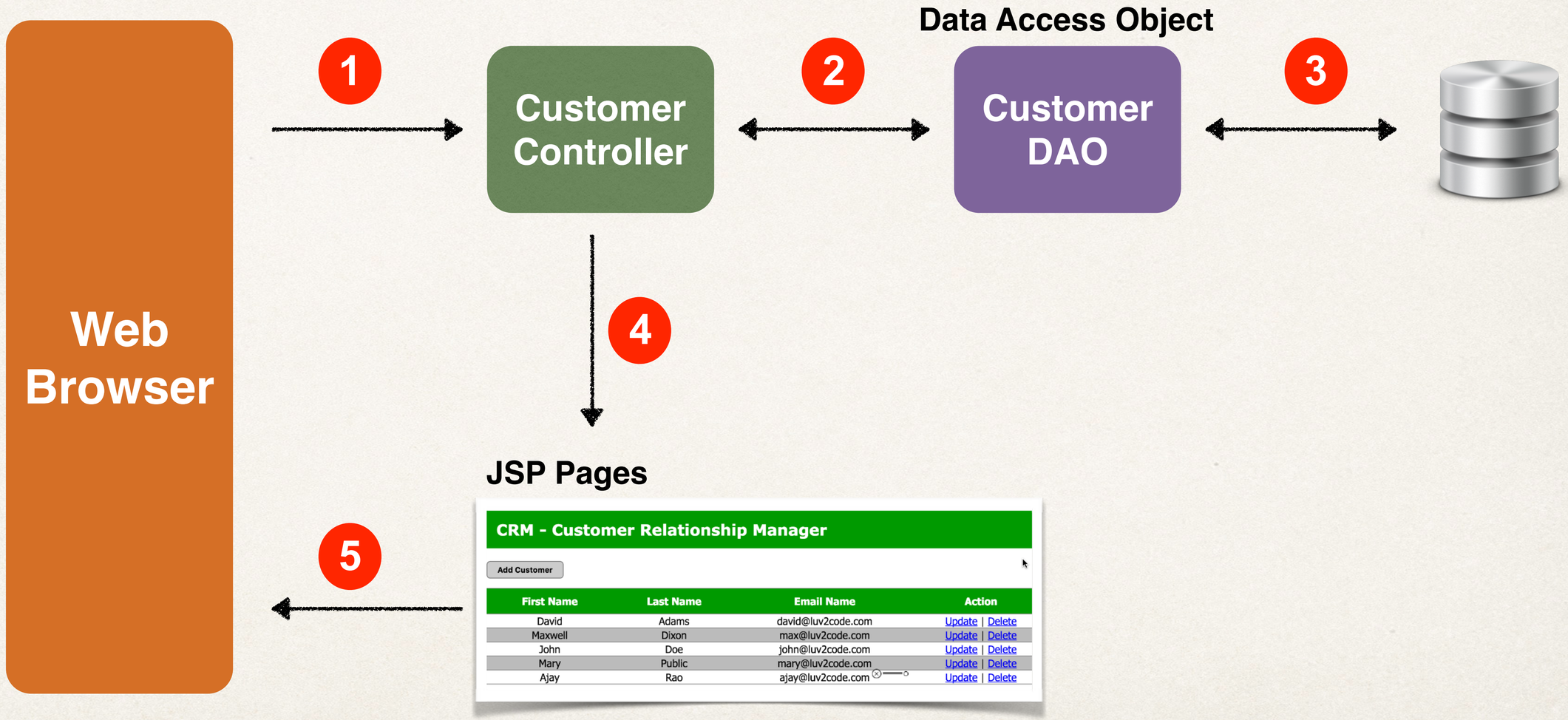
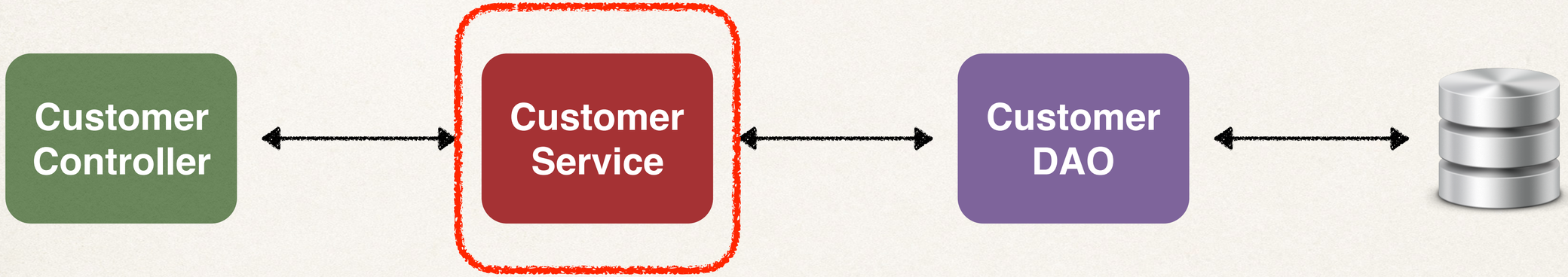
luv2code

