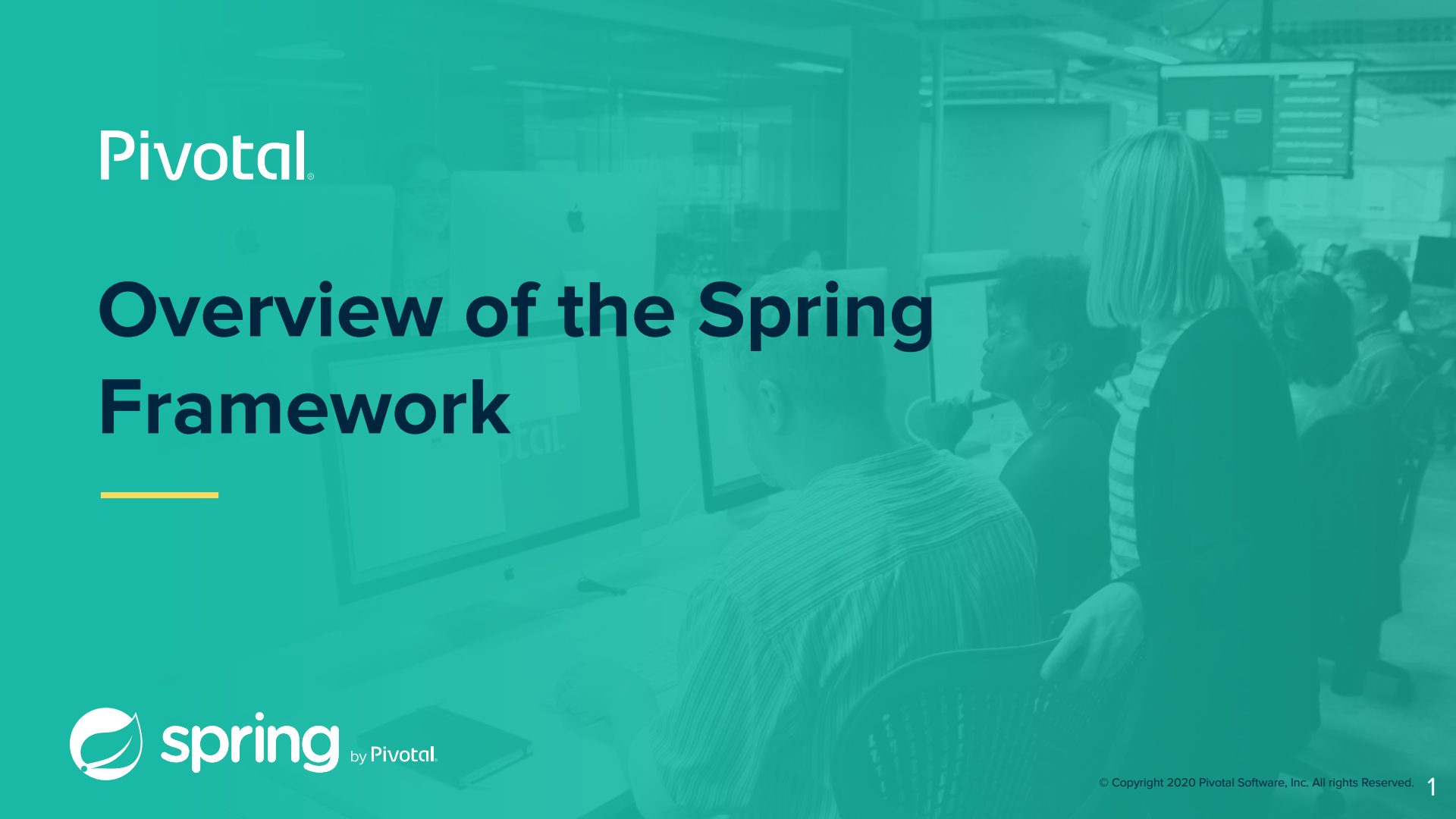
Pivotal

Overview of the Spring Framework



spring by Pivotal

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 Container
- Spring Framework History
- What is Spring Used For?

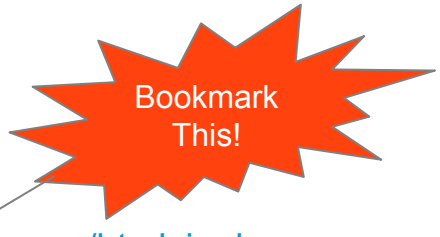
What is the Spring Framework?

