Overview of the Spring Framework

1.18.5

Module Objectives

After completing this lesson, you should be able to do the following

- Define the Spring Framework
- Explain what Spring is used for
- Discuss why Spring is successful
- Explain where it fits in your world

Agenda

- **What is the Spring Framework?**
- Spring is a DI Container
- Spring Framework History
- What is Spring Used For?

What is the Spring Framework?

Spring is an Open Source, Lightweight, DI (Dependency Injection) Container and Framework for building Java enterprise applications

- ✓ Open Source
- ✓ Lightweight
- ✓ DI Container (IoC Container)
- ✓ Framework

Spring Framework is Open Source

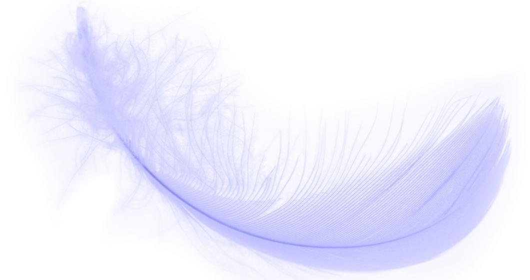
- Spring binary and source code are freely available
- Apache 2 licence
- Code is available at:
  - https://github.com/spring-projects/spring-framework
- Binaries available at Maven Central
  - http://mvnrepository.com/artifact/org.springframework
- Documentation available at:
  - https://docs.spring.io/spring-framework/docs/current/reference/html/

Bookmark This!

The use of a transitive dependency management system (Maven, Gradle, Ant/Ivy) is recommended for any Java application

The Spring Framework is Lightweight

- Spring applications do not require a Java EE application server
  - But they can be deployed on one
- Spring is not *invasive*
  - Does not require you to extend framework classes or implement framework interfaces for most usage
  - You write your code as POJOs
- Low overhead
  - Spring jars are relatively small



The Spring Framework Provides a DI Container

- Spring serves as a Dependency Injection (DI) container for your application objects
  - Your objects do not have to worry about finding / connecting to each other
- Spring instantiates and injects dependencies into your objects
- Spring also serves as a lifecycle manager

Dependency Injection (DI) Container is sometimes called Inversion of Control (IoC) Container
https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#beans-introduction

Spring Framework: More Than Just a DI Container

- Enterprise applications must deal with a wide variety of technologies / architectures / deployment-platforms
  - Containerization, Cloud, Micro-services
  - JDBC, Transactions, ORM / JPA, NoSQL
  - Events, Streaming, Reactive, Messaging, JMS, AMQP, Tasks, Scheduling
  - Security, OAuth2, OpenID Connect
  - Monitoring, Observability
  - …
- Spring provides framework classes, interfaces, and annotations to simplify working with lower-level technologies
- Highly extensible and customizable

Agenda

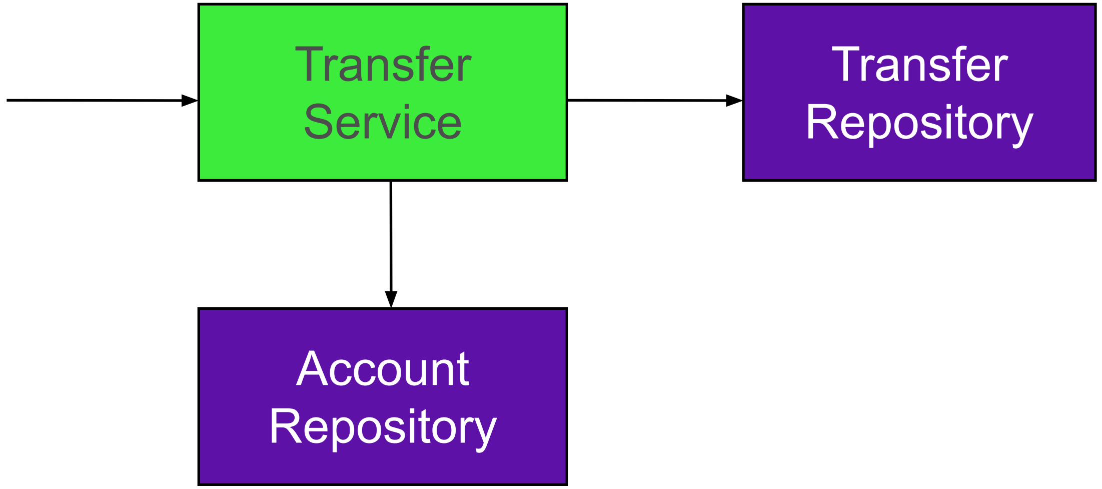
- What is the Spring Framework?
- **Spring is a DI Container**
- Spring Framework History
- What is Spring Used For?