Define Services with @Service

Big Picture

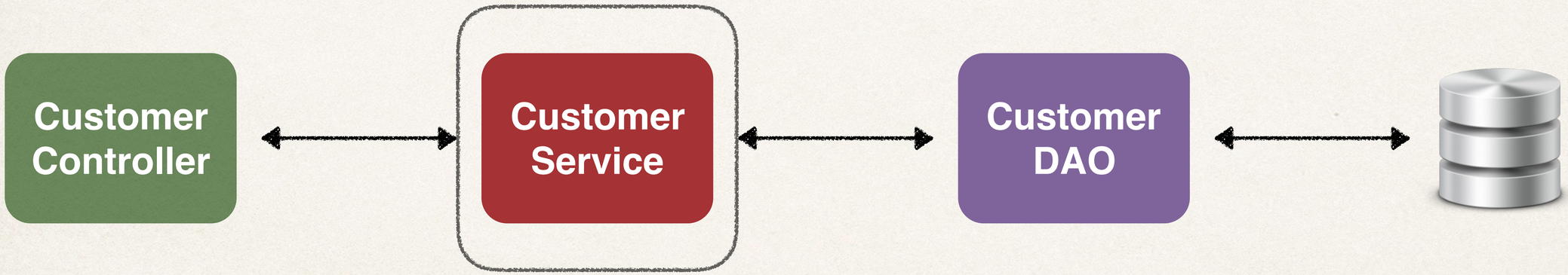

Data Access Object
Web Browser
1
Customer Controller
2
Customer DAO
3
4
JSP Pages
5
CRM - Customer Relationship Manager
Add Customer
First Name
Last Name
Email Name
Action
David
Adams
david@luv2code.com
Update | Delete
Maxwell
Dixon
max@luv2code.com
Update | Delete
John
Doe
john@luv2code.com
Update | Delete
Mary
Public
mary@luv2code.com
Update | Delete
Ajay
Rao
ajay@luv2code.com
Update | Delete