Spring is an Open Source, Lightweight, Container and Framework for building Java enterprise applications

- Open Source
- Lightweight
- Container
- Framework

Spring Framework is Open Source

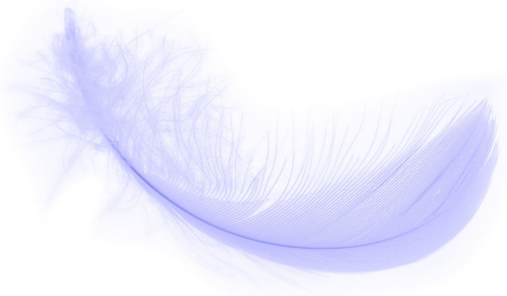
- Spring binary and source code are freely available
- Apache 2 licence
- Code is available at:
  - https://github.com/spring-projects/spring-framework
- Binaries available at Maven Central
  - http://mvnrepository.com/artifact/org.springframework
- Documentation available at:
  - http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle



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
    - JARs used in this course are < 8MB



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 dependency injects your objects
  - Serves as a lifecycle manager

Spring Framework: More Than Just a DI Container

- Enterprise applications must deal with a wide variety of technologies / resources
  - JDBC, Transactions, ORM / JPA, NoSQL, Security, Messaging, JMS, AMQP, Streaming, Events, Web, Tasks, Scheduling, Mail, Files, XML/JSON Marshalling, Remoting, REST services, SOAP services, Mobile, Social, Cloud, ...
- Spring provides framework classes, interfaces, and annotations to simplify working with lower-level technologies

Agenda

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

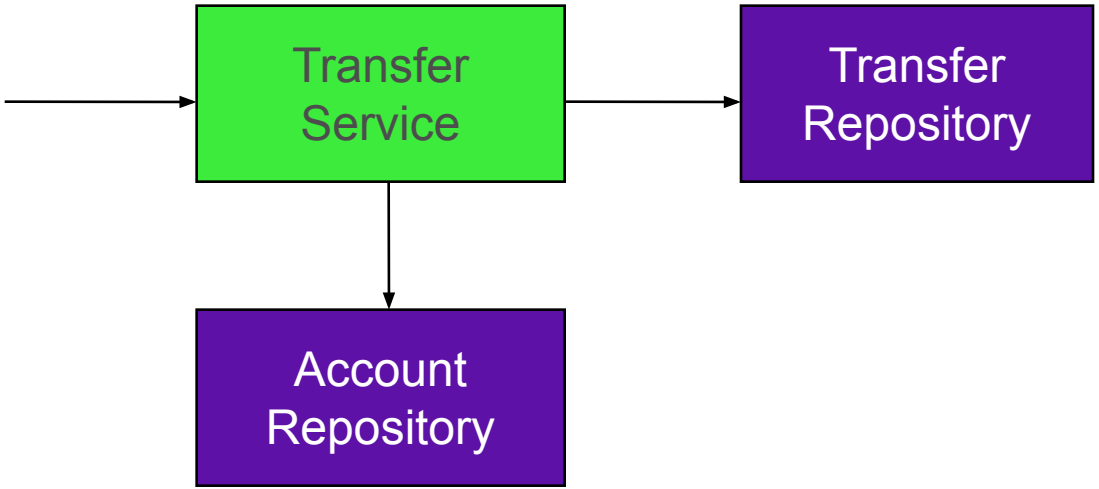
Goal of the Spring Framework

- Provide comprehensive infrastructural support for developing enterprise Java™ applications
  - Spring deals with the plumbing
  - You can focus on solving the domain problem

