

# What's an Aggregate Root?

Asked 10 years, 3 months ago   Active 2 years, 6 months ago   Viewed 107k times



439



I'm trying to get my head around how to properly use the repository pattern. The central concept of an Aggregate Root keeps coming up. When searching both the web and Stack Overflow for help with what an aggregate root is, I keep finding discussions about them and dead links to pages that are supposed to contain base definitions.

In the context of the repository pattern, **what is an aggregate root?**



200



design-patterns

repository-pattern

ddd-repositories

aggregateroot

asked Dec 24 '09 at 15:02



Dinah

46k ● 28 ● 122 ● 145

14 Consider reviewing the following case studies. Effective Aggregate Design Part I: Modeling a Single Aggregate [dddcommunity.org/wp-content/uploads/files/pdf\\_articles/...](http://dddcommunity.org/wp-content/uploads/files/pdf_articles/...) Part II: Making Aggregates Work Together [dddcommunity.org/wp-content/uploads/files/pdf\\_articles/...](http://dddcommunity.org/wp-content/uploads/files/pdf_articles/...) Part III: Gaining Insight Through Discovery [dddcommunity.org/wp-content/uploads/files/pdf\\_articles/...](http://dddcommunity.org/wp-content/uploads/files/pdf_articles/...) – Ben Vitale Feb 9 '12 at 20:50

## 10 Answers

Active

Oldest

Votes



305



In the context of the repository pattern, aggregate roots are the only objects your client code loads from the repository.

The repository encapsulates access to child objects - from a caller's perspective it automatically loads them, either at the same time the root is loaded or when they're actually needed (as with lazy loading).

For example, you might have an `Order` object which encapsulates operations on multiple `LineItem` objects. Your client code would never load the `LineItem` objects directly, just the `Order` that contains them, which would be the aggregate root for that part of your domain.

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email



Sign up with Google



Sign up with GitHub



Sign up with Facebook



- 21 Hypothetically, if the client code needed the `LineItem` for some other purpose, would that form a separate aggregate (assuming there would be other objects involved not related to the `Order` object)? – [Ahmad](#) Mar 8 '10 at 19:40
- 19 @Ahmad, other aggregates might refer to `LineItems` as read-only data, they just can't *change* them. If other aggregates could change them, you couldn't protect the order's invariants (nor the line items'). – [Jeff Sternal](#) Mar 8 '10 at 19:57
- 3 Take a look at this eg [lostechies.com/blogs/jimmy\\_bogard/archive/2010/02/23/...](http://lostechies.com/blogs/jimmy_bogard/archive/2010/02/23/...) . In the example, `Customer` is an invariant of `Order`, right? However, `Customer` can also be the another aggregate root? Or am I missing some fundamental understanding here? – [Ahmad](#) Mar 8 '10 at 20:04
- 3 @Jeff You said "they just can't change them" - is that enforceable, or a matter of convention? – [Neil Barnwell](#) Feb 3 '11 at 23:52
- 4 @Neil: I'd enforce it using whatever language mechanisms are available - for instance, by creating an immutable class to represent the data. – [Jeff Sternal](#) Feb 4 '11 at 14:02

From Evans DDD:

202

An AGGREGATE is a cluster of associated objects that we treat as a unit for the purpose of data changes. Each AGGREGATE has a root and a boundary. The boundary defines what is inside the AGGREGATE. The root is a single, specific ENTITY contained in the AGGREGATE.



And:

The root is the only member of the AGGREGATE that outside objects are allowed to hold references to[.]

This means that aggregate roots are the only objects that can be loaded from a repository.

