Nested population

Working of nested population

```
Post.find({})
      .populate('user')
      .populate({
          path: 'comments',
          populate: {
              path: 'user'
          }
      })
      .exec(function(err, posts){
        return res.render('home', {
            title: "Codeial | home",
            posts: posts
        });
      });
```

1. *Post.find({})* will fetch all entries from the posts database.

2. Further, *.populate('user')* will find the 'user' attribute of the fetched posts.

   This attribute has a value ObjectID which refers to the user's id who created the post.

   The populate function will replace this ObjectID with the object itself, i.e. the user id with the user object itself (we are doing this so that we can access other user properties like name, email, etc in the views).

3. Further, *.populate({*
   path: 'comments',
   populate: { path: 'user' }
   })

   Here, the *path:* specifies the attribute that needs to be populated, and *populate:* specifies the further population of the attribute.

First, the ‘comments’ attribute of the fetched posts will be found. It is an array of values which are ObjectIDs of all the users that commented on that post.

But we want to know the names of the users that commented on the post. That’s why we further populate this ‘comments’ attribute to find the user object which is found using the user ids in this comments array. It is done using: *populate: { path: ‘user’ }*

4. Finally, the result of the database query will be all the entries from the posts database, with
   - attribute value of ‘user’ = the user object itself (coming from populating ‘user’) and
   - attribute value of ‘comments’ = an array of user objects that commented on the post (coming from populating ‘comments’ and further populating the ids in the arrays)

In conclusion, nested population is used if a further population of an attribute is required.

Additional (In case you face any errors/bugs):

In the schema, if you define the ref value NOT same as the schema name to which we need to refer, then while populating the DB we’ve to EXPLICITLY add a ‘model’ field with a value equal to the schema name. Otherwise, it’ll not know the value of ‘ref’ added in the schema refers to exactly which model.

That’s why you may face errors/bugs if you give incorrect ref value for eg miss writing in a capital case i.e. write ‘user’ instead of ‘User’, or ‘comment’ instead of ‘Comment’, etc). So re-check your schema!

```
Post.find({})
      .populate({
          path: 'user',
          model: 'User'
      })
      .populate({
          path: 'comments',
          model: 'Comment',
          populate: {
              path: 'user',
              model: 'User'
          }
      })
      .exec(function(err, posts){
        return res.render('home', {
            title: "Codeial | home",
            posts: posts
        });
    });
```

Note:

Here,

.populate('user') and

.populate({
path: 'user',
model: 'User'
})

Are two different ways of writing the same thing.