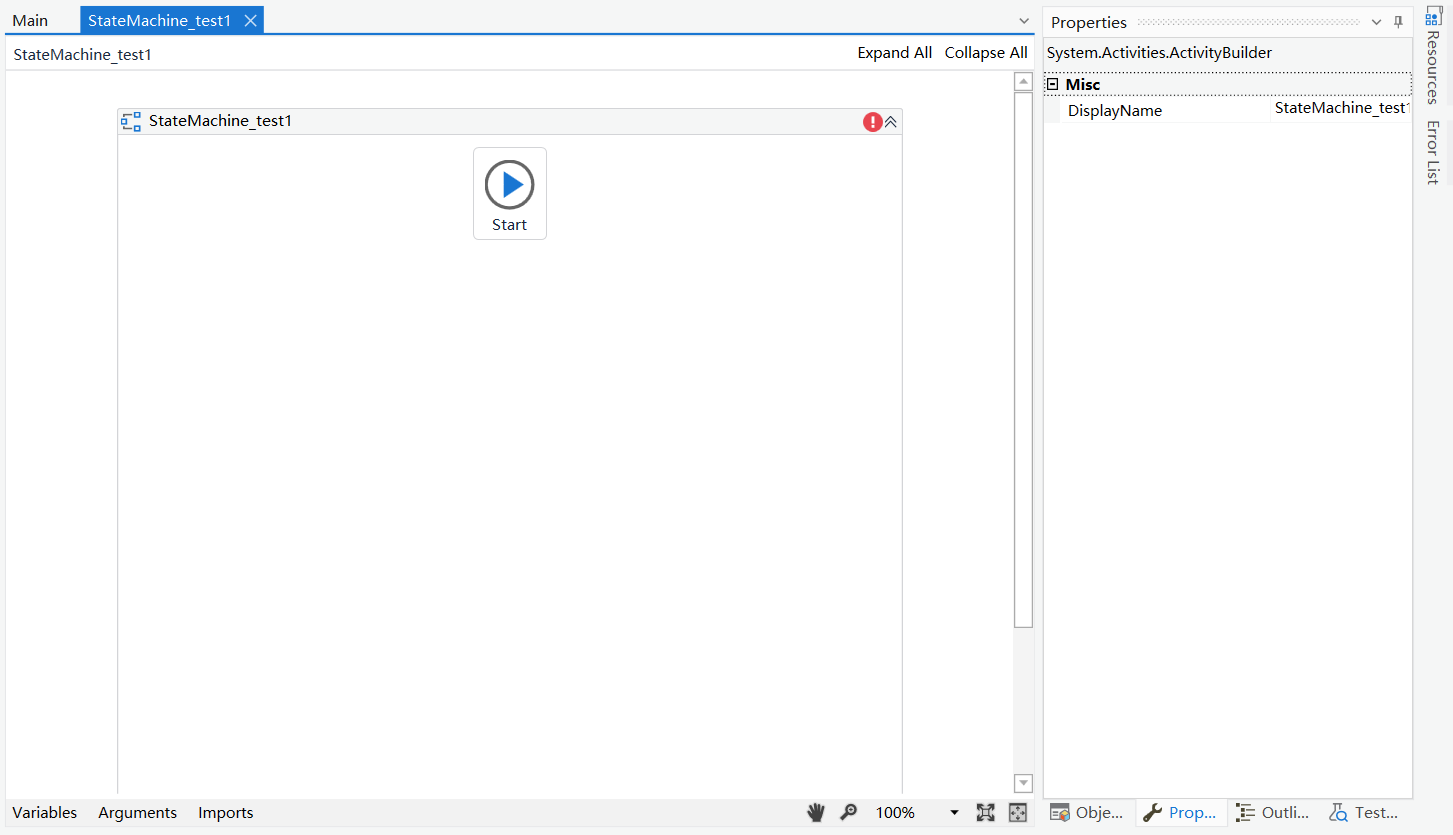
3.1 先新建一个流程名为：“state\_machine\_example”,点击“DESIGN”选择“NEW”