Refactor: Add a Service Layer

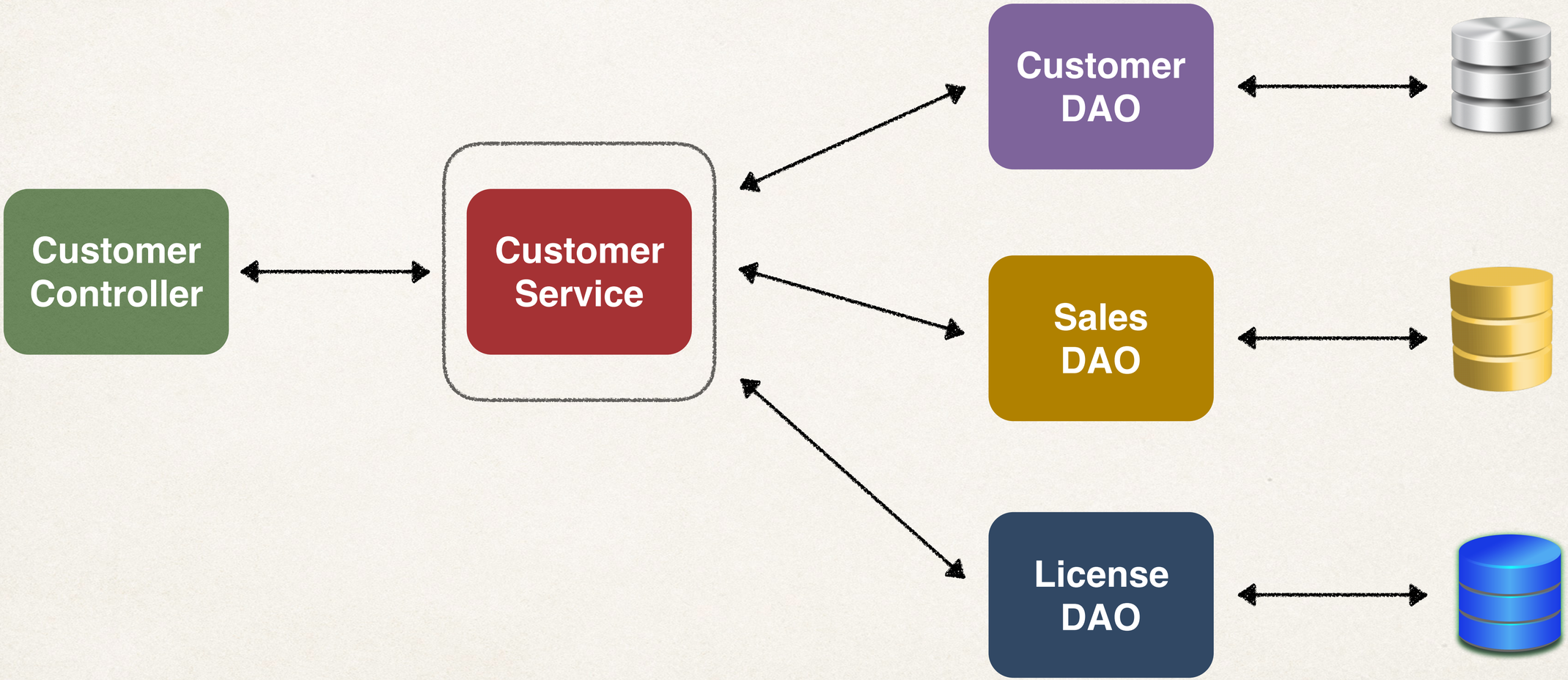


Purpose of Service Layer

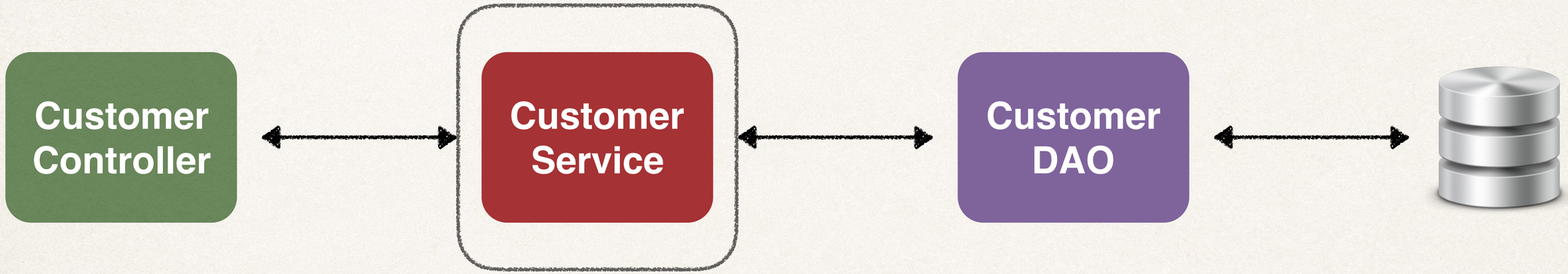
- **Service Facade** design pattern
- Intermediate layer for custom business logic
- Integrate data from multiple sources (DAO/repositories)



