selsoft
build better, deliver faster

Enterprise Application Development with Spring

Chapter 3: Inversion of Control (IoC) and Dependency Injection (DI)

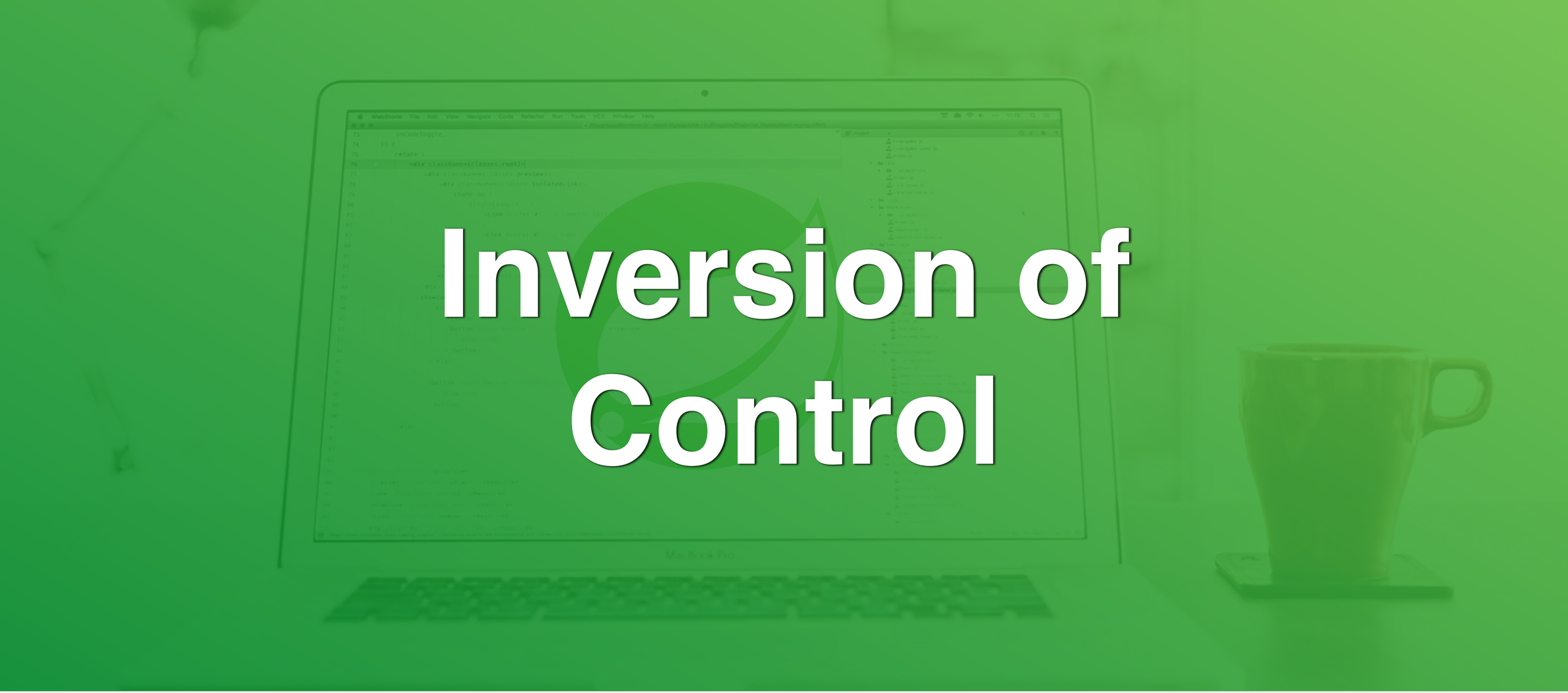
Instructor

Akın Kaldıroğlu

Expert for Agile Software Development and

Topics

- **Inversion of Control (IoC)**
  - History of IoC
  - Inversion of Control and Dependency Injection



Inversion of Control

selsoft
build better, deliver faster

Control of Flow

- **The control of flow** is about managing the flow (or flows in case of multi-threaded environment) in an application.
- Normally an application manages its flows i.e. it knows what to do in terms of creating a new object or thread, calling a method of an object it created or an object of another system, etc sequentially.
- So the application is the unique owner the control of its flows.

