**What is ChildActionOnly Attribute in ASP.NET MVC?**

Suppose you have a scenario where you have one action method and you don’t want that action method to be invoked via URL rather you want that action method to be invoked by other actions of your application. Then in such scenarios, **ChildActionOnly Attribute**can be handy. So, when we decorate an action method with the ChildActionOnly attribute, then it is called child action in ASP.NET MVC Application and child action methods are only accessible by a child request. That means once an action method becomes a child action, then it will not respond to the URL requests rather it will be invoked by other action methods of your application. Let us understand this with an example.

**Step1:** Create an empty ASP.NET MVC 5 application

**Step2:** Add HomeController. Copy and paste the following code. As you can see, here, we decorate the Countries action method with the ChildActionOnly attribute. Now, this method is accessible only by the child request. A runtime exception will be thrown if we try to access this method through a URL request.

**public** **class** HomeController : Controller

**{**

// Public action method that can be invoked using a URL request

**public** ActionResult Index**()**

**{**

**return** View**()**;

**}**

// This method is accessible only by a child request. A runtime

// exception will be thrown if a URL request is made to this method

**[**ChildActionOnly**]**

**public** ActionResult Countries**(**List**<**String**>** countryData**)**

**{**

**return** View**(**countryData**)**;

**}**

**}**

**Step3:** Right click on the **“Countries()”** action method and add the **“Countries”** view. This view will render the given list of countries as an unordered list.

@model List**<string>**

@foreach (string country in Model)

{

**<ul>**

**<li>**

**<b>**

@country

**</b>**

**</li>**

**</ul>**

}

**Step4:** Right click on the **“Index()”** action method and add the **“Index”** view. Copy and paste the following code in it. Notice that, here we are using the Action HTML helper method to invoke the childaction.

**<h2>**Countries List**</h2>**

@Html.Action("Countries", new { countryData = new List**<string>**() { "US", "UK", "India" } })

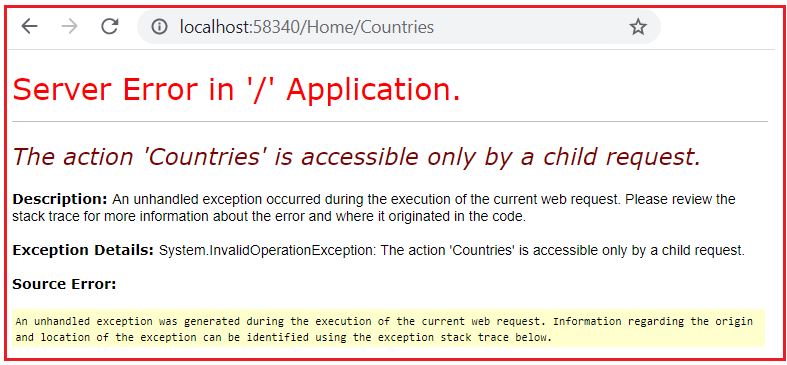
**Please Note:** The Child actions can also be invoked using the **“RenderAction()**” HTML helper as shown below.

@**{**

Html.RenderAction**(**"Countries", new **{** countryData = new List**<**string**>()** **{** "US", "UK", "India" **}** **})**;

**}**

Now run the application and navigate to the **Home/Index** action method and internally Index view make a call to the Countries child action which will return the data and those data you can see on the Index view as expected. Now, let’s try to access the Countries action method from the URL by making a request to Home/Countries and see what happens. We will get the following error page which clearly saying that the action Countries is accessible only by a child request.



**Points to remember while working with ChildActionOnly Attribute in ASP.NET MVC**

1. The child action methods will not be responded to incoming URL requests. If you try to invoke the child actions using URL, then you will get a runtime error saying – **Child action is accessible only by a child request**.
2. You can only access the child action methods by making a child request from a view either by using the “**Action()**” or “**RenderAction()**” HTML helper methods.
3. The most important point that you need to remember is, an action method doesn’t need to have the **ChildActionOnly** attribute to be used as a child action. You can access the normal action methods by making a child request. You need to use the **ChildActionOnly** attribute only if you want to prevent the action method to be invoked using URL.
4. Child action methods are different from the NonAction methods in MVC application. The difference is that the NonAction methods cannot be invoked as a child request either by using the Action() or RenderAction() HTML helpers.
5. The main advantage of Child Action method is that you can cache portions of a view. We will discuss this in **[OutputCache Attribute](https://dotnettutorials.net/lesson/outputcache-attribute-mvc/)** article.