An example is a model containing a `Customer` entity and an `Address` entity. We would never access an `Address` entity directly from the model as it does not make sense without the context of an associated `Customer` . So we could say that `Customer` and `Address` together form an aggregate and that `Customer` is an aggregate root.

answered Dec 24 '09 at 15:44



jason

209k ● 30 ● 386 ● 501

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email



Sign up with Google



Sign up with GitHub



Sign up with Facebook



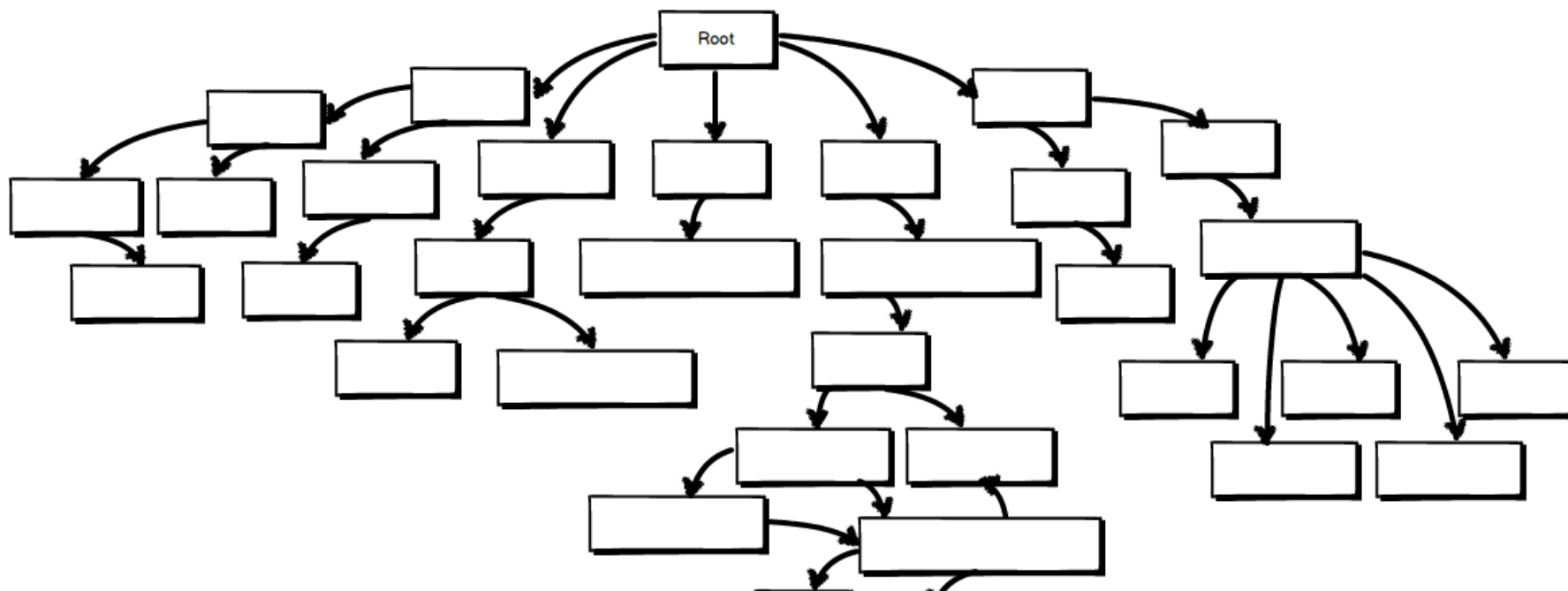
- 3 So the verbiage is forever confusing me. Each AGGREGATE has a root and The root is the only \*member\* of the AGGREGATE -- this verbiage implies that the root is property on the Aggregate. But in all of the examples, it's the other way around: the root contains properties that are aggregates. Can you clarify? – [Sinaesthetic](#) Oct 11 '15 at 7:32
- 1 Just to get my language right, is the `Customer` class considered the aggregate root, or `Customer` instances? – [Joe](#) Nov 17 '16 at 16:21
- 1 Generally speaking in the Customer-order-line-item paradigm the Customer would be the Aggregate Root. The Instance of a customer would be an instance of that Aggregate Root. When speaking of an Aggregate Root called Customer you are discussing the logical construction of a Customer that makes up the instance of a customer. A collection of Customers is just a collection. – [Ibrahim Malluf](#) Dec 5 '16 at 3:51

Aggregate root is a complex name for simple idea.

105

## General idea

Well designed class diagram encapsulates its internals. Point through which you access this structure is called `aggregate root`.