What is Framework? - I

- Framework is a semi-completed application with many abstractions that provides reusable foundation for applications.
- Frameworks need to be extended to be used in applications.
- An application extends a framework by either providing sub-classes to framework's mostly abstract classes or implementing its interfaces.
  - Those extension end points are also called **hook points**.

What is Framework? - II

- Frameworks can be of different types in terms of what they help to solve:
  - Application development frameworks
    - Web frameworks such as JSF, ADF or Spring MVC
    - Security frameworks such as JAAS or Spring Security
  - Business domain (domain-specific application) frameworks
    - Insurance frameworks such as Acord

What is IoC? - I

- **Inversion of Control** or **IoC** is about inverting the owner of the control of the flow in an application.
- It is a common principle applied by frameworks.
- A framework when extended by an application takes over the control of the flow of that application and manages it by making calls on its objects and the objects of the application.

What is IoC? - II

- So when the framework has the control of the flow the inversion of the control happens.
- The frameworks decides when and what to call.
- The application becomes a component that is used by the framework.
- But when an application uses a component or a library, it does not give up the control of the flow, it controls when to make calls on either its objects or libraries' objects.

Tomcat as A Servlet Container - I

- Think about a servlet container such as **Tomcat**.
- It creates all configured servlets and other necessary objects such as `ServletContext` for a web application to run.
- When a request comes Tomcat creates a request and response objects and injects them into the `doXXX()` methods of the mapped servlet provided by the developer depending on the HTTP method used in the request.

Tomcat as A Servlet Container - II

- When a request comes Tomcat also creates all necessary events and fires them according to the configuration.
- This is all achieved by the servlet contaier that implements the servlet framework through the mechanism of IoC.

History of IoC

History of IoC - I

- Stefano Mazzocchi used this concept in Avalon of Apache (https://avalon.apache.org/closed.html) and he says that IoC has been first coined by Michael Mattson in his thesis on "Object Oriented Frameworks: a survey on methodological issues" published in 1996.

History of IoC - II

- Michael Mattson says in page 96 of his thesis:

  An object-oriented framework is “a (generative) architecture designed for maximum reuse, represented as a collective set of abstract and concrete classes; encapsulated potential behaviour for subclassed specializations.”

  The major difference between an object-oriented framework and a class library is that the framework calls the application code. Normally the application code calls the class library. This inversion of control is sometimes named the Hollywood principle, “Do not call us, we call You”.

History of IoC - III

- But Johnson and Foote uses the term in their paper titled **Designing Reusable Classes** in JOOP of June 1988 (http://www.laputan.org/drc/drc.html):

  One important characteristic of a framework is that the methods defined by the user to tailor the framework will often be called from within the framework itself, rather than from the user's application code. The framework often plays the role of the main program in coordinating and sequencing application activity. This inversion of control gives frameworks the power to serve as extensible skeletons. The methods supplied by the user tailor the generic algorithms defined in the framework for a particular application.

History of IoC - IV

- Later M. Fowler wrote on the topic (https://martinfowler.com/bliki/InversionOfControl.html):

  Inversion of Control is a key part of what makes a framework different to a library. A library is essentially a set of functions that you can call, these days usually organized into classes. Each call does some work and returns control to the client.

  A framework embodies some abstract design, with more behavior built in. In order to use it you need to insert your behavior into various places in the framework either by subclassing or by plugging in your own classes. The framework's code then calls your code at these points.

greeting08

- `org.javaturk.spring.di.ch03.greeting08`

Benefits of Using Frameworks

- By using a framework an application gains a leverage with the reusability of framework but looses the control of the flow.
- Main gains of using a framework would be:
  - Faster development,
  - Focusing on business by outsourcing boilerplate code to the framework,
  - More reliable code,
  - etc.