Goal of the Spring Framework

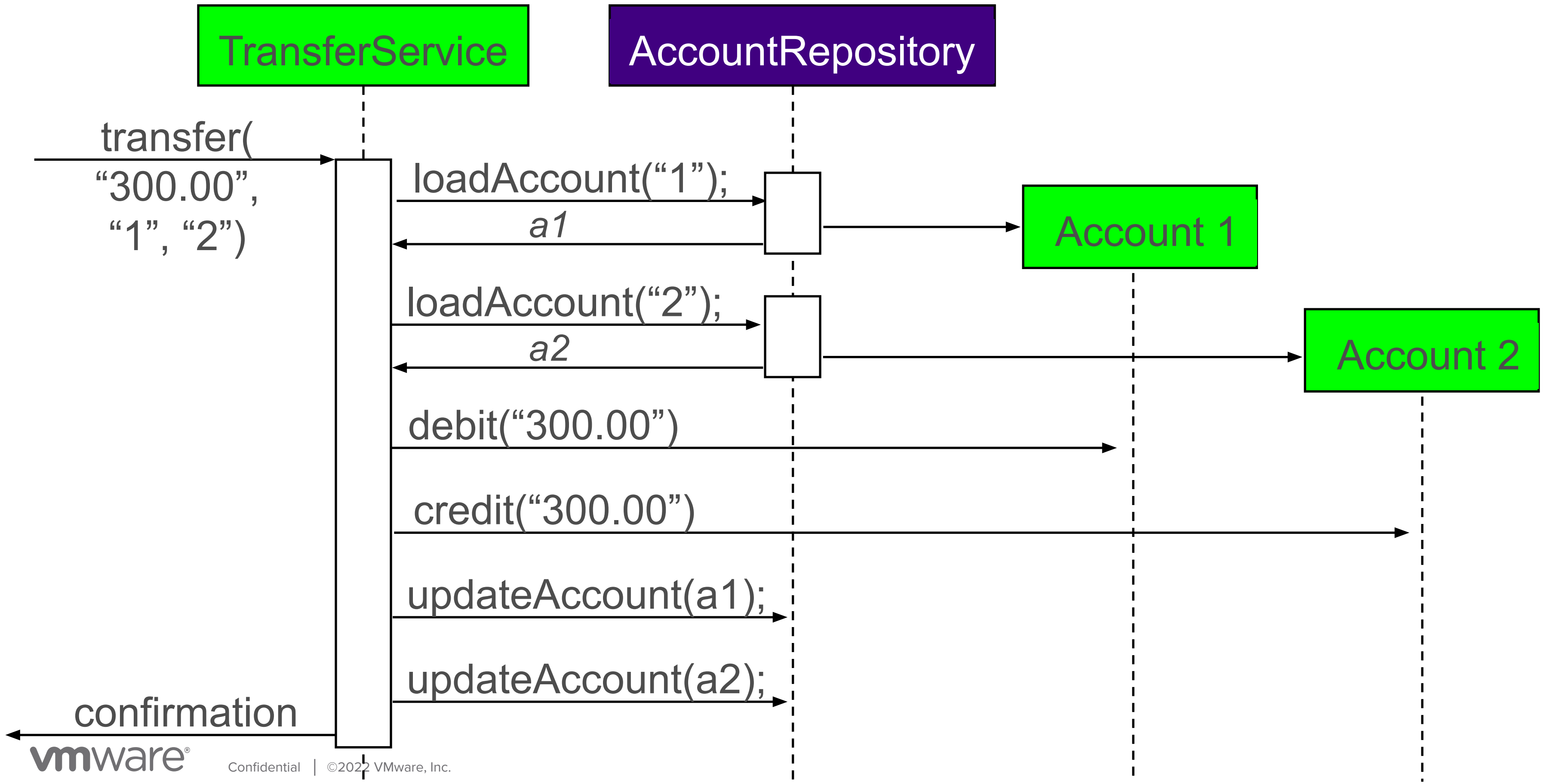
- Provide comprehensive infrastructural support for developing enterprise Java applications
  - Spring deals with the plumbing
  - You focus on solving the business domain problems
- *Key Principles*
  - Don’t Repeat Yourself (DRY)
  - Separation of Concerns
  - Convention over Configuration
  - Testability