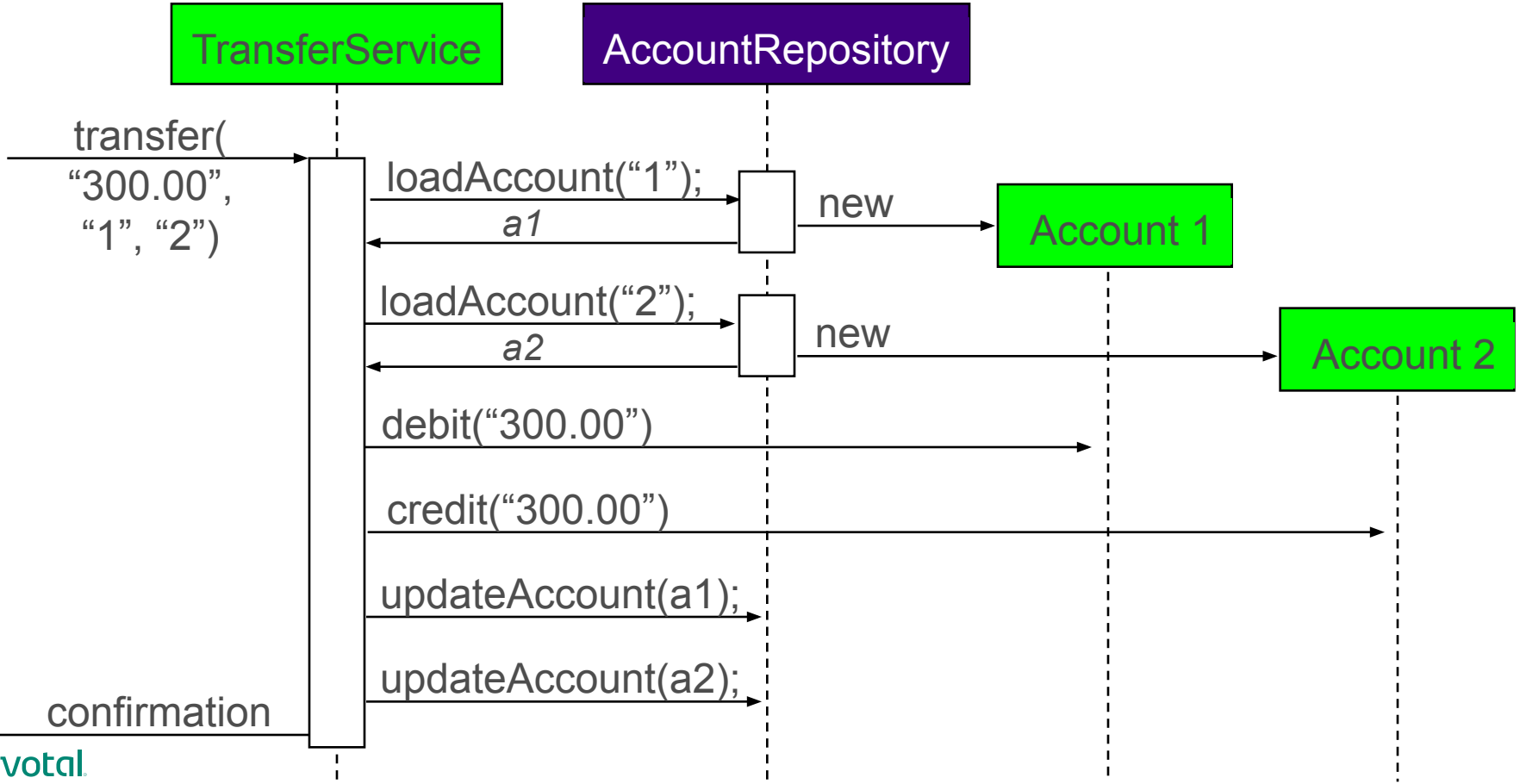
- *Key Principles*
  - **DRY** - Don't Repeat Yourself
  - **SoCs** - Separation of Concerns

