The instance of entity type 'BookLoan' cannot be tracked

Asked 2 years, 10 months ago Active 2 years, 10 months ago Viewed 10k times

6

1

I'm trying to update an entity and I've run into the following error:

> InvalidOperationException: The instance of entity type 'BookLoan' cannot be tracked because another instance of this type with the same key is already being tracked. When adding new entities, for most key types a unique temporary key value will be created if no key is set (i.e. if the key property is assigned the default value for its type). If you are explicitly setting key values for new entities, ensure they do not collide with existing entities or temporary values generated for other new entities. When attaching existing entities, ensure that only one entity instance with a given key value is attached to the context.

I've done a little research and from what I can tell I'm apparently trying to track an already tracked entity when I use `_context.Update(bookloan);` but I'm not really sure what to do.

What I'm trying to do is update an existing entity/record in my database. Here are the get and post controllers as I'm not sure what else to share.

Get

```
[HttpGet]
public async Task<IActionResult> Return(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    if (isBookCheckedOut(id) == false)
    {
        //Not checked out
        return RedirectToAction("Index");
    }
    else
    {
        var bookloan = (from book in _context.Books.Where(b => b.BookId == id)
                        join loan in _context.BookLoans.Where(x =>
```

```
                        {
                            BookLoanID = loanWithDefault.BookLoanID,
                            BookID = book.BookId,
                            Title = book.Title,
                            StudentID = loanWithDefault == null ? null :
loanWithDefault.StudentID,
                            StudentFristName = loanWithDefault == null ? null :
loanWithDefault.Student.FirstName,
                            StudentLastName = loanWithDefault == null ? null :
loanWithDefault.Student.LastName,
                            //Fines
                            CheckedOutOn = loanWithDefault == null ? (DateTime?)null :
loanWithDefault.CheckedOutOn,
                            IsAvailable = loanWithDefault == null,
                            AvailableOn = loanWithDefault == null ? (DateTime?)null :
loanWithDefault.DueOn
                        }).FirstOrDefault();

            if (bookloan == null)
            {
                return NotFound();
            }

            return View(bookloan);
        }
    }
```

Post:

```
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Return(BookReturnViewModel model)
    {


        if (ModelState.IsValid && isBookCheckedOut(1) == true)
        {
            var bookloan = new BookLoan()
            {
                BookLoanID = model.BookLoanID,
                BookID = model.BookID,
                StudentID = model.StudentID,
                CheckedOutOn = (DateTime)model.CheckedOutOn,
                DueOn = (DateTime)model.AvailableOn,
```

```
            try
            {

                _context.Update(bookloan);
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {

            }

            return RedirectToAction("Index");
        }
        else
        {

        }

        return View();
    }
```

c# asp.net-mvc entity-framework asp.net-core entity

asked Oct 23 '16 at 0:33

Christopher Johnson
147 1 2 12

1 Instead or 'creating' a new `BookLoan` , get the existing entity from the database based on the ID (e.g. `BookLoan data = db.BookLoans.Where(x => x.BookLoanID == model.BookLoanID).FirstOrDefault(); `) and update its properties based on the view model and then save it. – user3559349 Oct 23 '16 at 1:09

Thanks, that worked perfectly! I'd mark that as the answer but I don't think I can do that for comments. The way I was trying to do it was just me mimicking what I've seen Entity do when an edit with scaffolding is made. – Christopher Johnson Oct 23 '16 at 1:39

1 Answer

Your context already includes the entity, so rather that creating a new one, get the existing one based on the ID of the entity and update

13

```
if (ModelState.IsValid && isBookCheckedOut(1) == true)
{
    // Get the existing entity
    BookLoan bookLoan = db.BookLoans.Where(x => x.BookLoanID ==
model.BookLoanID).FirstOrDefault();
    if (bookLoan != null)
    {
        bookLoan.BookID = model.BookID;
        bookLoan.StudentID = model.StudentID;
        .... // update other properties as required
        _context.Update(bookloan);
        await _context.SaveChangesAsync();
        return RedirectToAction("Index");
    }
    ....
```

Side note: when returning the view, its good practice to pass back the model using `return View(model);` - your form controls will be correctly populated even if you don't (because they take the values from `ModelState`), but if you have any references to the model properties (e.g. `<div>@Model.someProperty</div>`) it would throw an exception.

answered Oct 23 '16 at 2:02
user3559349

Thanks for such a good answer – NullHypothesis Oct 22 '18 at 18:44

1 It would *potentially* be faster to do db.BookLoans.Find(model.BookLoanID) because it will try to retrieve it from the context first before going to the db – sdrevk Mar 12 at 16:58