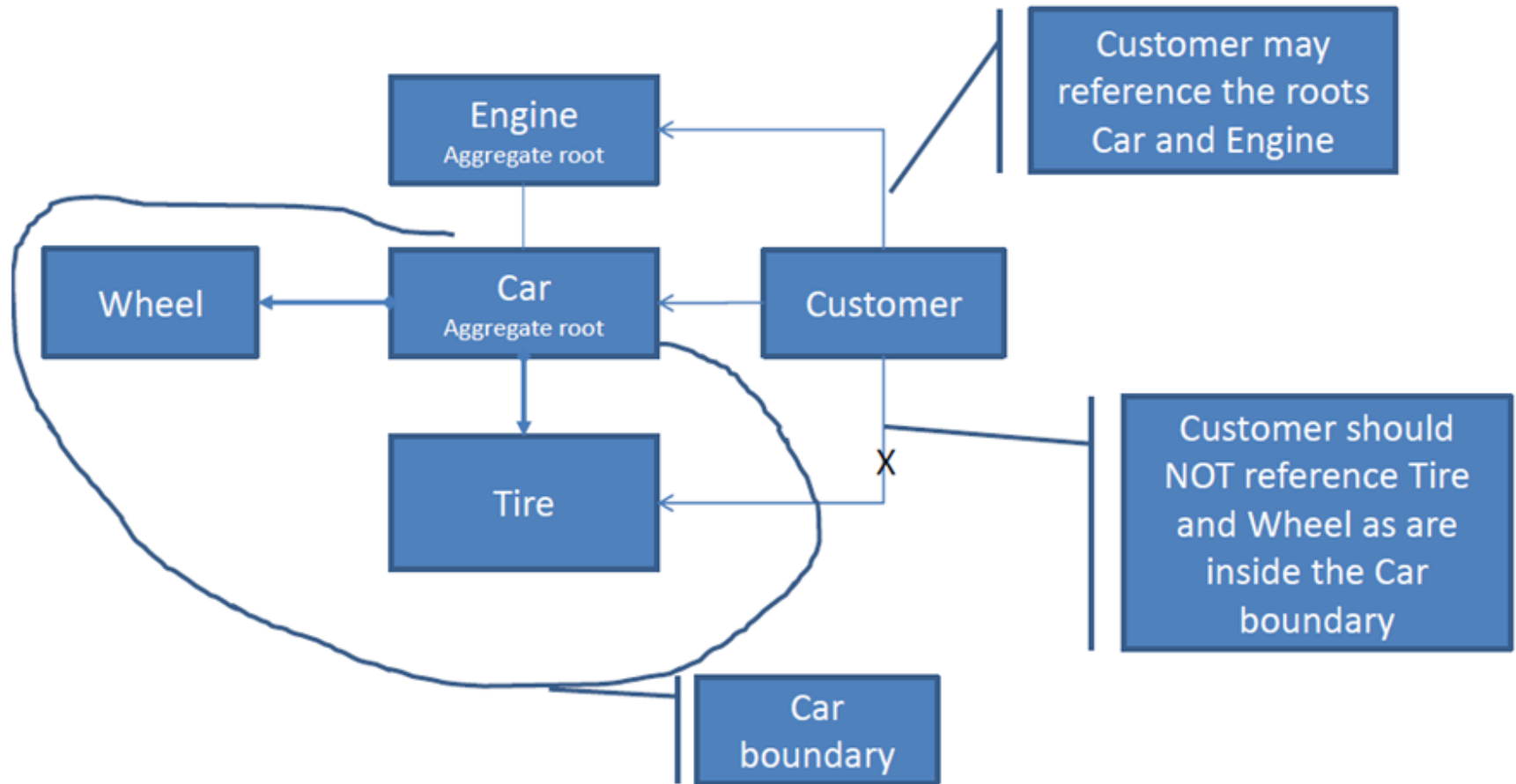


Join Stack Overflow to learn, share knowledge, and build your career.

[Sign up with email](#)[Sign up with Google](#)[Sign up with GitHub](#)[Sign up with Facebook](#)

## Example

Check this simple class hierarchy



How do you want to ride your car? Chose better api


Option A (it just somehow works):

```
car.ride();
```

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email

 Sign up with Google

 Sign up with GitHub

 Sign up with Facebook



```

if(car.getTires().getUsageLevel() < Car.ACCEPTABLE_TIRE_USAGE)
    for (Wheel w: car.getWheels()){
        w.spin();
    }
}

```

If you think that option A is better then congratulations. You get main reason behind `aggregate root`.

Aggregate root encapsulates multiple classes. you can manipulate whole hierarchy only through main object.

answered Sep 25 '15 at 12:34



**Marcin Szymczak**

8,477 ● 4 ● 41 ● 53

17 I like the example, but I'm struggling to find a scenario in which Customer should reference Engine. It seems like Engine should be encapsulated behind Car. Can you elaborate on this a little? – [emragins](#) Oct 29 '15 at 4:02

In my opinion the engine itself must be inside a car specific model, e.g a BMW series 5 with 3000cc engine. With this modeling the engine is a component for a car. – [Parama Dharmika](#) Oct 15 '16 at 19:45 ✎

1 @ParamaDharmika sure, you can model it that way. That depends on how 'advanced' with cars is your customers. In basic model he should have access to `car` aggregate root. You can also allow situation like one on the drawing. Correct solution depend on business model of application. It might be different in each case. – [Marcin Szymczak](#) Oct 16 '16 at 22:21

@MarcinSzymczak correct, couldn't agree more that solution is depend on the domain model itself – [Parama Dharmika](#) Oct 18 '16 at 9:34

Actually, the Wheel is an aggregate that contains Tire (and other parts). If your rules require that the Wheel aggregate can only be accessed through the Car Root-Aggregate, then the engine is also contained within the Car Root Aggregate and should not be accessed outside of the Car. That's in the realm of a Car instance. A car owner (Customer) would not reference an engine except in the context of his/her car. – [Ibrahim Malluf](#) Dec 5 '16 at 4:04

Imagine you have a Computer entity, this entity also cannot live without its Software entity and Hardware entity. These form the `Computer` aggregate, the mini-ecosystem for the Computer portion of the domain.

35

Aggregate Root is the mothership entity inside the aggregate (in our case `Computer`). it is a common practice to have your repository

**Join Stack Overflow** to learn, share knowledge, and build your career.

Sign up with email

Sign up with Google

Sign up with GitHub

Sign up with Facebook



```
public class Computer : IEntity, IAggregateRoot
{
    public Hardware Hardware { get; set; }
    public Software Software { get; set; }
}

public class Hardware : IEntity { }
public class Software : IValueObject { }

public class Repository<T> : IRepository<T> where T : IAggregateRoot { }
```

Keep in mind that Hardware would likely be a ValueObject too (do not have identity on its own), consider it as an example only.

edited Aug 23 '10 at 17:24



Dinah

46k ● 28 ● 122 ● 145

answered Dec 24 '09 at 15:15



Francisco Aquino

8,750 ● 1 ● 25 ● 37

5 where T : IAggregateRoot - This one made my day – [Cristian E.](#) Apr 14 '15 at 17:08

The wording is a little contradictory, I think and this is what confuses me when trying to learn this. You're saying that the Computer is the aggregate, but then you're saying that the root would be the mothership entity INSIDE the aggregate. So which one is the "mothership" entity inside the aggregate in this example? – [Sinaesthetic](#) Mar 18 '16 at 2:11

Greetings from the future!. What the guy means is that the Computer by itself is the aggregate root, while the computer AND everything inside it is the aggregate. Or more clearly: the case by itself is the aggregate root, while the whole computer is the aggregate (the collection of everything that makes up the "computer, e.g. RGB lighting, Hardware, Power Supply, OS, etc). – [Captain Kenpachi](#) Jan 7 at 13:52 ✎

▲ If you follow a database-first approach, you aggregate root is usually the table on the 1 side of a 1-many relationship.

16

The most common example being a Person. Each person has many addresses, one or more pay slips, invoices, CRM entries, etc. It's not always the case, but 9/10 times it is.