下的“State Machine”并点击它。之后在弹出的状态机对话框中输入State\_Machine\_test1即其他信息如下所示：

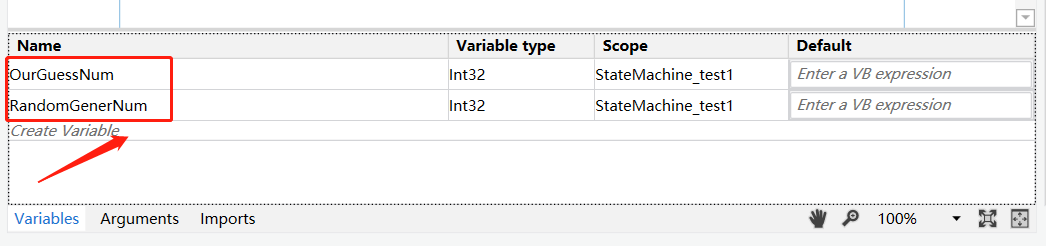




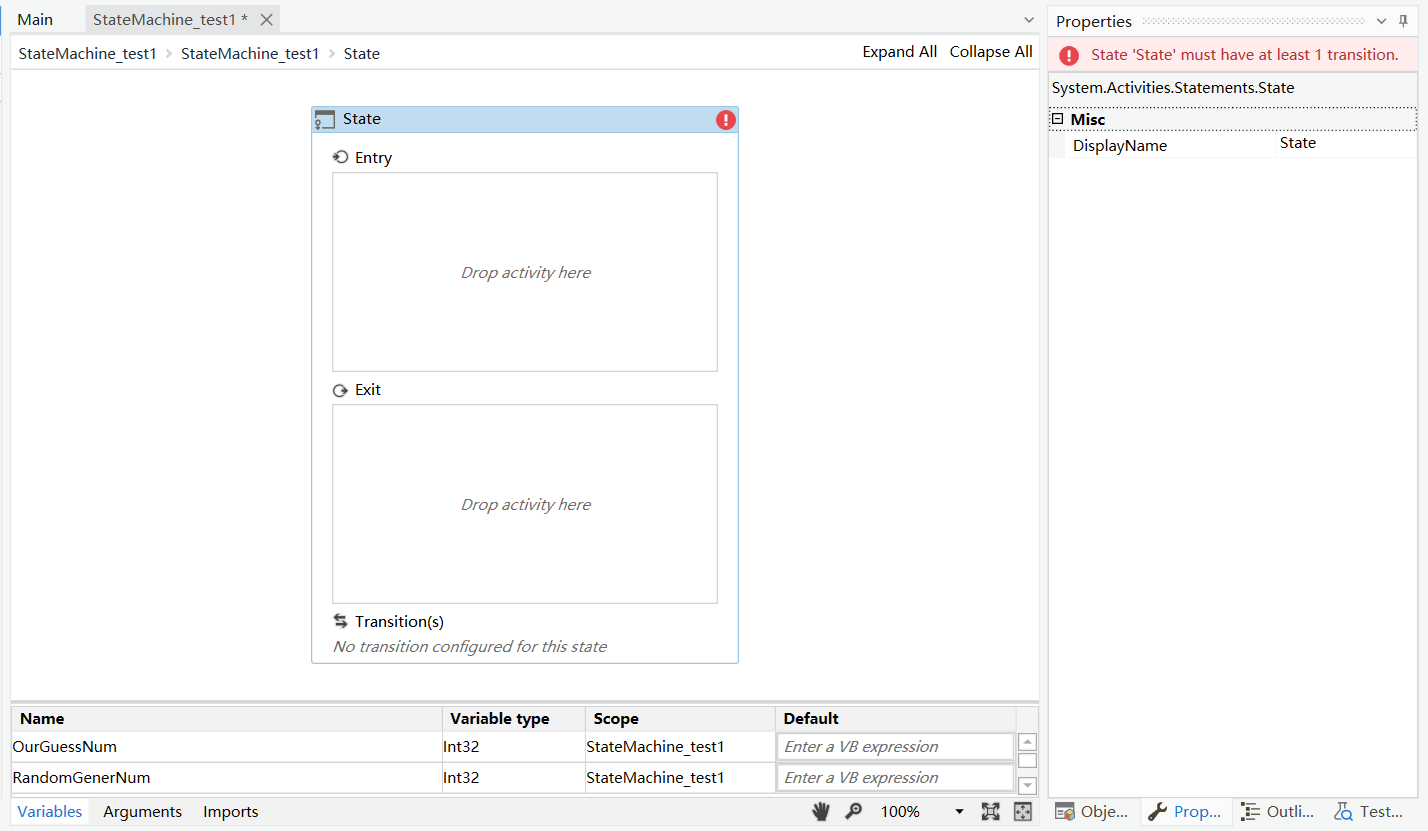
点击create之后的界面如下：



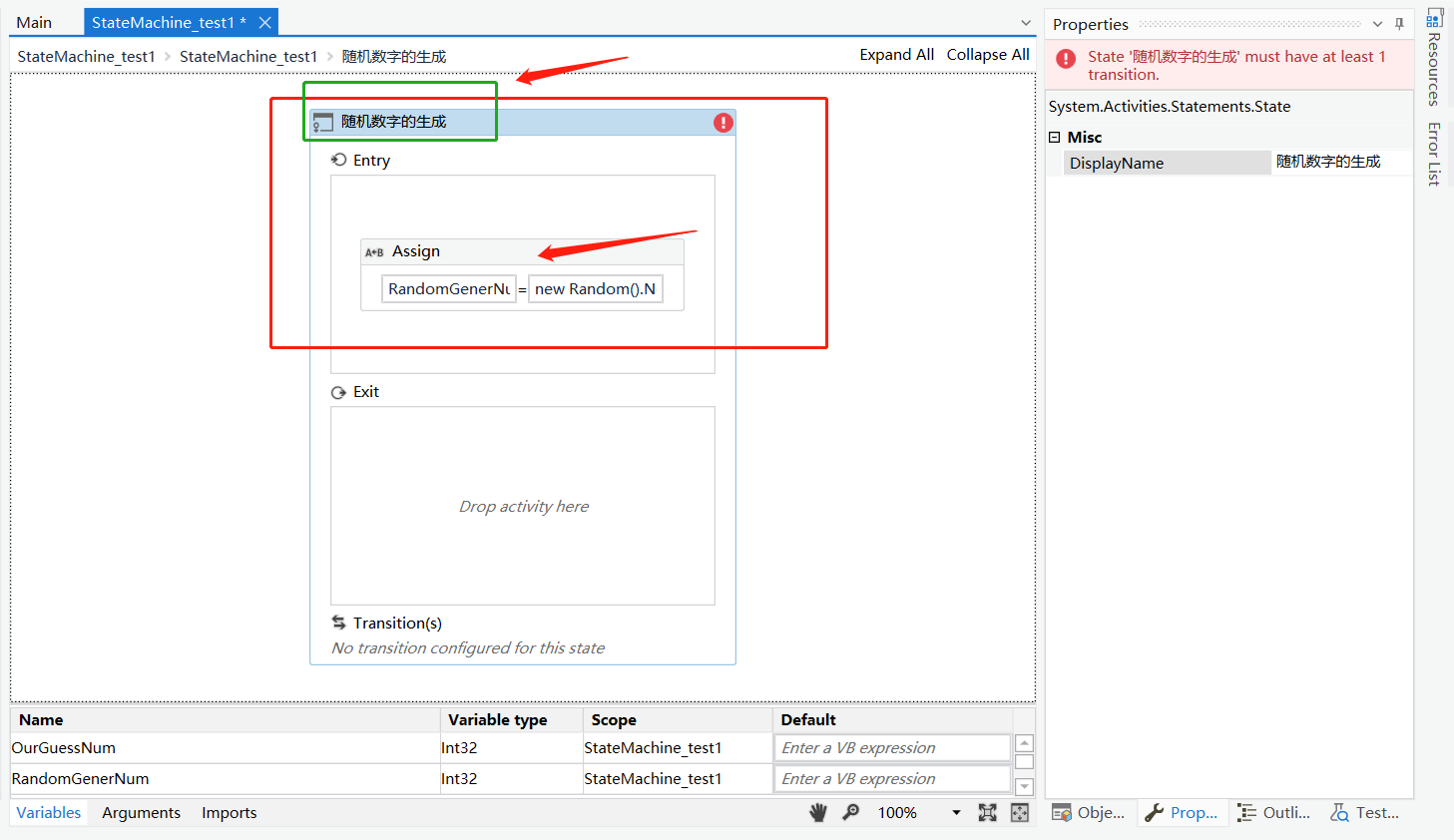
3.2 上述步骤完成之后，我们在变量面板中定义两个变量，一个为我们猜测的数字OurGuessNum，一个为随机产生的数字RandomGenerNum，变量类型我们都选择int32型的,如下所示：



