Volume 6, Issue 4, April 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Exploring the Benefits of Using Entity Framework at Presentation Layer

Manpreet Kaur, Virrat Devaser
School of Computer Science & Engineering, Lovely Professional University,
Punjab, India

Abstract – In Today's IT world, websites and online portals demand high performance and valid architecture. In any web development, good architecture has become more important for web development. Unfortunately, most of the IT companies place more emphasis on graphic design and layout instead of backend architecture and thus fail to provide efficient product. Backend involves database design and various approaches to access it. Different approaches of writing the code impact the loading time and space. In this paper, our main focus is on the loading of related class properties along with the base class. At database layer, we can avoid to call for related objects within the base objects by the use of entity framework. Entity framework provides the concept of eager loading and lazy loading to handle such situations where we need not to load dependent data explicitly. We can see the similar behavior for the user defined classes when we use properties of the other class types. Thus, we have proposed implementation of Eager loading and lazy loading at presentation layer. The main focus is to reduce memory usage during model initialization for presentation layer using concepts of Lazy and Eager loading.

Keywords— Aspect Oriented Programming, Object Oriented Programming, Lazy Loading, Eager Loading, MVC

I. INTRODUCTION

System memory is very expensive and important part and we should use it wisely. A program is said to be optimized if it consume less memory and resources while running. Every programmer can write a program to solve a problem, but writing an optimized program is a challenge. Modern programming provided inbuilt memory management techniques to minimize consumption of system memory; but still usage of memory depends on various other factors like our programming technique and our selected program architecture etc. When the object of a class is created, memory is allocated for it, the constructor is called and the object is considered live. So we should always avoid creating unnecessary class variable/objects as they consume a very expensive part of system "Memory".

If we come to modern programming framework like Model-View-Controller (MVC), model is used to pass data to presentation layer with the help of controller. Now it depends on us how we utilize these models, so that it consumes less memory. When we initialize model (class), it initializes all its nested classes in constructor. But sometimes, we need not to initialize all classes and memory allocation can be avoided. For example, we consider a web page that displays user personal details only. Suppose we have created a common model for holding user data (personal data, official data, an academic data). Now when we pass model to our presentation layer we will create an instance of this model class, then we'll fill this model from database with a particular user details and further pass this model to view (presentation layer) for displaying it on webpage. However if we notice here; we need to view user's personal data only, but when we initialize model it will also initialize its related components that are its official and academic data. But at this moment, we don't need these components and as a programmer view, this is nothing but wastage of memory as we don't use these components, but internally it will consume memory. One way to solve this is to separate all users related components; but that was not a good idea. Other way that we can follow is to implement the concepts of lazy and eager Loading at presentation layer. It is same type of concept as it is used in Entity Framework at database layer.

II. BACKGROUND & TERMINOLOGY

MVC is a framework that was built to be used in web applications. It has three components that separates different building blocks of an application; Model, View and Controller.

- The Model represents the application core (for instance of list of database records).
- The View displays the data (the database records).
- The Controller handles the input (to the database records).

In view component of MVC, we have full control for HTML, CSS and JavaScript. It has two view engines: ASPX and Razor view engine. The MVC that is used at presentation layer itself divides a web application in three different layers:

- The business layer (Model logic)
- The display layer (View logic)
- The input control (Controller logic)

This separation helps programmers to manage complex applications, because one will focus on only one aspect at a time. For example, UI developers can work on the view without any dependency on the business logic. This approach thus provides easy way to test an application. UI developers, backend developers, service programmers and other team members based on the application can work together at the same time in a group as MVC has simplified the group development.