We're currently working on an e-commerce platform, and we basically have two aggregate roots:



1. Customers

**Join Stack Overflow** to learn, share knowledge, and build your career.

Sign up with email



Sign up with Google



Sign up with GitHub



Sign up with Facebook



These are taken care of by the Customer and Seller repository respectively.

answered Sep 10 '13 at 8:15



[Captain Kenpachi](#)

5,500 ● 5 ● 39 ● 57

- 
- 8 If you follow a database-first approach then you are not practicing Domain Driven Design, you are following Data Driven Design. – [Sinaesthetic](#) Dec 27 '17 at 18:00
- 
- 5 It's a Q&A forum where people come to solve problems and/or learn -- That wasn't me poking at you. By definition, DDD is a mindset more than anything else and it is confusing for many, so this was me making sure the comment was made for those that are learning DDD in effort to help mitigate any potential conflation of design methodologies. – [Sinaesthetic](#) Dec 28 '17 at 21:27 ✎
- 

▲ Dinah:


12


▼ In the Context of a Repository the Aggregate Root is an Entity with no parent Entity. It contains zero, One or Many Child Entities whose existence is dependent upon the Parent for it's identity. That's a One To Many relationship in a Repository. Those Child Entities are plain Aggregates.



**Join Stack Overflow** to learn, share knowledge, and build your career.

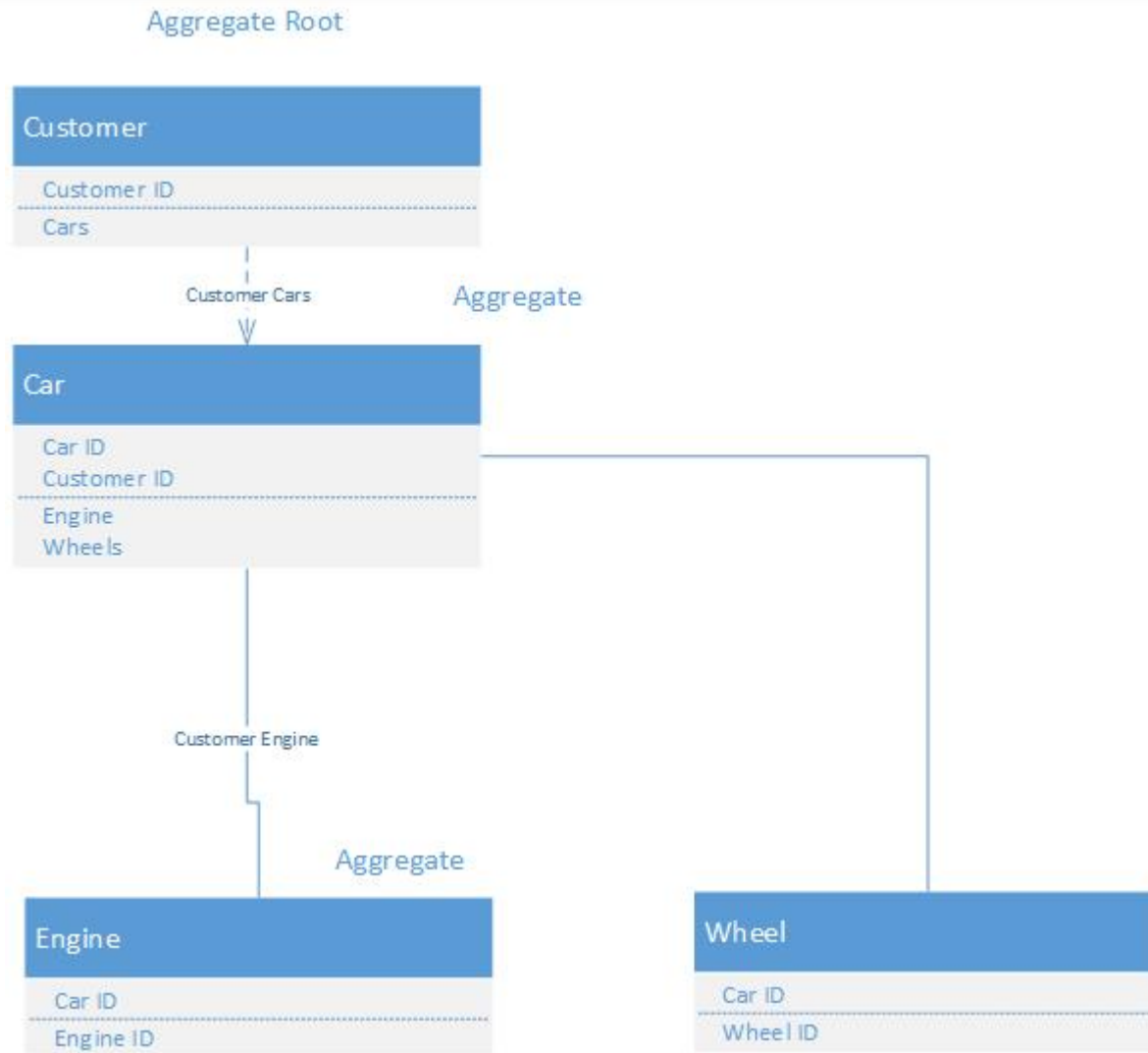
Sign up with email

 Sign up with Google

 Sign up with GitHub

 Sign up with Facebook





answered Dec 5 '16 at 4:45

**Ibrahim Malluf**

559 ● 3 ● 6

**Join Stack Overflow** to learn, share knowledge, and build your career.

Sign up with email



Sign up with Google




Sign up with GitHub



Sign up with Facebook





- 3 Child entities are not aggregates, they are just entities that happen to be members of the aggregate in which the aggregate root controls. An "aggregate" is a *logical* grouping of entities. – [Sinaesthetic](#) Dec 27 '17 at 17:56 

@JorgeeFG it really depends on the bounded context you are designing. If you are a car seller, then something like a Carshop becomes the aggregate root, and below it follows the Cars... – [jokab](#) Jul 24 '19 at 1:25

From a [broken link](#):

8

Within an Aggregate there is an Aggregate Root. The Aggregate Root is the parent Entity to all other Entities and Value Objects within the Aggregate.

