

Return list using select new in LINQ

[Ask Question](#)

This is my method which gives me error.

47



12

```
public List<Project> GetProjectForCombo()
{
    using (MyDataContext db = new MyDataContext (DBHelper.GetConnectionString()))
    {
        var query = from pro in db.Projects
                    select new { pro.ProjectName, pro.ProjectId };

        return query.ToList();
    }
}
```

If i change it with this:

```
public List<Project> GetProjectForCombo()
{
    using (MyDataContext db = new MyDataContext (DBHelper.GetConnectionString()))
    {
        var query = from pro in db.Projects
                    select pro;

        return query.ToList();
    }
}
```

Then it works fine no error.

Can you please let me know that how can i return only ProjectId and ProjectName.

Thanks.

[c#](#) [linq](#)

edited Jun 16 '11 at 9:58



Groo

35.8k 14 88 159

asked Jun 16 '11 at 9:54



Sami

1,748 8 29 49

- 1 What's the error? – [Ray](#) Jun 16 '11 at 9:55

what is the List there ? edit : now you have made it clear ;) – [Illuminati](#) Jun 16 '11 at 9:58

- 1 (note that the edit revision seem to imply that I added <Project> in the code, but the problem was that initial code was enclosed in <pre> tags, instead being indented, which

removed angle brackets) – [Groo](#) Jun 16 '11 at 10:03

9 Answers



Method can not return anonymous type. It has to be same as the type defined in method return type. Check the signature of GetProjectForCombo and see what return type you have specified.

Create a class ProjectInfo with required properties and then in new expression create object of ProjectInfo type.

```
class ProjectInfo
{
    public string Name {get; set; }
    public long Id {get; set; }
}

public List<ProjectInfo> GetProjectFo
{
    using (MyDataContext db = new MyD
    {
        var query = from pro in db.Pr
                    select new Projec
    });

    return query.ToList();
}
```

[edited Oct 23 '14 at 23:08](#)

answered Jun 16 '11 at 9:56



[Muhammad Hasan Khan](#)

28.6k 11 75 118

2 Hi Hassan, Thanks for your reply this is really helpfull. I have to know that without creating new class we can also get the required functionality. Is this fine? this works for me fine.

```
public List<Project>
GetProjectForCombo() { using
(MyDataContext db = new
MyDataContext
(DBHelper.GetConnectionString())) {
var query = from pro in db.Projects
select new Project(){ ProjectName =
pro.ProjectName, ProjectId =
pro.ProjectId }; return query.ToList(); }
} – Sami Jun 16 '11 at 10:12
```

5 It works but its incorrect. You're promising the whole object to the callee but you're only giving partial

care but you're only giving partial information in it. Only the implementation of the method reveals the truth which is bad design. – [Muhammad Hasan Khan](#) Jun 16 '11 at 10:15

Thanks i have got your point. – [Sami](#) Jun 16 '11 at 10:21

7

```

public List<Object> GetProjectForCombi
{
    using (MyDataContext db = new MyData
    {
        var query = db.Project
        .Select<IEnumerable<something>,Pr
        return new ProjectIni

    return query.ToList<Object>();
    }
}

```

answered Dec 16 '13 at 23:23



[John Peters](#)

3,801 2 28 47

query.ToList<Object>(); worked for me for list of collections from linq query. This cast whole objects from select new {}. Thank you. – [daremachine](#) Oct 1 '14 at 16:08

5

You cannot return anonymous types from a class... (Well, you can, but you have to cast them to object first and then use reflection at the other side to get the data out again) so you have to create a small class for the data to be contained within.

```

class ProjectNameAndId
{
    public string Name { get; set; }
    public string Id { get; set; }
}

```

Then in your LINQ statement:

```

select new ProjectNameAndId { Name = r

```

answered Jun 16 '11 at 10:00



[Colin Mackay](#)

14.6k 3 49 73

1 Just a quick note ProjectNameAndId

is not a good class name :) –

[Kieren Johnstone](#) Jun 16 '11 at 10:02

Yeah.... I just wrote it quickly. I wasn't thinking too hard about a good class name. – [Colin Mackay](#) Jun 16 '11 at 10:03