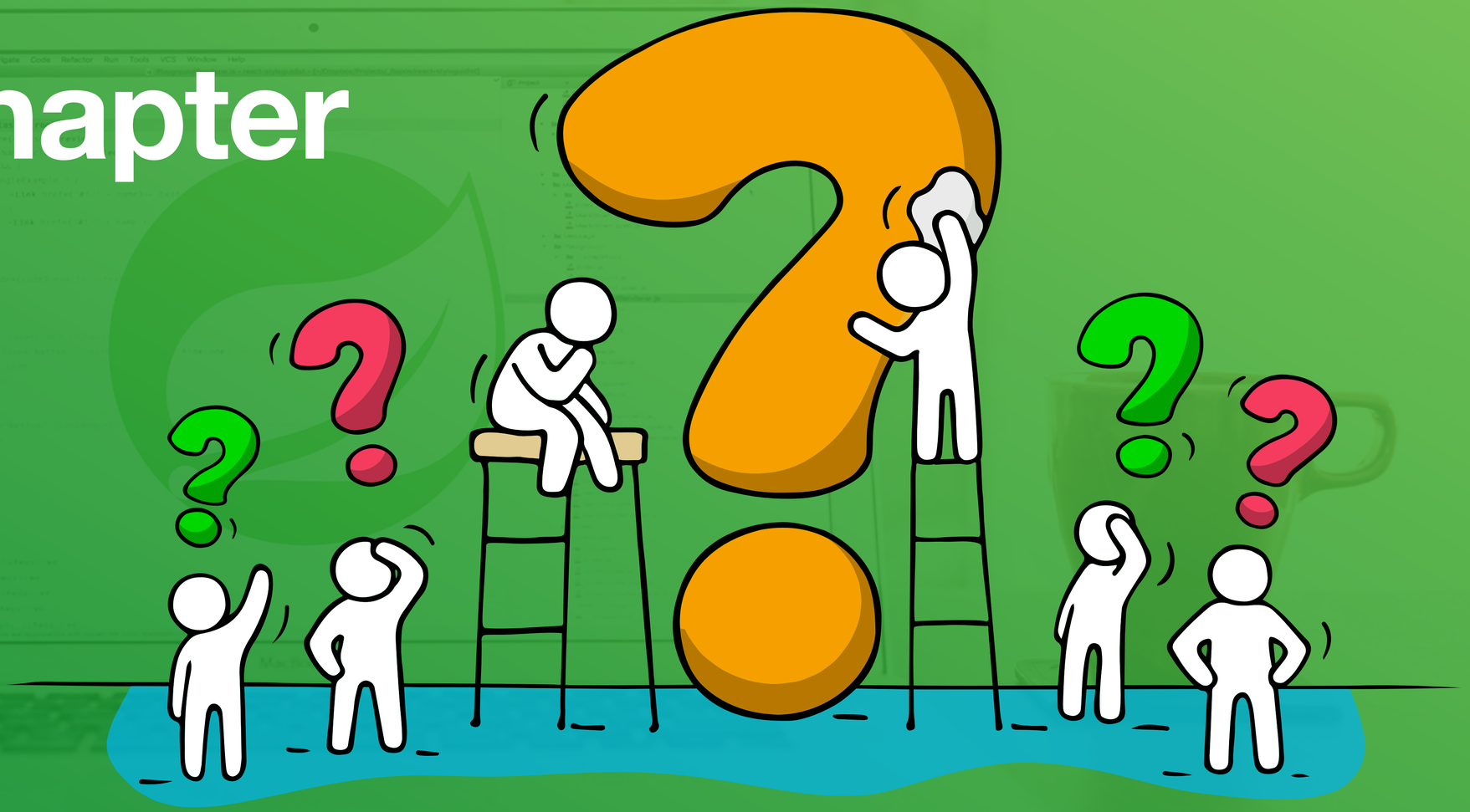
IoC & DI

DI and IoC

- Dependency Injection (DI) is achieved by IoC.
- So IoC is an underlying principle applied in the inner workings of frameworks and DI is one of many techniques where IoC is applied.
  - IoC is used to enable DI.
- IoC is used in many places other than creating and wiring objects such as managing object's life cycle, events, AOP, etc.
- Since all those areas eventually work on objects and create dependencies, DI is at the core of all of them.

End of Chapter

Time for Questions!



selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr