Announcements

- REST assignment due this Thursday

- Quiz 2 Statistics
  - Mean: 13.49/15; Std Dev: 1.36

- GCP assignment out this Thursday
  - Based on deployment of a very simple Java application (logger) on the Google App Engine
  - Individual Assignment

CS3300 Introduction to Software Engineering

Lecture 07: Tools of the Trade #4

Spring, Spring Boot

Dr. Nimisha Roy ▸ nroy9@gatech.edu

Contents

- Java Frameworks
- Spring
  - Advantages, IoC, Dependency Injection
- Spring Boot
  - Starter Projects, Auto Configurations
- Demo

Frameworks

- Framework is a predefined set of tools, libraries, best practices, and conventions that helps developers create applications more efficiently and effectively.

- Provides a foundation on which developers can build programs for a specific platform.

Provides a consistent structure for collaboration	Reusable Code	Includes Best practices for foundational elements
Extensions and Plugins	Inversion of Control	Learning Curve

Java Frameworks

- Java framework is a body of reusable prewritten code acting as templates used by developers to create apps using the Java programming language

Web frameworks

- SpringBoot, Vaadin, JavaServer Faces, JHipster

Data Access frameworks

- Hibernate, Java Persistence API, MyBatis

Microservices

- Spring Cloud, MicroProfile, Quarkus

Big Data

- Apache Kafka, Apache Spark, Apache Hadoop

Testing

- Junit, TestNG, Mockito, Spring Test

Security

- Spring Security

Framework *vs.* API

- Framework serves as a foundation for programming, while an API provides access to the elements supported by the framework.
- Framework includes an API
- Will also include code libraries, compiler and other programs needed for software development

Spring

- Most popular, powerful, lightweight and open-source application development framework used for Java Enterprise Edition (JEE). Other frameworks include Hibernate, JSF, Struts etc.
    - JEE is built upon Java SE (Standard Edition) . Provides functionalities like web application development, servlets etc.
    - JEE provides APIs for running large scale applications

Advantages of Spring

- MVC architecture. Distinct division between models (data), controllers (application logic) and views(user interface).
- Framework of frameworks. Can easily integrate with other frameworks like Struts, Hibernate etc.
- Flexibility
- One stop-shop for all enterprise applications. But modular, allows you to pick which modules you need.
- Allows loose coupling among modules
- Easier to test
- Increases efficiency due to reduction in application development time

Inversion of Control (IoC)

- Principle in software engineering which transfers the control of objects or portions of a program to a container or framework. Most often used in the context of object-oriented programming. The architecture helps in decoupling the execution of a task from its implementation.

- In Spring, objects configured in XML file and Spring container is responsible for creation and deletion of objects by parsing XML file.

Inversion of Control (IoC)

Example of highly coupled classes

```
public class DataAccess
{
    public DataAccess()
    {
    }

    public string GetCustomerName(int id) {
        return "Dummy Customer Name"; // get it
    }
}
```

```
public class CustomerBusinessLogic
{
    DataAccess _dataAccess;

    public CustomerBusinessLogic()
    {
        _dataAccess = new DataAccess();
    }

    public string GetCustomerName(int id)
    {
        return _dataAccess.GetCustomerName(id);
    }
}
```

- *CustomerBusinessLogic* and *DataAccess* classes are tightly coupled. Changes in the *DataAccess* class will lead to changes in the *CustomerBusinessLogic* class. For example, if we add, remove or rename any method in the DataAccess class then we need to change the CustomerBusinessLogic class accordingly.

- The *CustomerBusinessLogic* class creates an object of the *DataAccess* class using the **new** keyword. There may be multiple classes which use the *DataAccess* class and create its objects. So, if you change the name of the class, then you need to find all the places in your source code where you created objects of *DataAccess* and make the changes throughout the code.

- Because the *CustomerBusinessLogic* class creates an object of the concrete *DataAccess* class, it cannot be tested independently (TDD). The DataAccess class cannot be replaced with a mock class.

Inversion of Control (IoC)

Using Factory pattern of the IoC principle => Loosely coupled design

```
public class CustomerBusinessLogic
{

    public CustomerBusinessLogic()
    {
    }

    public string GetCustomerName(int id)
    {
        DataAccess _dataAccess =  DataAccessFactory.GetDataAccessObj();

        return _dataAccess.GetCustomerName(id);
    }
}
```

```
public class DataAccessFactory
{
    public static DataAccess GetDataAccessObj()
    {
        return new DataAccess();
    }
}
```

```
public class DataAccess
{
    public DataAccess()
    {
    }

    public string GetCustomerName(int id) {
        return "Dummy Customer Name"; // get it
    }
}
```

The *CustomerBusinessLogic* class uses the *DataAccessFactory*.*GetCustomerDataAccessObj*() method to get an object of the *DataAccess* class instead of creating it using the new keyword. Thus, we have inverted the control of creating an object of a dependent class from the *CustomerBusinessLogic* class to the *DataAccessFactory* class.

first step towards achieving fully loose coupled design.

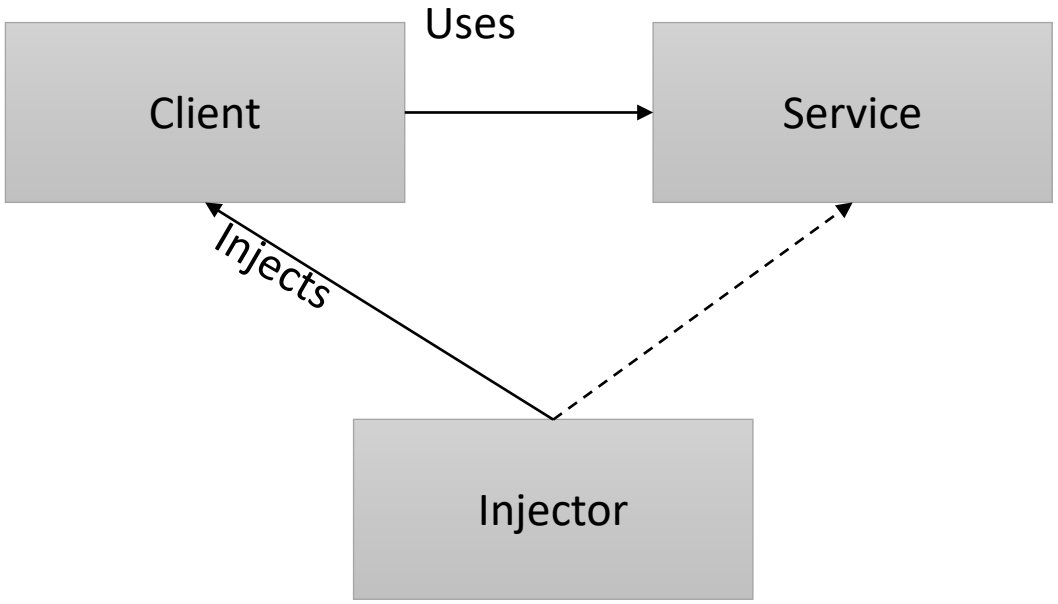
Dependency Inversion Principle (DIP)

- A high-level module (depends on other modules) should not depend on low-level modules (*DataAccess* Class). Both should depend on abstraction.
- Abstractions should not depend on details. Details should depend on abstractions

- Abstraction in OOPS means to create an interface or an abstract class which is non-concrete. This means we cannot create an object of an interface or an abstract class.