DI Example: Banking Application Configuration

A typical application consists of several parts working together to carry out a use case



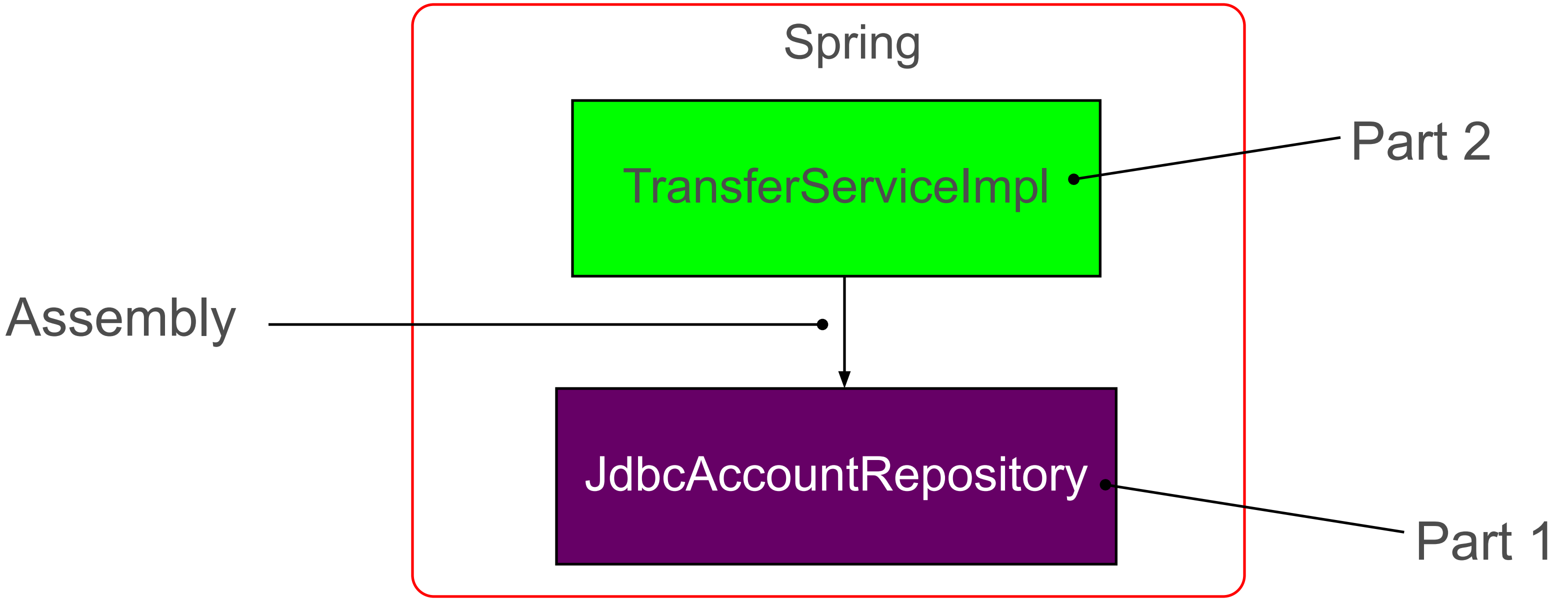
Example: Do Money Transfer



Questions to Consider

- How would we configure the application to ensure all components are assembled correctly?
  - How can we instantiate *TransferService* and *AccountRepository* objects?
  - How can we make the *AccountRepository* object available to the *TransferService* object?
- How can we easily swap out an implementation without re-writing the application?
  - How can we make different types of *AccountRepository* objects available to the *TransferService* object?