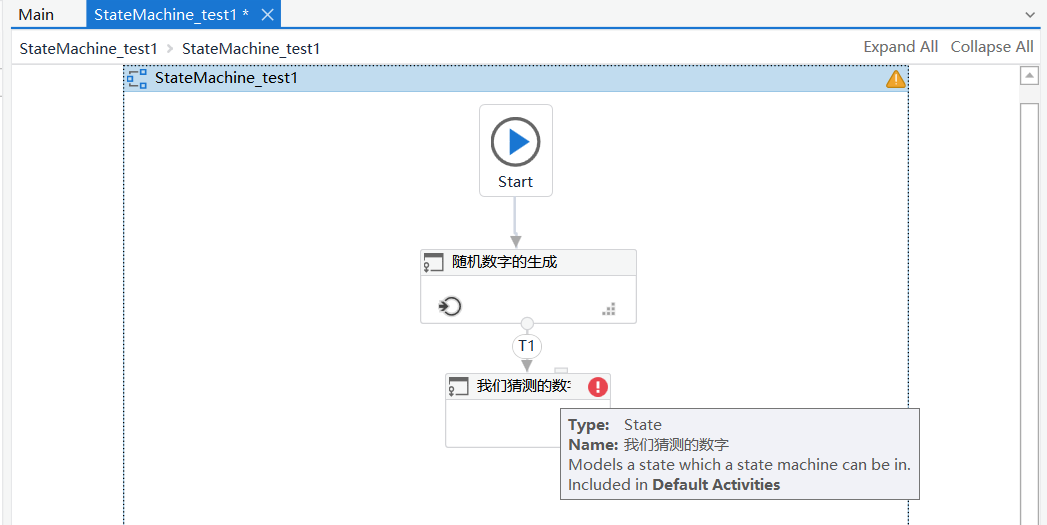
3.3 接着我们拖动一个State活动连接到设计器面板中显示的“start”节点下，并连接起来，然后双击State活动展开它，其在设计面板中的状态如下所示：



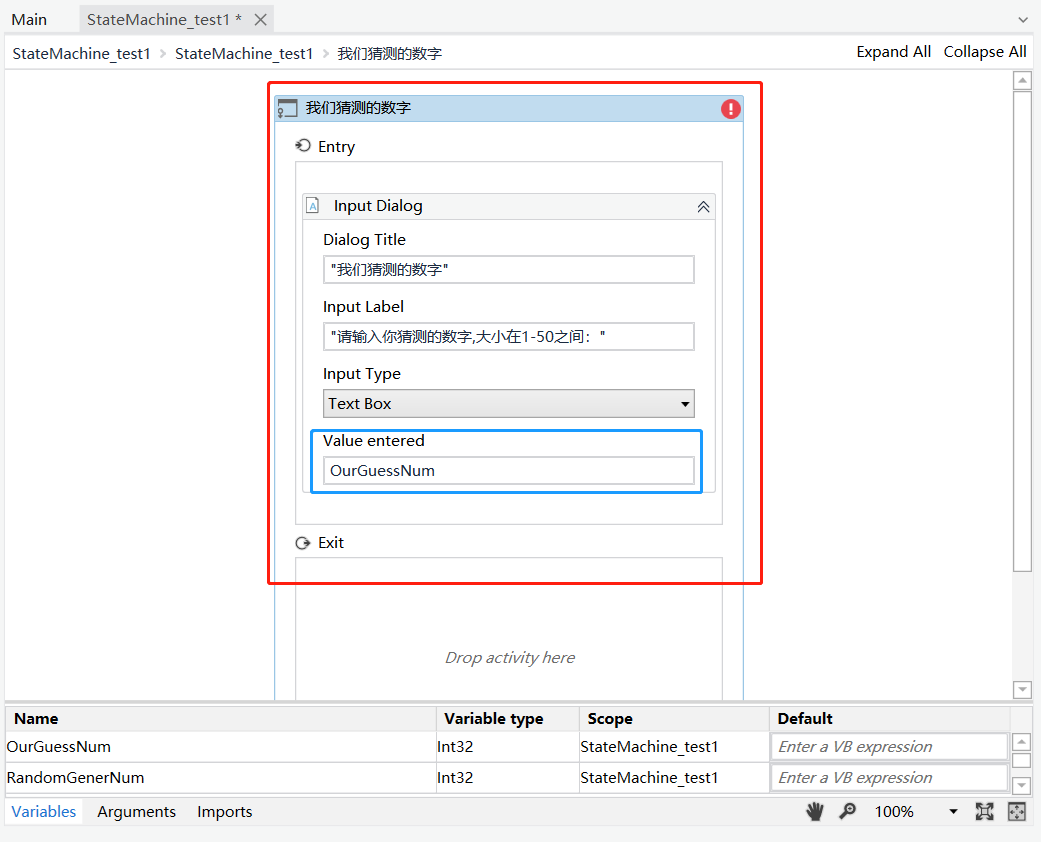
3.4 我们使用中文对State活动重新命名，次处我们将此State命名为：**随机数字的生成。**接着我们拖动一个Assign活动置于Entry当中，并且设置一个表达式：RandomGenerNum = new Random().Next(1,50)，即Assign的To输入框中输入RandomGenerNum ，在Value中输入new Random().Next(1,50)，示意如下：



