|  |  |
| --- | --- |
| **What is Lazy Loading or Deferred Loading? How to implement Lazy Loading in .Net with C#?**  Lazy loading is a technique, pattern where a data is loaded only on demand or only when it is required in order to increase the program efficiency, performance, etc.  According to wiki,  *Lazy loading is a design pattern commonly used in computer programming to defer initialization of an object until the point at which it is needed.*  Lazy loading is also called deferred loading or on-demand loading. It is called so because, usually the data for an object will be loaded whenever it is initialized but in this technique the loading is postponed or deferred till there is a requirement for the data. This is done when the data is really big or involves some sort of complexities in loading and it may not be used frequently. |  |
| |  |  | | --- | --- | | In some cases, users may not be interested to view entire data too. Hence, loading entire data is not that great idea in these scenarios. In these cases, one can decide to lazy load the data depending upon their situations for gaining better performance, etc.  Moving forward, we will see how to lazy load data in C#.  For simplicity and easy understanding, consider an Employee details table which has all employee related data including a picture of the employee. Refer the below figure,  Lazy Loading in C#    In this case, whenever we need to load the employee details it will be a bad idea to load employee data including the picture column in the initial data load. It is always good and efficient to load the picture only when it is required instead of loading it initially because there may be cases it may not be used. Since Picture is a binary data, loading it whenever it is required will in turn give you some performance benefits.  Let’s see how we can implement this in the Employee entity.  Lazy Loading in C#  public class Employee  {  int \_empID;  string \_empname = String.Empty;  int \_dept;  int \_age;  string \_address = String.Empty;  byte[] \_picture = null;    public Employee()      {               //               // TODO: Add constructor logic here               //      }        #region Public Members  public int EmpID      {  get          {  return \_empID;          }  set          {              \_empID = value;          }      }    public string EmpName      {  get          {  return \_empname;          }  set          {              \_empname = value;          }      }    public int DeptID      {  get          {  return \_dept;          }  set          {              \_dept = value;          }      }    public int Age      {  get          {  return \_age;          }  set          {              \_age = value;          }      }    public string Address      {  get          {  return \_address;          }  set          {              \_address = value;          }      }  **public byte[] Picture**  **{**  **get**  **{**  **if (\_picture == null)**  **\_picture = GetPicture();**  **return \_picture;**  **}**  **set**  **{**  **\_picture = value;**  **}**  **}**      #endregion    private byte[] GetPicture()      {  byte[] pic = null;          //get image code  return pic;      }  }  If you see the above code, the Picture property has the capability to load the data only when it is accessed (in other words, the picture will be loaded only when it is required). So, whenever you want to load the employee details from database you can initialize all the other properties except Picture which can be lazy loaded at any time. Again, to re-iterate you can decide to do this only in cases where you have a need or when this will benefit your application, because doing like this may have adverse effect like increased the number of DB calls, etc.  Lazy loading is also adapted in .Netframework’s ORM tools LINQ to SQL and LINQ to Entities. Let’s see about this in next section. |  | |  | | **Immediate Load or Eager Load**  At some point in time, we may also have a need to load the Picture property along with the other properties. This is called immediate load or eager load, an opposite term of lazy load.  To eagerly load or immediately load the Picture with Employee object we can use DataLoadOptions (namespace:System.Data.Linq)object to specify the framework to load Picture object along with the main object. Refer the below code,    EmployeeDataClassesDataContextdatacontext = new EmployeeDataClassesDataContext();  Var emps = from em in datacontext.Employees  Select em;            DataLoadOptions dboption = new DataLoadOptions();  dboption.LoadWith<Employee>(lemp =>lemp.Picture);  datacontext.LoadOptions = dboption;    foreach (Employee emp in emps)          {  string name = emp.EmpName;  byte[] pic = emp.Picture.ToArray();          }    You can enable the profiler to check the query sent to SQL server when executing the above code. Refer the below figure,  Lazy Loading vs Eagerly Loading  If you would like to know more about DataLoadOptions and Immediate load you read the below article,  [Using DataLoadOptions to Fetch(Immediate Load) the Related Objects in LINQ](http://www.codedigest.com/Articles/ASPNET/364_Using_DataLoadOptions_to_Fetch(Immediate_Load)_the_Related_Objects_in_LINQ.aspx) | | |
| **Conclusion**  Lazy loading is a very good pattern to opt for when we look to make our application efficient. One has to do a proper analysis on the data flow nature of the application to decide whether to use it not to use. Lazy loading technique can also be used in presentation level. For example, when you categorize the stuffs you are offering in your site it is best to show only main categories instead of showing its sub categories too in a shot. Looking at the categories the users may be able to select the category of their interest and expand the same to see it sub categories. It is in this case you can decide to lazy load the sub categories of the items you are offering in your website. By doing this, you are offering your users a better experience, performance and also efficiently managing screen real estate. Examples:  [Lazy Loading jquery Accordion Panel in ASP.Net](http://www.codedigest.com/Articles/jQuery/337_Lazy_Loading_jquery_Accordion_Panel_in_ASPNet.aspx) and [Lazy Loading jQuery Collapsible Panel in ASP.Net Using JSON](http://www.codedigest.com/Articles/jQuery/283_Lazy_Loading_jQuery_Collapsible_Panel_in_ASPNet_Using_JSON.aspx) |  |