Money Transfer System Assembly



```
(1) repository = new JdbcAccountRepository(…);
(2) service = new TransferServiceImpl();
(3) service.setAccountRepository(repository);
```

Parts are Just Plain Old Java Objects (POJOs)

```
public class JdbcAccountRepository implements AccountRepository {
    …
}
```

Part 1

Implements an interface

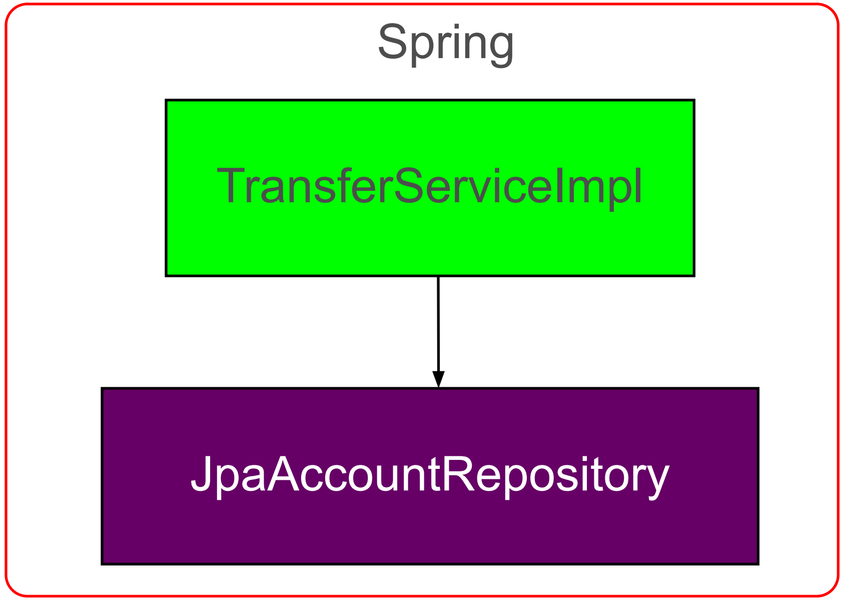
```
public class TransferServiceImpl implements TransferService {
    private AccountRepository accountRepository;

    public void setAccountRepository(AccountRepository ar) {
        accountRepository = ar;
    }
    …
}
```

Part 2

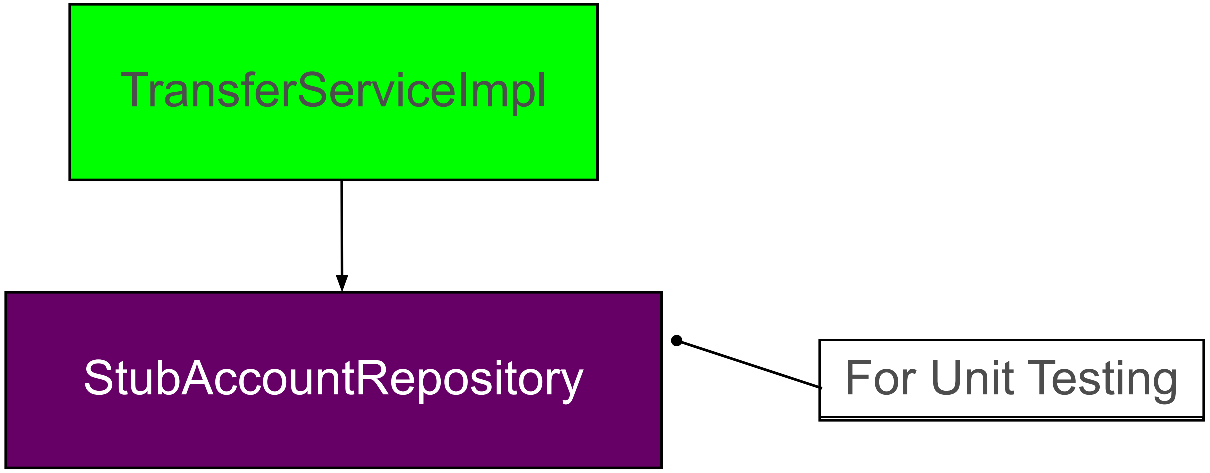
Depends on an *interface:*
– conceals complexity of implementation;
– allows for swapping out implementation