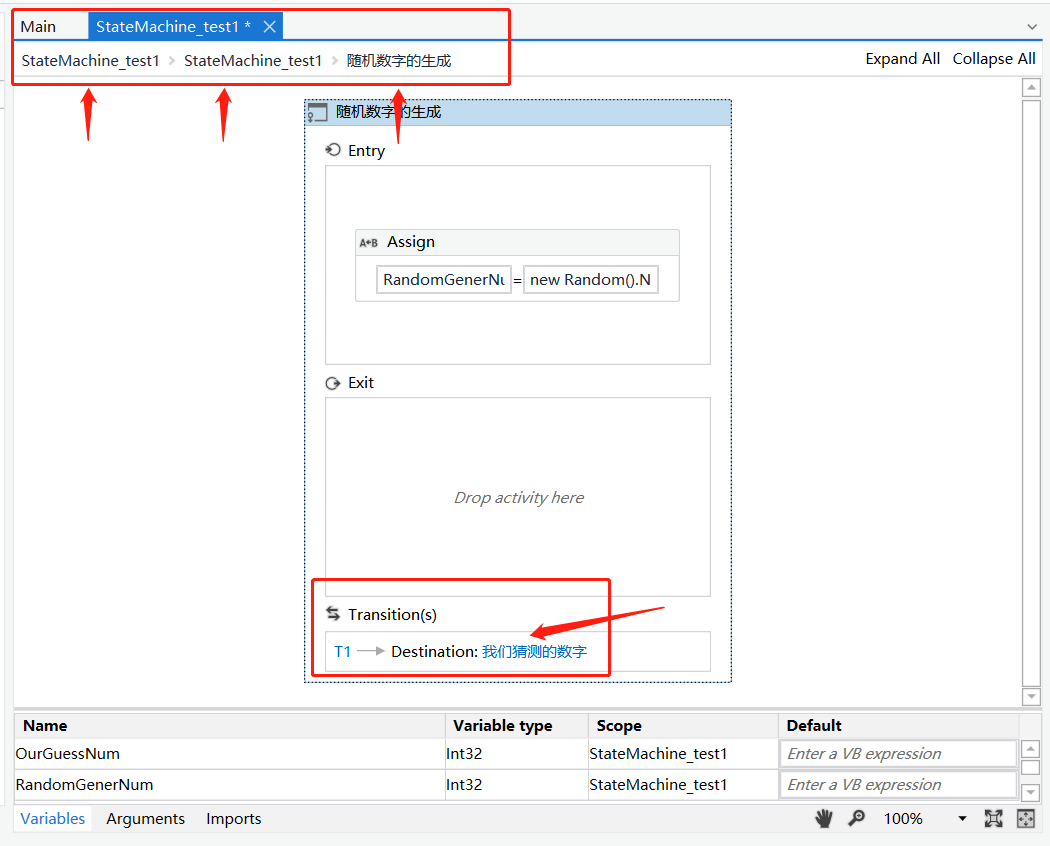
3.5 接着我们在之前的State下再连接一个新的State，这个新的State用来对我们的猜测的数据进行一些操作。连接好新的一个State之后，我们把它的DisplayName设置为**我们猜测的数字**。如下所示：



3.6 我们双击展开这个活动，拖动一个Input Dialog置于Entry中，注意的是在Value entered填写我们之前我们设置的参数OurGuessNum,这个参数值是用来接收猜测的数值的，同时填写其他相应的信息，如下所示：（当然了，这些信息都可以在Properties面板中进行设置的）



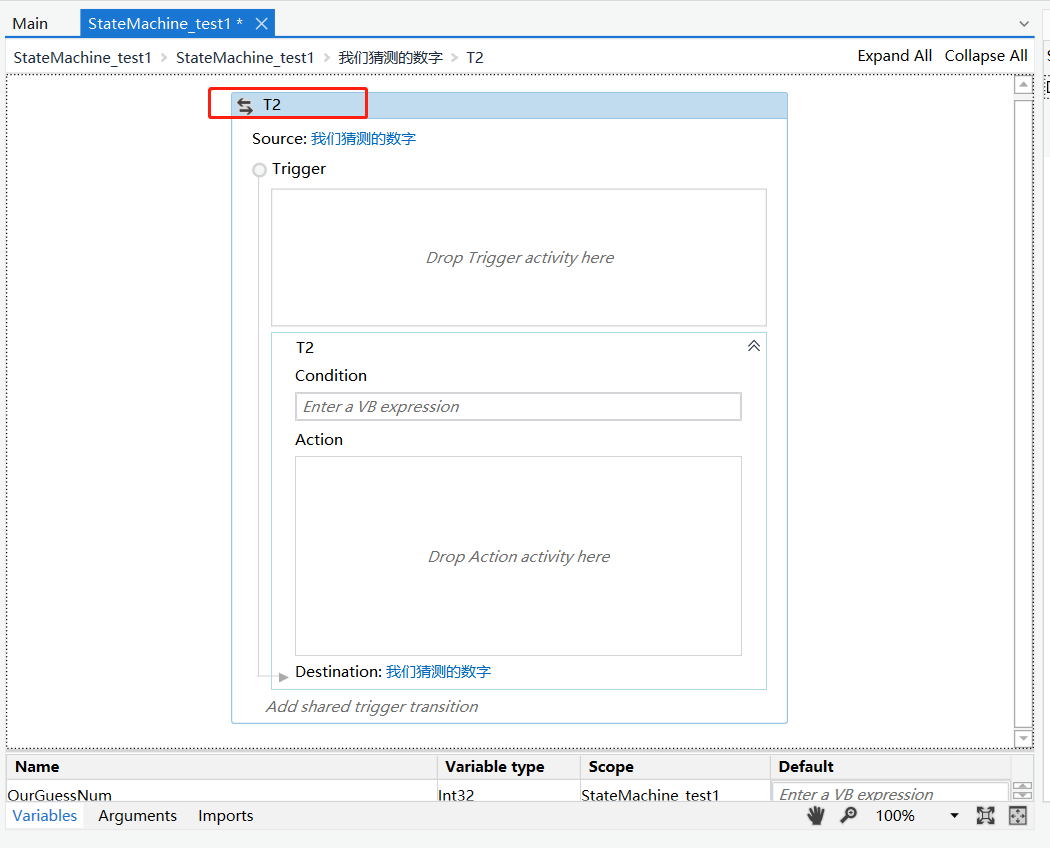
3.7 上述两步设置好之后，我们返回到主视图，并点展开第一个State，如下所示。通过点击下图三个箭头的字可以进行主视图和编辑视图的切换（点击第一个箭头所指地方就可以回到主视图）。可以看出第一个State下的Transitions(s)已经有了一个T1内容出现，这里我们先不解释它的含义。