- 2 I know it's nitpicking but SO questions have a tendency to become valuable googlable resources, and figured the world would benefit from my little comment :) – [Kieren Johnstone](#) Jun 16 '11 at 10:06

3

```
public List<Object> GetProjectForCombi
{
    using (MyDataContext db = new MyDc
    {
        var query = from pro in db.Pr
        select new {pro.I

        return query.ToList<Object>();
    }
}
```

answered May 14 '13 at 20:18



[user2383325](#)

31 1

You can also use nameless class and in this case will return type Object. –

[user2383325](#) May 14 '13 at 20:19

1 Your method's return value has to be a List<Project> .

Using select new you are creating an instance of an anonymous type, instead of a Project .

answered Jun 16 '11 at 9:59



[Groo](#)

35.8k 14 88 159


May I ask why we should use select new to create an instance ? – [CYB](#) Feb 23 '14 at 9:17


@CYB: select new is used when

you want your query to create new instances of a certain class, instead of simply taking source items. It allows you to create instances of a completely different class, or even an anonymous class like in OP's case

anonymous class like in OP's case.

For a LINQ provider like LINQ-to-Entities, projecting to a new instance of an anonymous class means that it can issue an SQL query which only fetches columns which are used inside the select statement, instead of fetching all columns. – [Groo](#) Feb 24 '14 at 10:56

@CYB: in OP's example, first snippet will create an SQL query similar to `SELECT [ProjectName],[ProjectId] FROM [Projects]`, while the second one needs to construct an entire `Project` instance and will use something like `SELECT * FROM [Projects]`. If you don't need all properties from a given query, you can save a bunch of DB operations by selecting only the properties you need. The only problem with OP's question is that you cannot return an anonymous class from a method, but instead have to create a new one (e.g. `ProjectInfo` [like suggested](#)). – [Groo](#) Feb 24 '14 at 11:01 

Thanks a lot. I've understood it. – [CYB](#) Feb 25 '14 at 5:44 

▲
1 What is being returned is an anonymous type so create a new class with 2 fields

▼

```
class BasicProjectInfo {
    string name;
    string id;
}
```

and return `new BasicProjectInfo(pro.ProjectName, pro.ProjectId);`. Your method in this case will return a `List<BasicProjectInfo>`

answered Jun 16 '11 at 9:57



[sh_kamalh](#)

3,306 3 32 49

..and just a quick note that won't compile, there's no constructor defined there. The better option would be like Colin Mackay's: `new BasicProjectInfo { name = pro.ProjectName etc` – [Kieren Johnstone](#) Jun 16 '11 at 10:03

try this solution for me its working

0

```

public List<ProjectInfo> GetProjec
{
    using (MyDataContext db = new MyD:
        (DBHelper.GetConnectionString()))
    {
        return (from pro in db.Projec
            select new { query
        }
    }
}

```

answered Jul 19 '17 at 17:24



Martin Chinome

294 2 11

▲

You can do it as following:

0

```

class ProjectInfo
{
    public string Name {get; set; }
    public long Id {get; set; }

    ProjectInfo(string n, long id)
    {
        name = n;    Id = id;
    }
}

public List<ProjectInfo> GetProjectFor
{
    using (MyDataContext db = new MyD:
    {
        var query = from pro in db.Pr
            select new Project

        return query.ToList<ProjectIr
    }
}

```

answered Mar 20 '13 at 19:44



Husam Hilal

101 1 1

This won't work since only parameterless constructors and initializers are supported by linq to entities (e.g see stackoverflow.com/questions/3571084/...) – Yaur May 14 '13 at 21:10

▲

-1

You're creating a new type of object therefore it's anonymous. You can return a dynamic.

▼

```

public dynamic GetProjectForCombo()
{
    using (MyDataContext db = new MyD:
    {
        var query = from pro in db.Pr

```

```
return query.ToList();
```

 nib_alejo
1