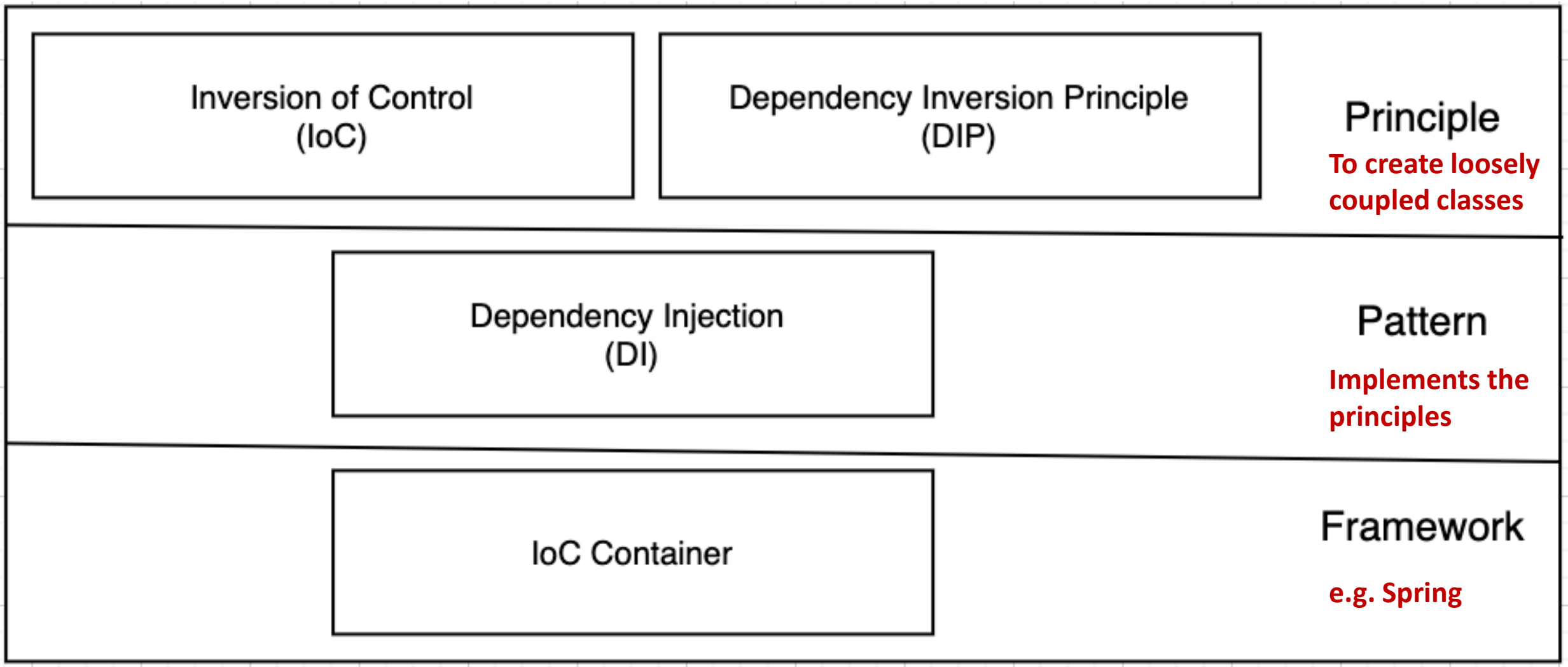
Dependency Injection (DI)

- Dependency Injection (DI) is a design pattern used to implement IoC. It allows the creation of dependent objects outside of a class and provides those objects to a class through different ways.
- Increases possibility to reuse classes and test independently of other classes while unit testing
- DI pattern includes 3 class types:



Injector class creates an object of service class, injects it to a client object. Hence, separates the responsibility of creating an object of service class out of the client class.

IoC, DIP, DI


Inversion of Control
(IoC)
Dependency Inversion Principle
(DIP)
Principle
To create loosely coupled classes
Dependency Injection
(DI)
Pattern
Implements the principles
IoC Container
Framework
e.g. Spring

Spring Boot

- Spring Boot builds on top of Spring Framework, offering a streamlined approach to developing Spring applications with minimal boilerplate code, auto configuration, embedded servers, and other features. It means that you can **just run t**he application.

- Dependencies and configurations are managed by Spring Boot.

- Normally to run application, you need: hardware + OS + Server +Application file (.war for web applications). With Spring Boot: server embedded (Tomcat), executable files generated automatically.

- Features:
  - Provides with a starter project for the application along with auto configuration
  - Does not generate XML file but configuration can be modified (using YAML files, properties or XML)

DEMO TIME!!

- Create a simple Web app using SpringBoot
- We needed Express in Node.js to establish the server. Spring Boot has Tomcat server embedded– very convenient

WHAT IF WE HAD TO CREATE THE SAME APP WITHOUT SPRING BOOT

- We have Spring framework
- Setup and manage all maven dependencies and versions in pom.xml manually – very extensive
- Define Web.xml file to configure web related front controller
- Define a Spring context XML file to define component scans
- Install Tomcat or configure Tomcat maven plugin
- Deploy and run application IN TOMCAT

Spring Boot Starter Projects

- Goal is to help you get a project up and running quickly
  - Web application: Spring-Boot-Starter-Web
  - REST API: Spring-Boot-Starter-Web
  - Talk to database using JPA- Spring-Boot-Starter-Data-JPA
  - Talk to database using JDBC- Spring-Boot-Starter-JDBC
  - Secure web application- Spring-Boot-Starter-Security

- Manage list of maven dependencies & versions for different apps:
  - Spring-Boot-Starter-Web: Frameworks needed by typical web applications. Spring-webmvc, spring-web, spring-boot-starter-tomcat, sprint-boot-starter-json

Spring Boot Auto Configuration

- Starter defines dependencies
- Auto configurations provides basic configuration to run application using frameworks defined in your maven dependencies
- Decided based on:
  - Which frameworks are in class path?
  - What is the existing configuration? (*Maven Dependencies* – *springframework.boot.autoconfigure* – classes that will be checked)
  - Type *logging.level.org.springframework=DEBUG* in *application.properties* to see what is being auto-configured.