3.8 在上述步骤的基础之上，我们新建一个转换，这个转换从“我们猜测的数字”指向自身，如下所示：



3.9 点击T2 Transition（红色框选位置），展开设计面板，如下所示：



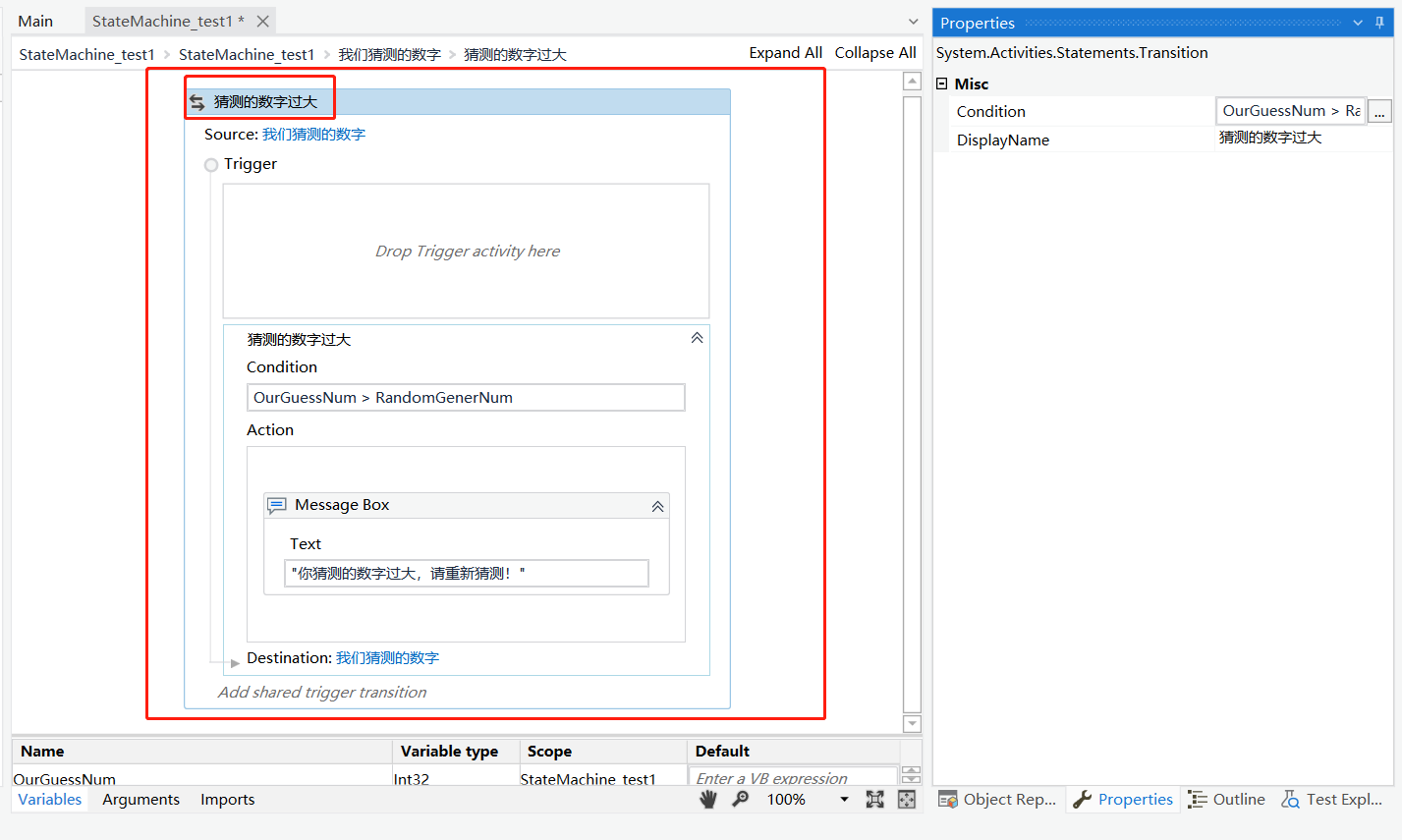
可以看出，Transition被展开以后有三个部分：

Trigger（触发器）

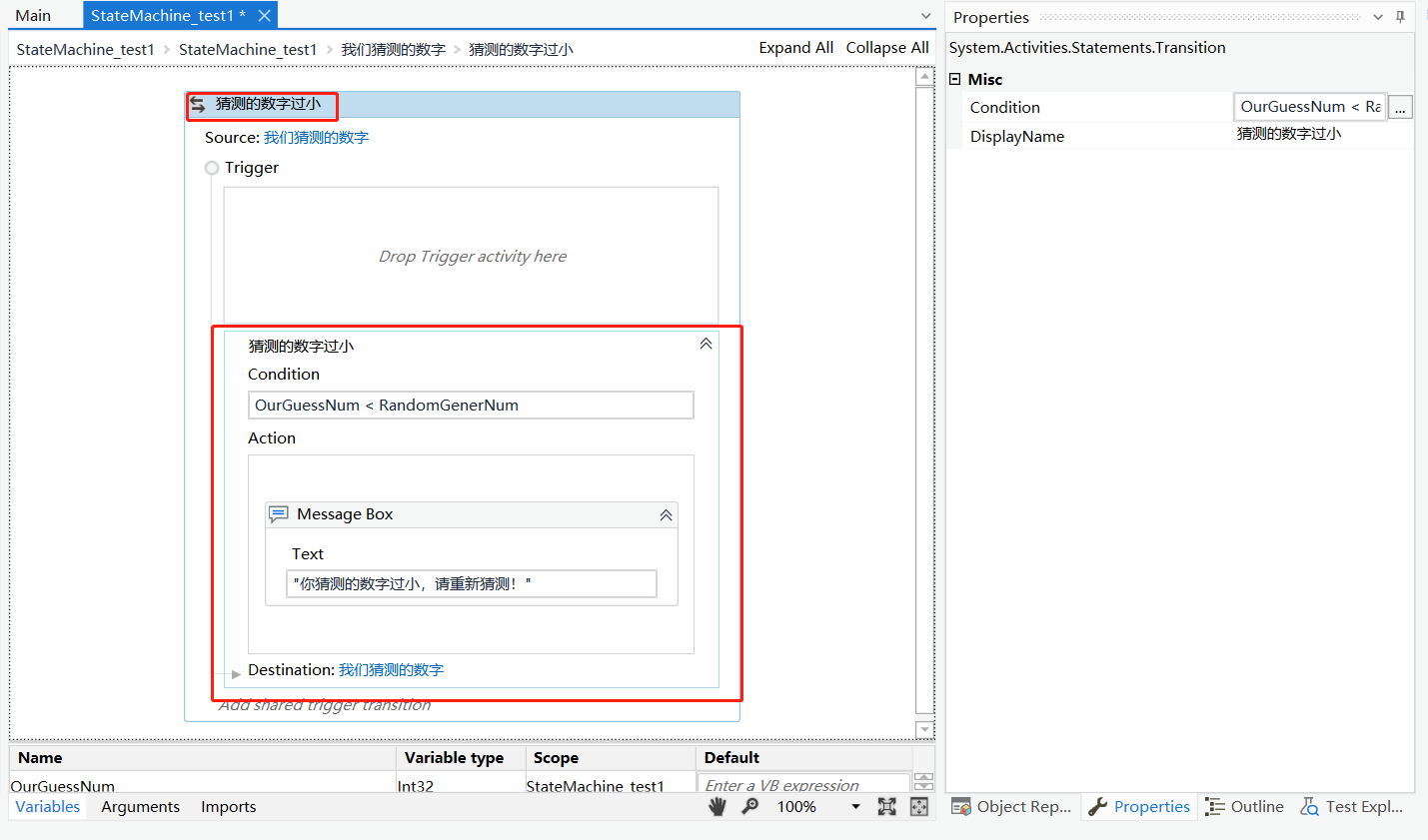
Condition（条件）

Destination（目标）

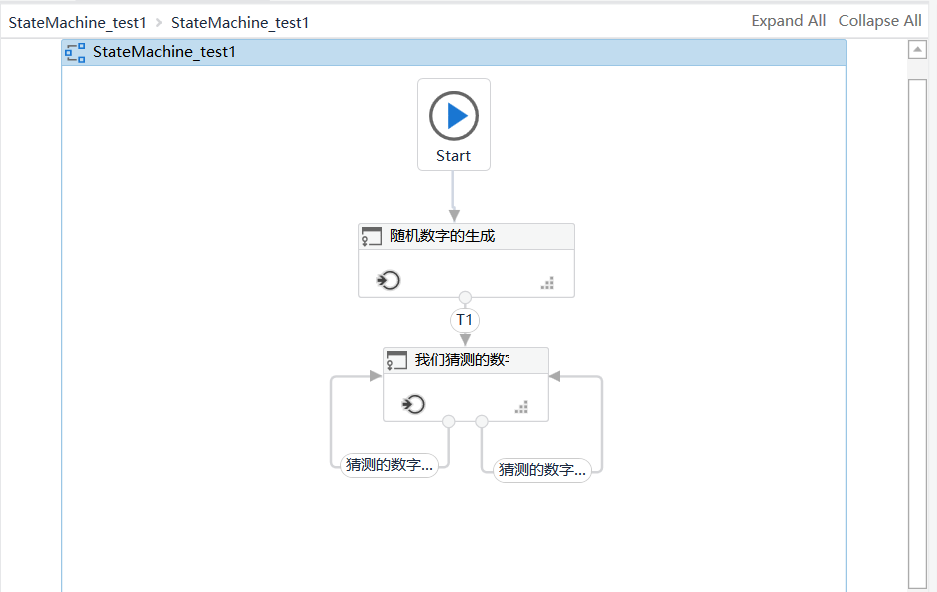