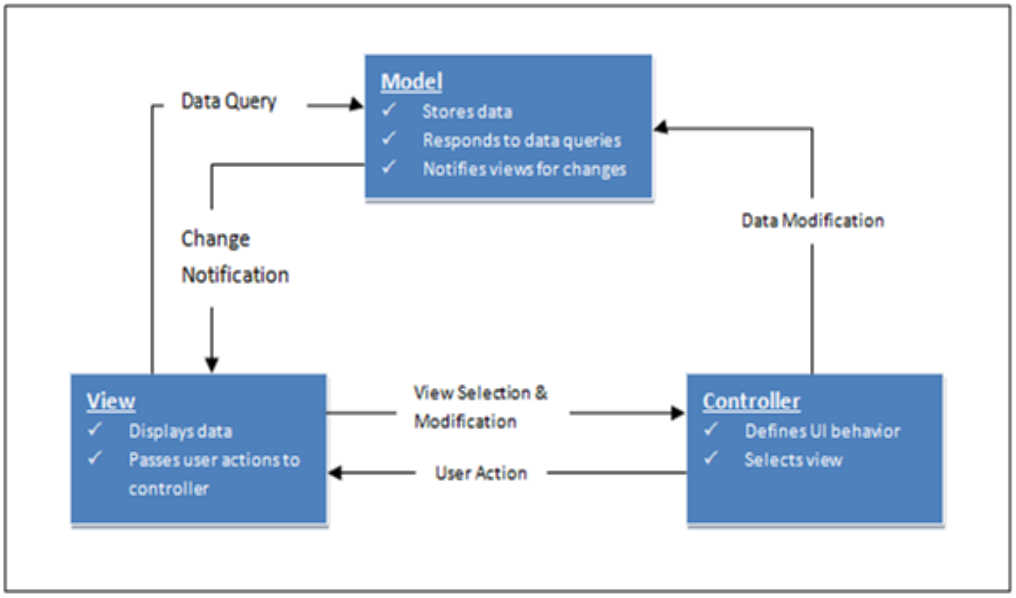

Figure 1

The Microsoft Entity Framework is an ORM framework that enables developers to work with relational data in the form of domain-specific objects, eliminating the need for most of the data access plumbing code that developers usually need to write. Various loading methods of data from database are,

- Lazy Loading
- Eager Loading
- Explicit Loading

In case of lazy loading, related objects are not loaded automatically with its parent object until they are requested from database. This is mainly used when we have a lot of related objects. For example loading all students (related entities) when we fetch details of a university (main entity). For more details, please see figure 2.

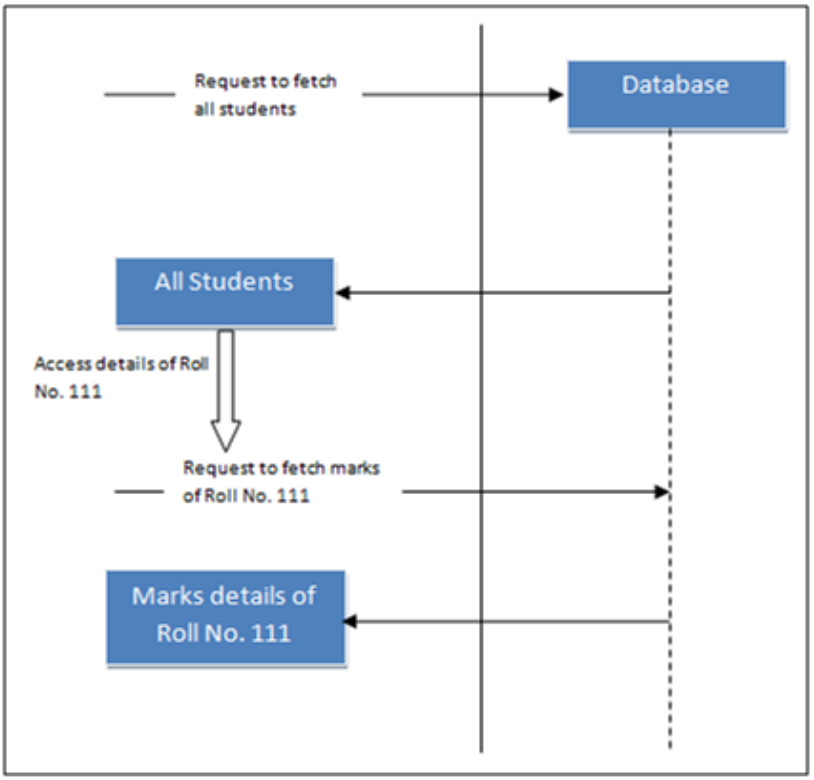

Figure 2

In case of eager loading, related/child objects are loaded automatically with its parent object. This is suitable for situations when we have less number of related objects. For example; loading all subjects (related objects) of a student along with basic details, when we just make a call for basic details of a student (main entity). For more details, please see figure 3.

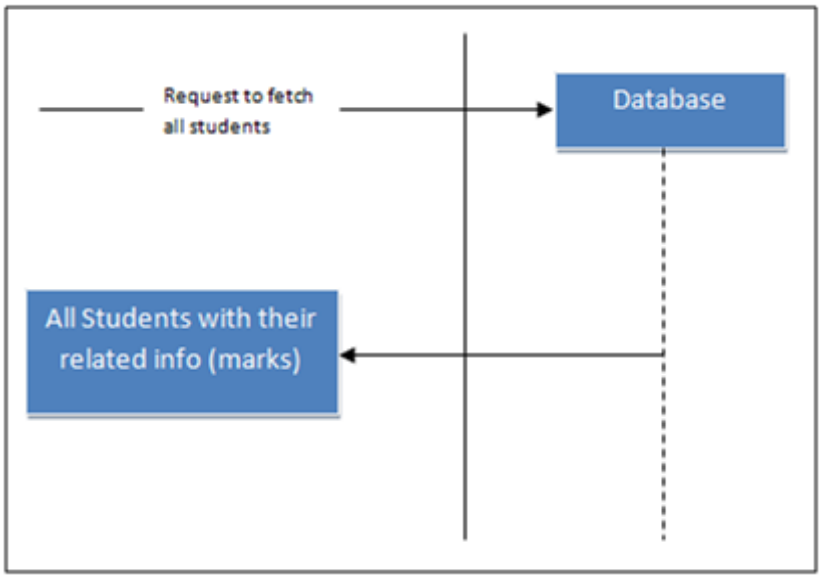

Figure 3

Currently concepts of lazy loading and eager loading limited to database layer only, means we use these mechanisms for fetching data from database. We have proposed the idea to implement same thing in presentation layer to optimize our front end memory consumption as well. For this, we will introduce the concept of Entity Framework at presentation layer while passing data model from controller to view in an MVC framework. By doing so we'll be able to minimize the consumption of memory at user presentation layer. Not only lazy loading but also eager loading is useful in some situations as we explained earlier in above section. We can use existing framework and technology to introduce a new idea that can be helpful in increasing performance of a web application and reducing server load and memory consumption.

III. OBJECTIVES

The main objective of this research is to use system memory in an optimized way and prevent memory wastage and also to make use existing Entity Framework to implement eager/lazy loading in Model component of MVC.

IV. RESEARCH METHODOLOGY

The methodology provides an understanding of implementation of eager/lazy loading in presentation layer. This methodology is based on various articles and websites related to eager and lazy loading. Below are few steps that are concluded, as of now, to solve our main problem.

- Fetch all classes (models) in an object
- Convert all classes (models) to XML data.
- Use XML file as data source of connection string in .edmx file.
- Use this .edmx file at presentation layer.
- Implement eager loading and lazy loading for models.

A. Fetch all classes (models) in an object

Models that we use to pass data from database layer to presentation layer are simple classes that contain various properties and class variables. We define these classes in model section of MVC architecture. We need to fetch all the model classes and its properties that exist under folder Model in an object.

```
modelList =
    GetTypesInNamespace(Assembly.GetExecutingAssembly(),
    "ModelsToXML.Models");
```

```
private Type[] GetTypesInNamespace(Assembly assembly, string nameSpace)
{
    return assembly.GetTypes()
        .Where(t => String.Equals(t.Namespace,
            nameSpace, StringComparison.Ordinal)).ToArray();
}
```


B. Convert all classes (models) to XML data

We need to convert models/classes to XML (Extensible Mark-up Language) data. **XML** is a mark-up language to define a rules set for encoding documents in a format which is both human readable and machine readable. Unlike HTML in XML we can define our own custom tags. We can convert a class to XML easily using various libraries provided by programming languages like XML serialization which is default library provided by Microsoft in .Net framework. Similarly, we can use JAXB in java to convert a class/object to XML.

```
Type[] modelList;
XmlWriter xmlWriter;

public ActionResult Index()
{
    modelList =
        GetTypesInNamespace(Assembly.GetExecutingAssembly(),
        "ModelsToXML.Models");
    xmlWriter = XmlWriter.Create(HttpContext.Server
        .MapPath("/Content/ModelToXML.xml"));
    xmlWriter.WriteStartDocument();
    xmlWriter.WriteStartElement("Models");
    foreach (var model in modelList)
    {
        xmlWriter.WriteStartElement(model.Name);
        var modelProperties = model.GetProperties();
        SetProperties(modelProperties);
        xmlWriter.WriteEndElement();
    }
    xmlWriter.WriteEndElement();
    xmlWriter.WriteEndDocument();
    xmlWriter.Close();
    return View();
}
```

```
private void SetProperties(PropertyInfo[] modelProperties)
{
    foreach (var property in modelProperties)
    {
        xmlWriter.WriteStartElement(property.Name);
        var propertyType = property.PropertyType.Name.ToString();
        if (modelList.Select(m => m.Name).ToArray()
            .Contains(property.PropertyType.Name.ToString()))
        {
            var subProperties = property.PropertyType.GetProperties();
            SetProperties(subProperties);
        }
        xmlWriter.WriteEndElement();
    }
}
```

C. Use XML file as data source of connection string in .edmx file

Now we are ready with XML files of classes/models. We will use this .xml file as data source for our application. There is an inbuilt option in .Net framework that we can provided XML file as data source to the connection string while specifying in web.config file of the application. Also we need to do the same thing in connection setting of Entity Framework that we'll use as data access for the application. By doing so we will be ready with our configuration setting to use .xml file as our data source. Entity Framework adds a file with extension .edmx which contains all the entities of its data source.

D. Use .edmx file at presentation layer

Now we have .edmx file of Entity Framework that hold a data source of type xml which contains definition of our all model used by the application to serve our presentation layer. We need to use this .edmx files in our presentation layer while using model. We'll use only required properties of a class that we need to display or manipulate at that particular view only. In this way only those component will be initialized that are being used through .edmx.

E. Implement eager loading and lazy loading for models

Likewise, we use Lazy and eager loading to improve the performance at database layer, make the use of same concepts at presentation layer to avoid unwanted initialization.