Integrate Multiple Data Sources

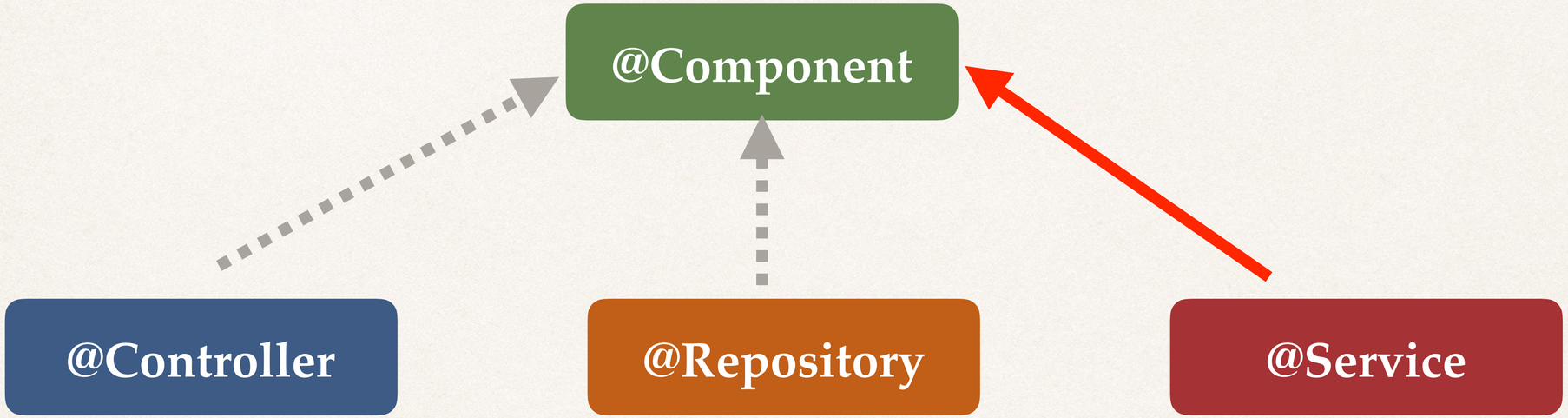


Most Times - Delegate Calls



Specialized Annotation for Services

- Spring provides the **@Service** annotation



Specialized Annotation for Services

- **@Service** applied to Service implementations

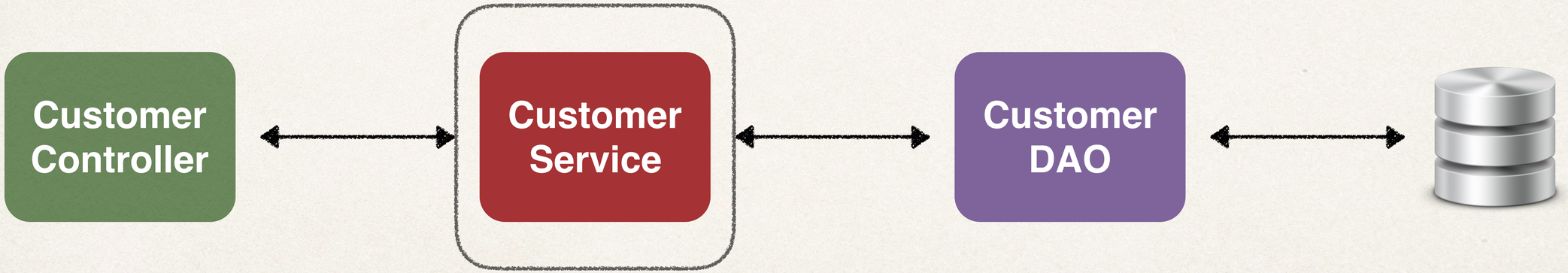
- Spring will automatically register the Service implementation
  - thanks to component-scanning

Customer Service

Step-By-Step

1. Define Service interface

2. Define Service implementation
   - Inject the CustomerDAO



Step 1: Define Service interface

```
public interface CustomerService {

   public List<Customer> getCustomers();

}
```

Step 2: Define Service implementation

```
@Service
public class CustomerServiceImpl implements CustomerService {

   @Autowired
   private CustomerDAO customerDAO;

   @Transactional
   public List<Customer> getCustomers() {
     ...
   }
}
```

Updates for the DAO implementation

```
@Repository
public class CustomerDAOImpl implements CustomerDAO {

    @Autowired
    private SessionFactory sessionFactory;

    public List<Customer> getCustomers() {
      ...
    }
}
```

Revised Big Picture