3.10 在展开的T2中我们进行输入一些信息，在Condition中输入两个数字的比较表达式：OurGuessNum > RandomGenerNum,之后拖动一个Message Box到Action中，并输入内容，最后我们修改一下这个Transition的名字为：猜测的数字过大。如下所示：



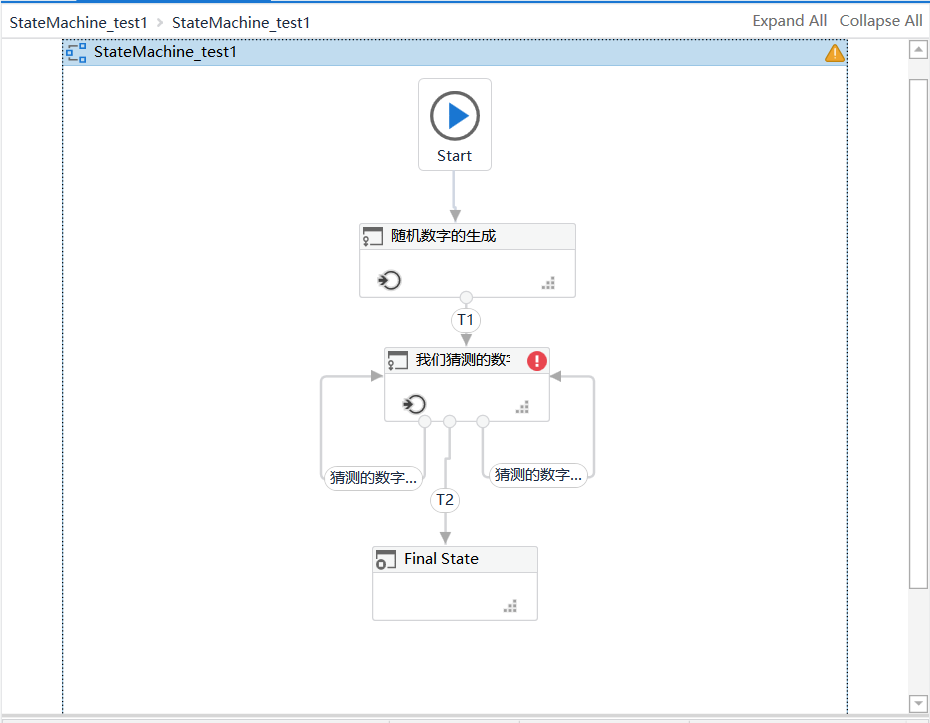
3.11 之后同样的操作，我们再创建一个指向自身的Transition，并双击展开。在展开的Condition中输入两个数字的比较表达式：OurGuessNum < RandomGenerNum,之后拖动一个Message Box到Action中，并输入内容，最后我们修改一下这个Transition的名字为：猜测的数字过大小。如下所示：



