This is continuation to [Part 93](http://csharp-video-tutorials.blogspot.com/2013/09/part-93-what-is-ajax-and-why-should-we.html), Please watch [Part 93](http://csharp-video-tutorials.blogspot.com/2013/09/part-93-what-is-ajax-and-why-should-we.html) from the [ASP.NET MVC tutorial](http://www.youtube.com/user/kudvenkat/videos?view=1&flow=grid)before proceeding with this video.  
  
**One problem with partial page updates is that, they donot provide any visual feedback if the request is taking a long time to process**. With full page postbacks, we don't have this problem, because the entire page is posted to the server and hence the user knows something is happening.  
  
   
  
For example, let's say a request in an ajax enabled application takes 10 seconds to complete. When the user clicks the button and if we don't provide him with any visual feedback that the request is being processed, the user may think the website is not working and he may click the button again. So, it is very important that we provide some visual feedback. To achieve this here are the steps.  
  
**Step 1:** Right click on the project name in solution explorer, and a folder with name = Images. Download and paste the following animated GIF image, in the Images folder.  
http://2.bp.blogspot.com/-1yBNkrITOKA/Uji72zy_uUI/AAAAAAAAF-M/EN0YeLRA5Hs/s1600/spinner.gif   
  
Please note: You can create animated GIF images using the following website.  
[http://spiffygif.com](http://spiffygif.com/)  
  
**Step 2:** Place the following DIV tag on the view, where you want the image to show up when request is still being processed.  
<div id="divloading" style="display:none;">  
   <img src="~/Images/spinner.gif"/>  
</div>  
  
**Step 3:** Modify the Ajax.ActionLink() html helper method on the view as shown below. Notice that, we have set **LoadingElementId = "divloading"**  
@Ajax.ActionLink("All", "All",   
    new AjaxOptions   
    {  
      HttpMethod = "GET",   
      UpdateTargetId = "divStudents",   
      InsertionMode = InsertionMode.Replace,   
      LoadingElementId = "divloading"  
    })  
  
**Step 4:** The controller action method in our example, completes in no time. So the image never shows up. To introduce some latency include, System.Threading.Thread.Sleep(1000); statement in the controller action method. The duration for the sleep method is in milliseconds. We have specified 1000 mill-seconds, so the thread is going to sleep for 1 second.  
public PartialViewResult All()  
{  
    System.Threading.Thread.Sleep(1000);  
    List<Student> model = db.Students.ToList();  
    return PartialView("\_Student", model);  
}  
  
Now run the application and notice that image shows up, while the request is still being processed. When request processing completes and when the response is received, the loading image disappears and the UI is automatically updated, with the data received.