Example: Banking Application Configuration

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



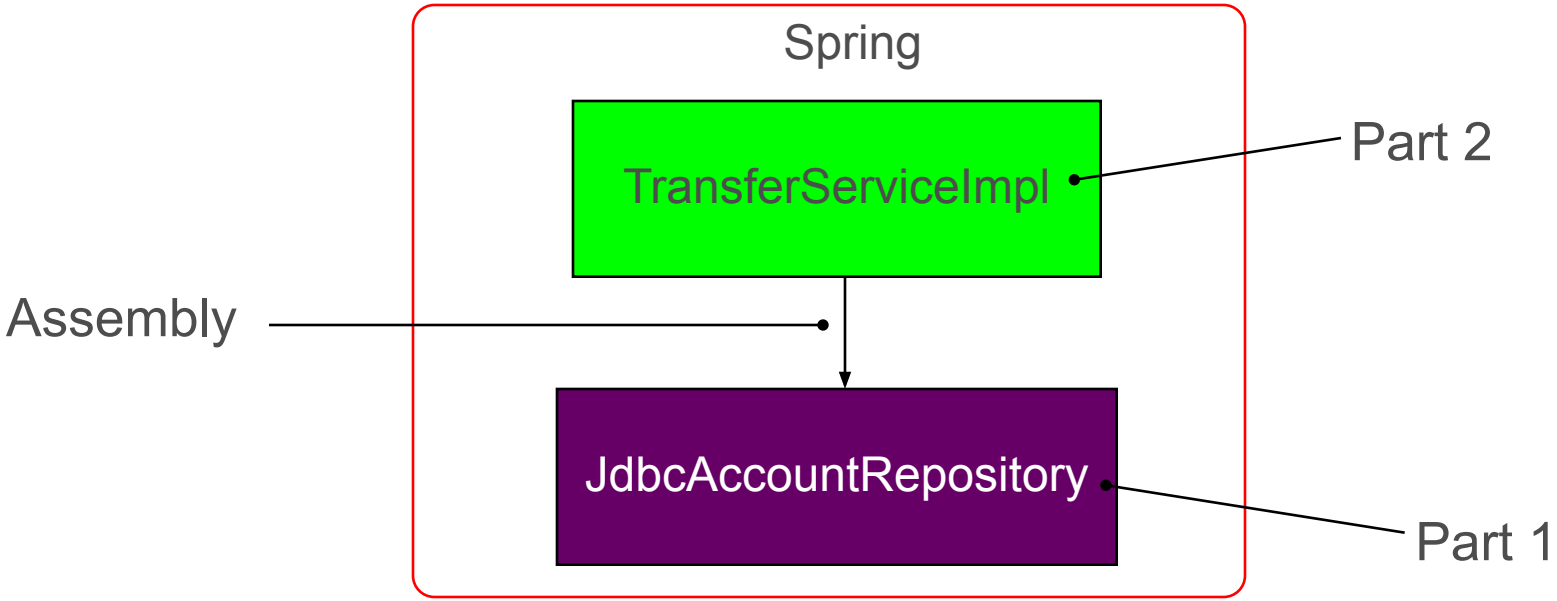
Example: Do Money Transfer



Questions to Consider

- How would we configure the application to ensure all components are assembled correctly?
- How can we easily swap out an implementation without re-writing the application?

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 {
    …
}
```

Implements an interface

Part 1

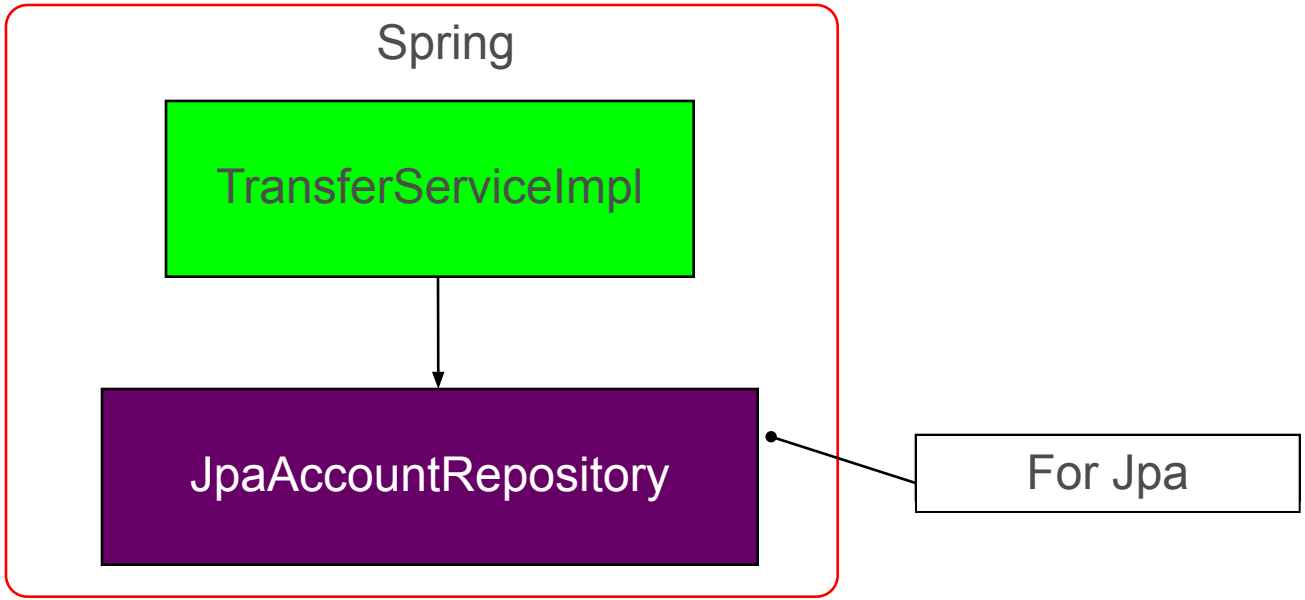
```
public class TransferServiceImpl implements TransferService {
    private AccountRepository accountRepository;