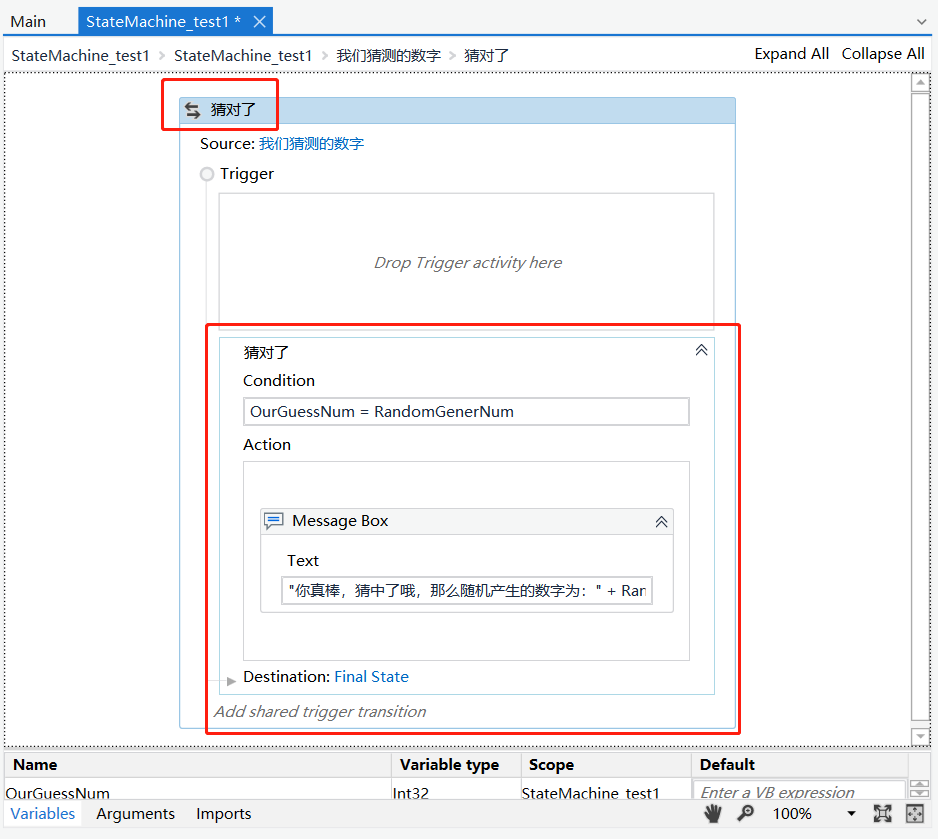
返回项目主视图，如下所示：



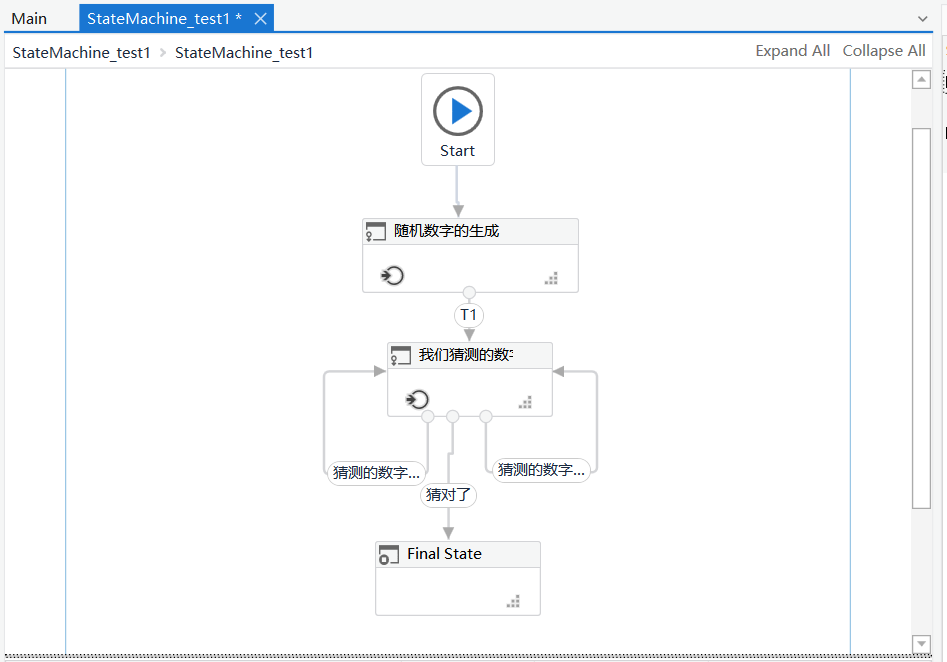
3.12 在此基础之上，我们拖动一个Final State到主视图中并连接到“我们猜测的数字”State上。如下所示：



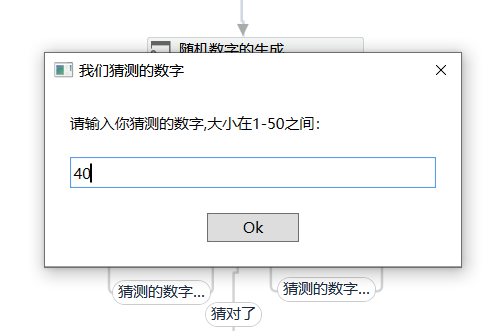
3.13 之后双击展开连接“Final State”的T2,之后在条件（Condition）中输入：OurGuessNum = RandomGenerNum，并拖动一个Message Box 置于Action中，并输入:你真棒，猜中了哦，那么随机产生的数字为：！ + RandomGenerNum.ToString，同时更改“T2”为“猜对了”。这里要注意，我们在输入信息的时候要使用**英文的引号**。如下所示：

