Contents

[Explaining how a table’s data and its related tables’ data is returned using EF Core and LINQ 2](#_Toc202733402)

[eagerly load related data 2](#_Toc202733403)

[.Include(rb => rb.Resource) 2](#_Toc202733404)

[APPENDIX 4](#_Toc202733405)

[“Eager Loading” General Definition 4](#_Toc202733406)

[**Eager loading** refers to the strategy of **loading all necessary data up front**, rather than waiting until it's needed (which would be **lazy loading**). 4](#_Toc202733407)

[“Eagerly Load” Definition re. EF Core 5](#_Toc202733408)

[Entity Framework Core tells EF Core to include associated entities when it retrieves data from the database. 5](#_Toc202733409)

# Explaining how a table’s data and its related tables’ data is returned using EF Core and LINQ

[Skip to content](https://chatgpt.com/#main)

**You said:**

the following Entity Framework Core code allows a related table's data to be included in a list of data. Can you explain specifically what it does: .Include(rb => rb.Resource) and how it serves to get data from the Resource table from rb (ResourceBooks)

**ChatGPT said:**

Yes, I can explain that! Let's break it down step by step:

**Context**

In Entity Framework Core (EF Core), .Include(...) is used to

eagerly load related data. That means it tells EF Core to include associated entities when it retrieves data from the database.

So when you write:

### .Include(rb => rb.Resource)

You're telling EF Core:

"When retrieving ResourceBooks, also retrieve the related Resource entity for each one."

**Assuming the following:**

Let’s assume you have these two entity classes:

public class ResourceBook

{

public int Id { get; set; }

public int ResourceId { get; set; } // Foreign key

public Resource Resource { get; set; } // Navigation property

}

public class Resource

{

public int Id { get; set; }

public string Name { get; set; }

}

Here:

* ResourceBook has a navigation property called Resource.
* That means each ResourceBook is **linked to a single Resource** (via ResourceId foreign key).

**What .Include(rb => rb.Resource) Does**

When you query ResourceBooks, e.g.:

var books = context.ResourceBooks

.Include(rb => rb.Resource)

.ToList();

EF Core will:

1. Execute a **JOIN** behind the scenes between ResourceBooks and Resources.
2. Load both the ResourceBook and its related Resource.
3. Populate the Resource navigation property for each ResourceBook.

Without .Include(...), EF Core will still load the ResourceBook records, but the Resource navigation property will be null unless you explicitly load it later (lazy loading or manual querying).

**Summary**

✅ .Include(rb => rb.Resource):

* Tells EF Core to eagerly load the Resource entity that is related to each ResourceBook.
* Helps avoid multiple round-trips to the database (which would happen with lazy loading).
* Ensures the Resource navigation property is fully populated.

Let me know if you want to see the generated SQL or a deeper example!

By messaging ChatGPT, you agree to our [Terms](https://openai.com/terms) and have read our [Privacy Policy](https://openai.com/privacy). See Cookie Preferences.  
Don't share sensitive info. Chats may be reviewed and used to train our models. [Learn more](https://help.openai.com/en/articles/5722486-how-your-data-is-used-to-improve-model-performance)

Top of Form

Bottom of Form

ChatGPT can make mistakes. Check important info. See Cookie Preferences.

**We use cookies**

Some cookies are essential for this site to function and cannot be turned off. Others are set to help us understand how our service performs and is used, and to support our marketing efforts. Learn more in our [Cookie Policy](https://openai.com/policies/cookie-policy/). For more info or to customize your preferences, tap ‘Manage Cookies’.

# APPENDIX

## “Eager Loading” General Definition

**eager loading** is a general concept in computer science and software development—not just something specific to Entity Framework Core (EF Core).

**General Meaning of Eager Loading**

### **Eager loading** refers to the strategy of **loading all necessary data up front**, rather than waiting until it's needed (which would be **lazy loading**).

**Examples Outside EF Core**

**1. Programming Languages (e.g., Java, Python, C#)**

* **Eager evaluation**: Variables or expressions are computed immediately.
* **Lazy evaluation**: Values are only computed when accessed.

**2. Web Development**

* Eager loading of assets (e.g., images, scripts): All resources are loaded right away when the page loads.
* Lazy loading: Resources are loaded as needed (e.g., when the user scrolls to a certain section).

**3. ORMs Beyond EF Core**

* Other Object-Relational Mappers like Hibernate (Java), SQLAlchemy (Python), or ActiveRecord (Ruby on Rails) also use **eager loading** and **lazy loading** strategies to fetch related database data.

**4. Caching and Data Processing**

* Eager loading in caching means pulling and storing data **immediately**, anticipating future use.
* Lazy caching would mean fetching and caching only when a request for the data is made.