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

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

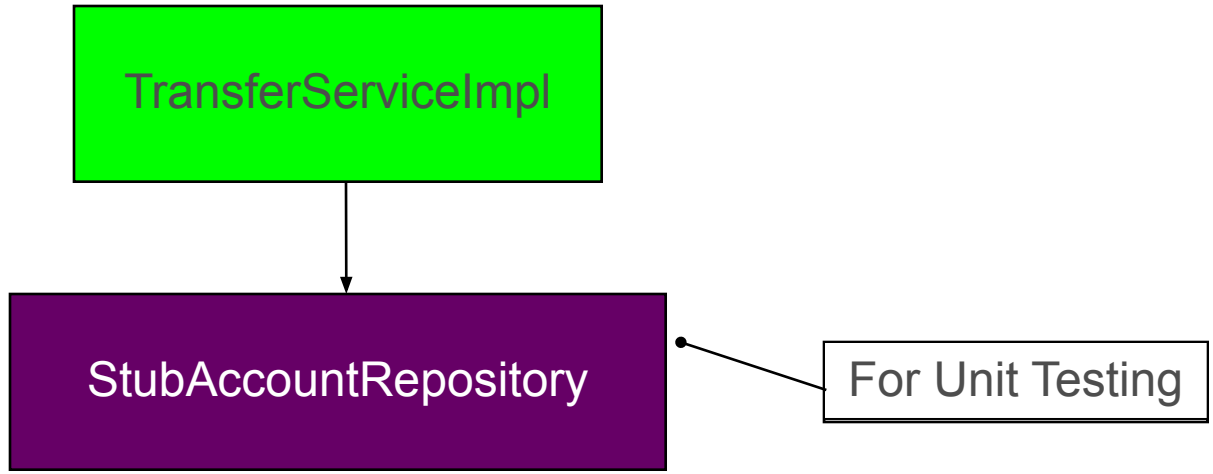
Part 2

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 Container
- **Spring Framework History**
- What is Spring Used For?

Why is Spring Successful? Brief History of Java

- The early years
  - 1995: Java introduced, Applets are popular
  - 1997: Servlets introduced
    - Efficient, dynamic web pages become possible
  - 1999: JSP introduced
    - Efficient, dynamic web pages become easy
- Questions arise regarding “Enterprise” applications
  - How should a Servlet/JSP application handle
    - Persistence? Business logic? Transactions? Messaging? Security? Etc?

Introducing J2EE and EJB

- Java's answer: J2EE
  - 1999: J2EE introduced
    - Featuring Enterprise Java Beans (EJB)
    - Answers the questions of persistence, transactions, business logic, security, etc
- However EJBs prove to be problematic:
  - Difficult to code
    - Must extend/implement specific classes/interfaces
    - Complicated programming model required
  - Difficult to unit test
  - Expensive to run
    - Must have application server, resource intensive

The Birth of Spring

- Rod Johnson publishes J2EE Development without EJB

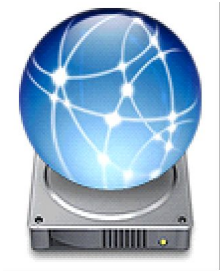
- 2004: Spring Framework 1.0 released
  - Champions Dependency Injection (DI)
  - Encourages POJOs
  - Uses XML files to describe application configuration
  - Becomes popular quickly as an EJB alternative

Agenda

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

What is Spring Used For?

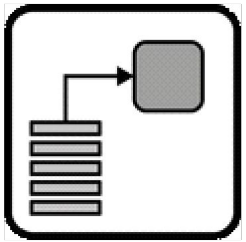
- Spring provides comprehensive infrastructural support for developing enterprise Java ™ applications
  - Spring deals with the plumbing
  - So you can focus on solving the business domain
- Spring used to build enterprise applications dealing with:



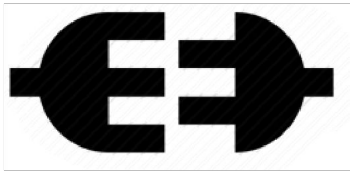
Web apps

Messaging

Persistence



Batch/Tasks



Integration/Streaming

The Current World

- Spring is not simply an alternative to Java EE/EJB
  - Modern application development are different today than 2000
- Spring continues to innovate
  - **Web**: AJAX, WebSockets, REST, Mobile, Reactive
  - **Data**: NoSQL, Big Data, Stream processing
  - **Cloud**: Distributed systems, Cloud, Microservices
  - **Productivity**: Spring Boot, Spring Cloud Data Flow
  - And many more

More on Spring's Ecosystem

- Visit http://spring.io/projects

Lab: Developing an Application from Plain Old Java Objects

Lab project:
10-spring-intro

Anticipated Lab time:
30 Minutes