

程序启动之后 \_N的添加UIBehaviour

```
private void Awake()
{
    //从自己的子类身上 包括自己 找到 所有的Transform组件
    Transform[] allChild= transform.GetComponentsInChildren<Transform>();

    for (int i = 0; i < allChild.Length; i++)
    {
        //N 结尾的 表示将来用的上的控件
        if(allChild[i].name.EndsWith("_N"))
        {
            allChild[i].gameObject.AddComponent<UIBehaviour>();
        }
    }
}
```

UIBehaviour主动注册到UIManager

```
private void Awake()
{
    //找到自己属于哪个子控件
    UIBase tmpPanel= transform.GetComponentInParent<UIBase>();

    Debug.Log("panelname==" + tmpPanel.name + " myName=" + transform.name);
    //将子控件 主动注册到 Ulmanager 里面
    UIManager.Instance.RegistGameObject(tmpPanel.name, transform.name,gameObject);

}
```

UIBase

The screenshot shows the Unity code editor with the file `UIBase.cs` open. The code defines two methods: `GetGameObject` and `GetBehaviour`. Both methods include XML-style documentation blocks for their parameters and return types.

```
cs UIBase.cs X UIBase Awake()
}
/// <summary>
/// 拿到物体
/// </summary>
/// <param name="widgeName"></param>
/// <returns></returns>
GameObject GetGameObject(string widgeName)
{
    return UIManager.Instance.GetWidget(transform.name, widgeName);
}

/// <summary>
/// 拿到behaviour
/// </summary>
/// <param name="widgeName"></param>
/// <returns></returns>
UIBehaviour GetBehaviour(string widgeName)
{
    GameObject tmpObj = GetGameObject(widgeName);

    if (tmpObj != null)
    {
        UIBehaviour tmpBehaviour = tmpObj.GetComponent<UIBehaviour>();

        return tmpBehaviour;
    }
    return null;
}
```

UIBase

```
public void AddButtonListen(string widgeName, UnityAction callBack)
{
    UIBehaviour tmpBehaviour= GetBehaviour(widgeName);

    if(tmpBehaviour != null)
        tmpBehaviour.AddButtonListen(callBack);
}
```

## UIBehaviour

```
public void AddButtonListen(UnityAction callBack)
{
    Button tmpBtn = GetComponent<Button>();

    if(tmpBtn != null)
    {
        tmpBtn.onClick.AddListener(callBack);
    }
}
```