Swapping Out Part Implementations



```
(1) repository = new JpaAccountRepository(…);
(2) service = new TransferServiceImpl();
(3) service.setAccountRepository(repository);
```

Swapping Out Part Implementations



```
(1) repository = new StubAccountRepository();
(2) service = new TransferServiceImpl();
(3) service.setAccountRepository(repository);
```

Agenda

- What is the Spring Framework?
- Spring is a DI Container
- **Spring Framework History**
- What is Spring Used For?

Why is Spring Successful?

- Started in the early 2000s with Rod Johnson's book
- Java ecosystem was radically different than today
  - J2EE APIs were often difficult to use and test
  - Spring aimed to simplify
    - Configuration via Dependency Injection
    - Transaction Management and JDBC Data Access
    - Support for multiple deployment environments
- Spring becomes popular as an example of creating enterprise applications
  - Integration with [selected JSR Specs]

Programmer to Programmer
Rod Johnson
Rod Johnson is an enterprise Java architect specializing in scalable web applications. He has worked with both Java and J2EE since their release, and he is a member of JSR 154 Expert Group defining the Servlet 2.4 specification.
expert one-on-one
J2EE Design and Development
wrox

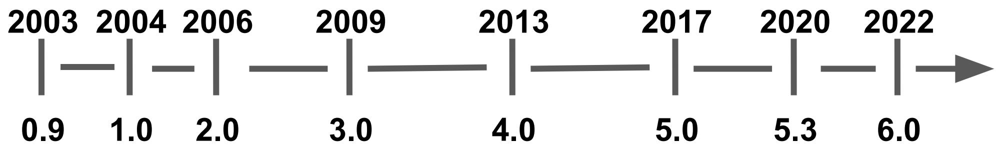
Why is Spring Successful?

- Provide choice at every level
- Embrace change and different perspectives
- Strong backwards compatibility
- Careful API design
- High standard for code quality
- OSS Community
- Developer support on forums, Stack Overflow
- Support of conferences and user groups

Spring excels at being adaptable to change

- Initial integration with other open source projects
  - Hibernate, Quartz, Multiple View Technologies
- Spring projects created for common enterprise domains
  - Spring Security, Batch, Integration
- Spring projects created for new domains
  - Spring Data: NoSQL + JPA, Spring Cloud
- Spring Boot created to further simplify DevEx
- Spring Framework support for Kotlin

Spring Framework Release Timeline:



Agenda

- What is the Spring Framework?
- Spring is a DI Container
- Spring Framework History
- **What is Spring Used For?**

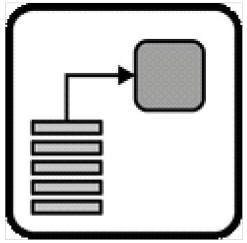
What is Spring Used For?

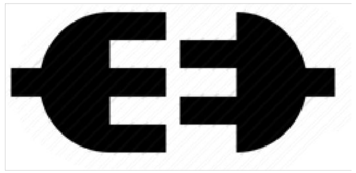
- Spring provides comprehensive infrastructural support for developing enterprise Java applications
  - Spring deals with the plumbing
  - So you can focus on solving the business domain
- Spring used to build enterprise applications dealing with:

Web apps

Messaging

Persistence


Batch/Tasks


Integration/Streaming

The Current World

- Spring continues to adapt and innovate
  - **JDK Versions**
  - **Native Compilation**
  - **Reactive Programming**
  - **Stream Processing**
  - **Kotlin Support**
  - **Kubernetes**

Lab: Developing an Application from Plain Old Java Objects

Lab project:

https://github.com/Nimedas/imt-spring-2024