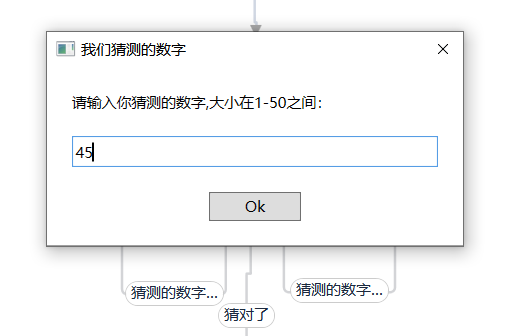


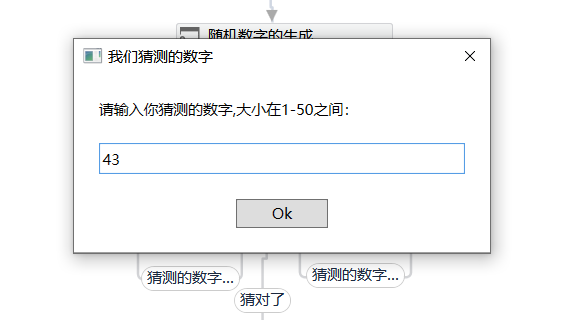
3.14 最后我们回到主编辑视图，可以看到我们建立的流程各个活动如下：

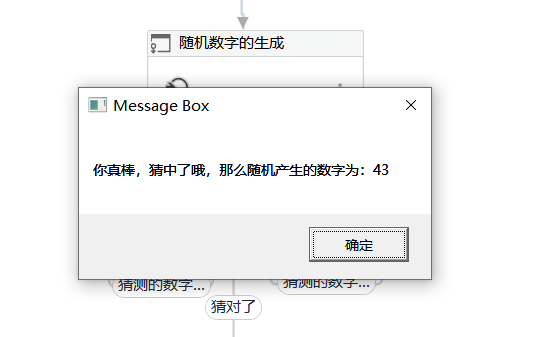


3.15 我们运行按快捷键Ctrl+F6运行此流程,我猜测的过程如下：

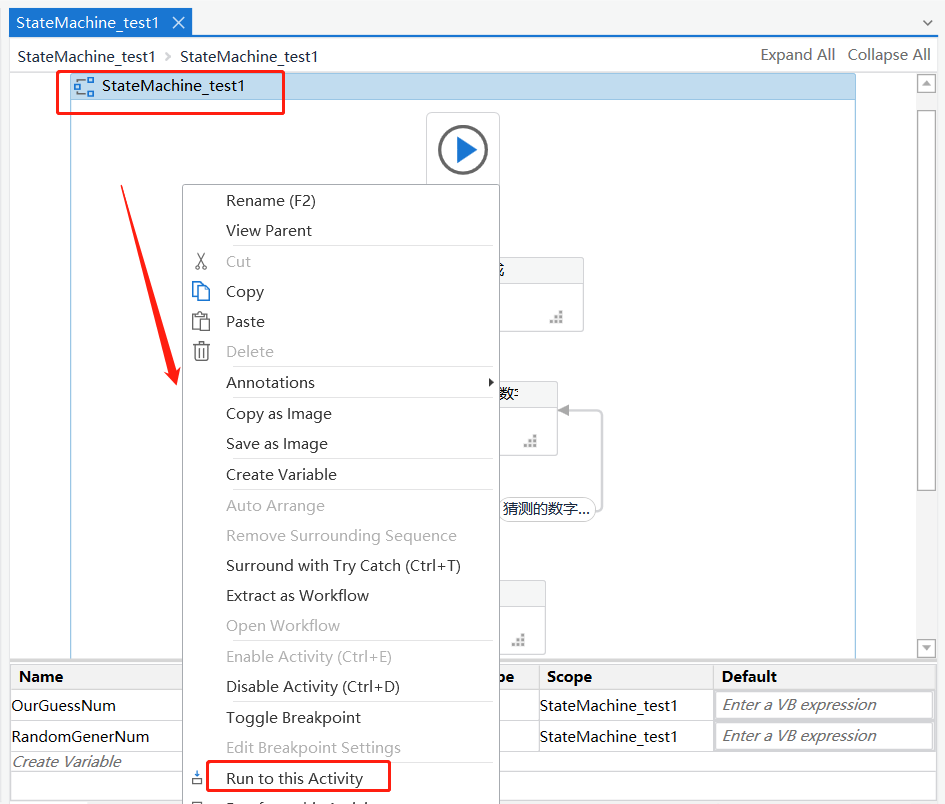




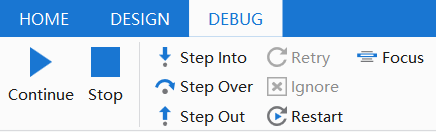




当然了，我们也可以将鼠标置于该状态机的流程中，然后右键选择“Run to this Activity”：



之后点击DEBUG下的Continue即可运行。



1. **总结**

总结一下，我们使用状态机实现了一个猜数字的流程。与之前的FlowChart比较，大家可能会更加的习惯FlowChart进行流程的创建，实际上状态机也是比较好用的。

状态机在流程中的主要功能就是进行状态的转换，在Transition中我们通过设计条件来进行状态的改变。使用UiPath工程师都认为State Machine基本上就是有状态转换功能的Flowchart。

通过在State中添加不同的状态（上述案例3个状态），在状态中我们又添加了一些活动（实现不同的操作），整个State的主要用法就是这样。

最后总结一下优缺点：

**优点：**

1. 有利于较为复杂的连续工作流的创建工作；
2. 使用状态转换可以使得流程更加的灵活。
3. 在一定角度上可以简化流程（比如说复杂的条件判断）

**缺点：**

1. 比Sequence和FlowChart更难理解一些。
2. 每一个状态转换需要非常明确。

还有一些优缺点小伙伴们可以自由补充，这次的文章内容就到这里，感谢大家的阅读，我们下次再见。