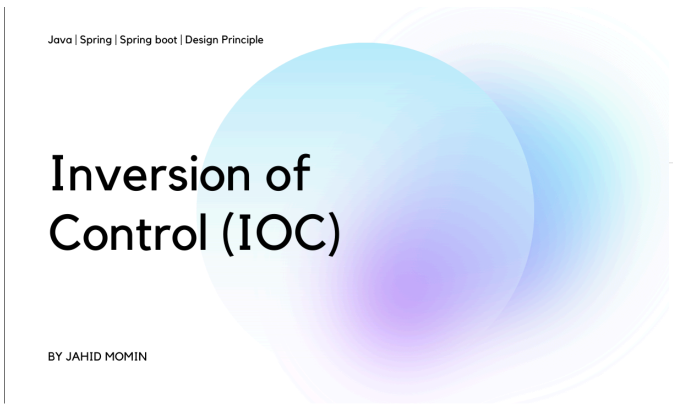

Java | Spring | Spring boot | Design Principle
Inversion of Control (IOC)
BY JAHID MOMIN

Love ❤️ Canva

Inversion of Control (IoC) Design principle .

Jahid Momin
Team Lead I Serverless | ExpressJS | SpringBoot | Driving Innovation in Backend Development |...
6 articles + Follow

January 14, 2023

Open Immersive Reader

Inversion of Control (IoC) is a design principle in which a software component is designed to receive its dependencies from an external source, rather than creating them itself. This is in contrast to traditional software design, where a component is responsible for creating and managing its own dependencies.

There are two types of IoC: type 1 (interface-based) and type 2 (template-based).

Type 1, also known as "Hollywood Principle 😎" states "Don't call us, we'll call you" meaning, the system will call the dependent objects when it needs them, rather than the dependent objects calling the system when they need something.

Type 2, also called "template method" pattern, defines the skeleton of an algorithm in a base class, and allows derived classes to fill in specific details.

Dependency injection (DI) is a design pattern that allows a programmer to remove hard-coded dependencies and make it possible to change them, whether at run-time or compile-time. It is a technique that allows an object to supply its dependencies, instead of having them hard-coded or created within the object itself. This means that an object does not create its dependencies, but they are passed to it. This allows for more flexibility and a cleaner separation of concerns.

IoC is often implemented using a Dependency Injection (DI) container, which is responsible for managing the dependencies between objects and providing them to the objects that need them. The DI container uses reflection to create objects and wire them together based on the configuration provided in the application.

Inversion of Control is a design pattern that helps to **decouple** the components in a system, making them more flexible, reusable, and testable. It allows developers to write more **loosely-coupled code,** which results in a more maintainable and extensible system.

Meaning of decouple here ,

In software development, "**decoupling" refers to the separation of concerns between different components of a system**. When components are tightly coupled, they have a strong dependency on each other, meaning that changes to one component may have a ripple effect on other components. This can make the system difficult to maintain, test, and extend.

For example, consider a system that includes a user interface, a database, and a business logic layer. If the user interface is tightly coupled to the business logic layer and the database, any changes made to the user interface will likely require changes to be made to the business logic layer and the database as well. This can make the system difficult to maintain and test.

On the other hand, if the components are decoupled, changes to one component will have minimal impact on the others. The user interface can be changed without affecting the business logic layer or the database. This allows the system to be more maintainable, testable, and extensible.

Lets talk about IOC and DI implementations ,

In Spring framework, Dependency Injection (DI) and Inversion of Control (IoC) work together to provide a powerful and flexible way to manage dependencies between objects in an application. The Spring container is responsible for managing the dependencies and providing them to the objects that need them, using reflection and configuration provided in the application.

One Last point is , Spring Boot uses type 1 Inversion of Control, also known as "interface-based" Inversion of Control, which is based on the Hollywood Principle, where the system (the Spring container) is responsible for managing the dependencies between objects and providing them to the objects that need them.

In Short , IOC is nothing but the principle which says don't create/manage dependencies by own someone (container) will take care means system will provide you when object is created . and DI is nothing but the design pattern which says don't create hard coded dependencies instead of dynamic creation using reflection & annotations in Springboot.

Thank You Reading .

Report this

Published by

Jahid Momin
Team Lead I Serverless | ExpressJS | SpringBoot | Driving Innovation in Backend...
Published • 1y

6 articles

+ Follow

Discover the benefits of decoupling your components with Inversion of Control & Dependency Injection .

#springboot #java #spring #springframework #sharethispost

Like Comment Share 49 • 3 comments

Reactions

+37

3 Comments

Most relevant

Add a comment...

Amandeep Singh IN • 3rd+ 1y (edited) •••
Top Application Architecture Voice | SSE@Nagarro | Ex-Infosys/Amdocs | YouTube - Lazy Programmer | 5★ HackerRank Java

Great explanation. I have also tried explaining the concept at https://youtu.be/8xBF3RUMQfU

Like · 👍 2 | Reply · 1 Reply

Jahid Momin • 3rd+ 1y •••
Team Lead I Serverless | ExpressJS | SpringBoot | Driving Innovation in Backend Development | Empowering Learners in Selflearn Module

Thanks , Nice video with examples. 💯

Like · 👍 1 | Reply

Awais alwaisy • 2nd 7mo •••
Frontend Developer | Transforming Ideas into Digital Realities | Vue.js | React.js | JavaScript/TypeScript | Figma to Code

Well explained. How it goes in the React. When a component is dependent on Data from API, and state from the parent component, then how decoupling works?

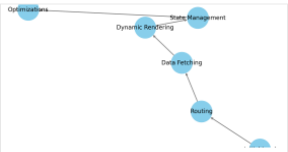
Like | Reply

Jahid Momin
Team Lead I Serverless | ExpressJS | SpringBoot | Driving Innovation in Backend Development | Empowering Learners in Selflearn Module

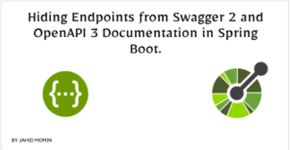
+ Follow

More from Jahid Momin

Messaging Queues :)
Jahid Momin on LinkedIn



What is an SPA & why its so popular ?
Jahid Momin on LinkedIn


Hiding Endpoints from Swagger 2 and OpenAPI 3 Documentation in Spring Boot.
BY JAHID MOMIN

Hiding Endpoints from Swagger 2 and OpenAPI 3 Documentation in Spring...
Jahid Momin on LinkedIn

See all 6 articles