A Repository operates upon an Aggregate Root.



More info can also be found [here](#).

answered Dec 24 '09 at 15:12



[Otávio Décio](#)

67.6k ● 14 ● 150 ● 217

- 4 Thank you. That is definitely the most common and frustrating broken link I continually ran across. – [Dinah](#) Dec 24 '09 at 15:14

Also, the wording seems backwards. How can the root be *within* the aggregate and be it's parent at the same time? – [Sinaesthetic](#) Oct 11 '15 at 7:34

- 1 The Aggregate Root is the root class. A plain Aggregate is always contained within an Aggregate Root. Using the Diagram posed above...The Customer is the Aggregate-Root. The Customer can own one or more cars. Cars are Aggregates in relation to the Customer. Cars have an Engine. The Engine is an Aggregate contained in the Car Aggregate. What makes the Customer an Aggregate Root is the model's assumption that access to a car or it's components are always through the customer who owns the car. – [Ibrahim Malluf](#) Dec 5 '16 at 4:11

**Aggregate** means collection of something.

**root** is like top node of tree, from where we can access everything like `<html>` node in web page document.

7

Blog Analogy, A user can have many posts and each post can have many comments. so if we fetch any user then it can act as **root** to access all the related posts and further comments of those posts. These are all together said to be collection or **Aggregated**

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email



Sign up with Google



Sign up with GitHub



Sign up with Facebook





Aggregate is where you protect your invariants and force consistency by limiting its access through aggregate root. Do not forget, aggregate should design upon your project business rules and invariants, not database relationship. you should not inject any repository and no queries are not allowed.



answered Sep 16 '17 at 9:02



Alireza Rahmani Khalili

1,041 ● 2 ● 16 ● 23



-1



In Erlang there is no need to differentiate between aggregates, once the aggregate is composed by data structures inside the state, instead of OO composition. See an example: <https://github.com/bryanhunter/cqrs-with-erlang/tree/ndc-london>

answered Mar 9 '16 at 12:27



Henry H.

759 ● 7 ● 10

Join Stack Overflow to learn, share knowledge, and build your career.

[Sign up with email](#)[Sign up with Google](#)[Sign up with GitHub](#)[Sign up with Facebook](#)