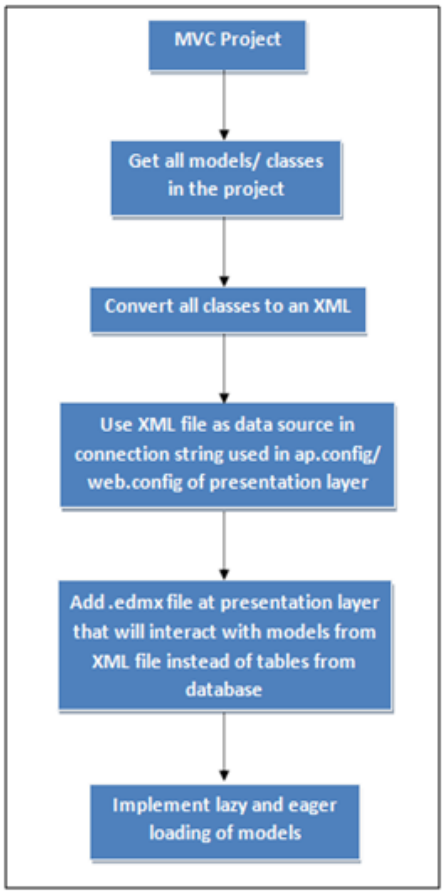

Figure 4

V. EXPECTED OUTCOMES

After implementation of the proposed technique; we will be able to lower the memory wastage due to unnecessary initialization of class components. Existing Entity Framework will be consumed to implement the concept of Eager and Lazy loading in model component of MVC. Class/model definition in model section will not be altered as we can use selected component using .edmx file concept. Newly built architecture will be more robust and with good factor of efficiency.

VI. CONCLUSIONS

The main objective of this research is to build a robust architecture that can provide a better and reliable internet services using stable websites and web portals. Here reliable means a web platform that is fast, secure and less expensive (which consumes less system resource while running). As we are specific to web components in this research area; so proposing an architecture that consumes less system memory (not at the cost of quality) is our make objective. To fulfill this we have proposed Eager/Lazy loading concepts of database in presentation layer of the system. The newly proposed architecture will use same concept of Eager and lazy loading as Entity Framework uses at database access layer. As a result of this we are able to initialize only those components of a class/model which are usable in a particular section. We need not to initialize unwanted components which results into wastage of system memory. The most important thing is that we can use existing Entity Framework for this research as we are using for database access in database layer. We are able to integrate class/model definition XML file to connection string of Entity Framework's ".config" file. With this implementation are able to use Eager loading and Lazy loading concept in front end pages as well. So we can use any of them as per our requirement. In currently existing model system; we initialize all components and sub components of a model class irrespective of the usage of those components in a particular view. With the use of this mythology, we will be able to minimize memory wastage. So, we came with an idea to use existing systems in a new way to form new and stable web architecture.

REFERENCES

[1] I. Boticki, M. Katic and S. Martin, "Exploring the Educational Benefits of Introducing Aspect-Oriented Programming Course", Education, IEEE Transactions, 2013, vol. 56.

[2] Zhongming Xie, Huabing He, Yunfei Li and Juncheng Jia, "Design and application of lighting energy consumption monitoring platform based on MVC and entity framework", Information and Automation (ICIA), IEEE International Conference, 2014.

[3] A. Shaw, "Teaching Students to Design and Implement Social Networks Using MVC as a Capstone Experience", Information Technology: New Generations (ITNG), 2013 Tenth International Conference.

[4] H. Mcheick and Yan QI, "Dependency of components in MVC distributed architecture", Electrical and Computer Engineering (CCECE), 2011.

[5] Weng Wen and Shiming Zhang, "Research and Implementation of AOP Technology In .NET Framework", Enterprise Systems Conference (ES), 2014.

[6] Hui Li, Mingji Zhou, GuiJun Xu and Lingling Si, "Aspect-Oriented Programming for MVC Framework", International Conference on Biomedical Engineering and Computer Science, 2010.

[7] José A. Blakeley, S. Muralidhar and Anil Nori, "The ADO.NET Entity Framework: Making the Conceptual Level Real", 25th International Conference on Conceptual Modeling, Tucson, AZ, USA, 2006.

[8] Atul Adya, José A. Blakeley, Sergey Melnik and S. Muralidhar, "Anatomy of the ADO.NET Entity Framework", ACM SIGMOD International Conference on Management of Data, SIGMOD'07, 2007.

[9] Julia Lerman, "Programming Entity Framework: Building Data Centric Apps with the ADO.NET Entity Framework", O'Reilly Media, Inc., 2010.

[10] http://www.dotnet-tricks.com/Tutorial/entityframework/RIWW210913Differencebetween-Lazy-Loading-and-Eager-Loading.html

[11] http://stackoverflow.com/questions/2990799/difference-between-fetchtype-lazyandeager-in-java-persistence