**In Summary**

✅ **Eager loading** is a **general design pattern** that means:

"Load everything you might need **now**, rather than waiting."

While EF Core uses the term frequently in the context of querying related data, the idea applies broadly to software design, performance tuning, and memory management.

Let me know if you want a visual comparison between eager and lazy loading in different domains!

Get smarter responses, upload files and images, and more.

Top of Form

Bottom of Form

Top of Form

Bottom of Form

ChatGPT can make mistakes. Check important info. See Cookie Preferences.

**We use cookies**

Some cookies are essential for this site to function and cannot be turned off. Others are set to help us understand how our service performs and is used, and to support our marketing efforts. Learn more in our [Cookie Policy](https://openai.com/policies/cookie-policy/). For more info or to customize your preferences, tap ‘Manage Cookies’.

## “Eagerly Load” Definition re. EF Core

[Skip to content](https://chatgpt.com/#main)

**You said:**

the following Entity Framework Core code allows a related table's data to be included in a list of data. Can you explain specifically what it does: .Include(rb => rb.Resource) and how it serves to get data from the Resource table from rb (ResourceBooks)

**ChatGPT said:**

Yes, I can explain that! Let's break it down step by step:

**Context**

In Entity Framework Core (EF Core), .Include(...) is used to **eagerly load related data**.

### Entity Framework Core tells EF Core to include associated entities when it retrieves data from the database.

So when you write:

.Include(rb => rb.Resource)

You're telling EF Core:

"When retrieving ResourceBooks, also retrieve the related Resource entity for each one."

**Assuming the following:**

Let’s assume you have these two entity classes:

public class ResourceBook

{

public int Id { get; set; }

public int ResourceId { get; set; } // Foreign key

public Resource Resource { get; set; } // Navigation property

}

public class Resource

{

public int Id { get; set; }

public string Name { get; set; }

}

Here:

* ResourceBook has a navigation property called Resource.
* That means each ResourceBook is **linked to a single Resource** (via ResourceId foreign key).

**What .Include(rb => rb.Resource) Does**

When you query ResourceBooks, e.g.:

var books = context.ResourceBooks

.Include(rb => rb.Resource)

.ToList();

EF Core will:

1. Execute a **JOIN** behind the scenes between ResourceBooks and Resources.
2. Load both the ResourceBook and its related Resource.
3. Populate the Resource navigation property for each ResourceBook.

Without .Include(...), EF Core will still load the ResourceBook records, but the Resource navigation property will be null unless you explicitly load it later (lazy loading or manual querying).

**Summary**

✅ .Include(rb => rb.Resource):

* Tells EF Core to eagerly load the Resource entity that is related to each ResourceBook.
* Helps avoid multiple round-trips to the database (which would happen with lazy loading).
* Ensures the Resource navigation property is fully populated.

Let me know if you want to see the generated SQL or a deeper example!

**You said:**

what does eagerly load mean?

**ChatGPT said:**

Great question!

**What “Eagerly Load” Means**

**Eager loading** means that **related data is retrieved from the database at the same time as the main entity**, as part of the initial query.

So, when you eagerly load data using .Include(...), Entity Framework Core:

* Generates a single SQL query (typically using JOINs).
* Fetches the main entity **and** its related entities **together**.
* Fills in the navigation properties automatically.

**Example:**

Suppose you have a Book and each Book has an Author.

var books = context.Books

.Include(b => b.Author)

.ToList();

**Without eager loading**:

* Only the Books are retrieved.
* The Author property on each book will be null or require a separate query later (lazy loading or manual load).

**With eager loading**:

* EF Core does something like:

SELECT b.\*, a.\*

FROM Books b

LEFT JOIN Authors a ON b.AuthorId = a.Id

* It gets both the book and its author in one trip to the database.

**Why It Matters**

**Pros of Eager Loading:**

* Fewer queries = better performance in many cases.
* Related data is ready to use immediately.

**Cons:**

* The query is more complex.
* You might load more data than you need if you're not careful.

**Other Loading Strategies**

* **Lazy loading**: EF loads related data **only when you access the navigation property**. It issues additional SQL queries on demand.
* **Explicit loading**: You manually tell EF to load related data with something like context.Entry(book).Reference(b => b.Author).Load();.

Let me know if you want a diagram or code example comparing eager, lazy, and explicit loading!

Top of Form

Bottom of Form

Top of Form

Bottom of Form

ChatGPT can make mistakes. Check important info. See Cookie Preferences.

**We use cookies**

Some cookies are essential for this site to function and cannot be turned off. Others are set to help us understand how our service performs and is used, and to support our marketing efforts. Learn more in our [Cookie Policy](https://openai.com/policies/cookie-policy/). For more info or to customize your preferences, tap ‘Manage Cookies’.