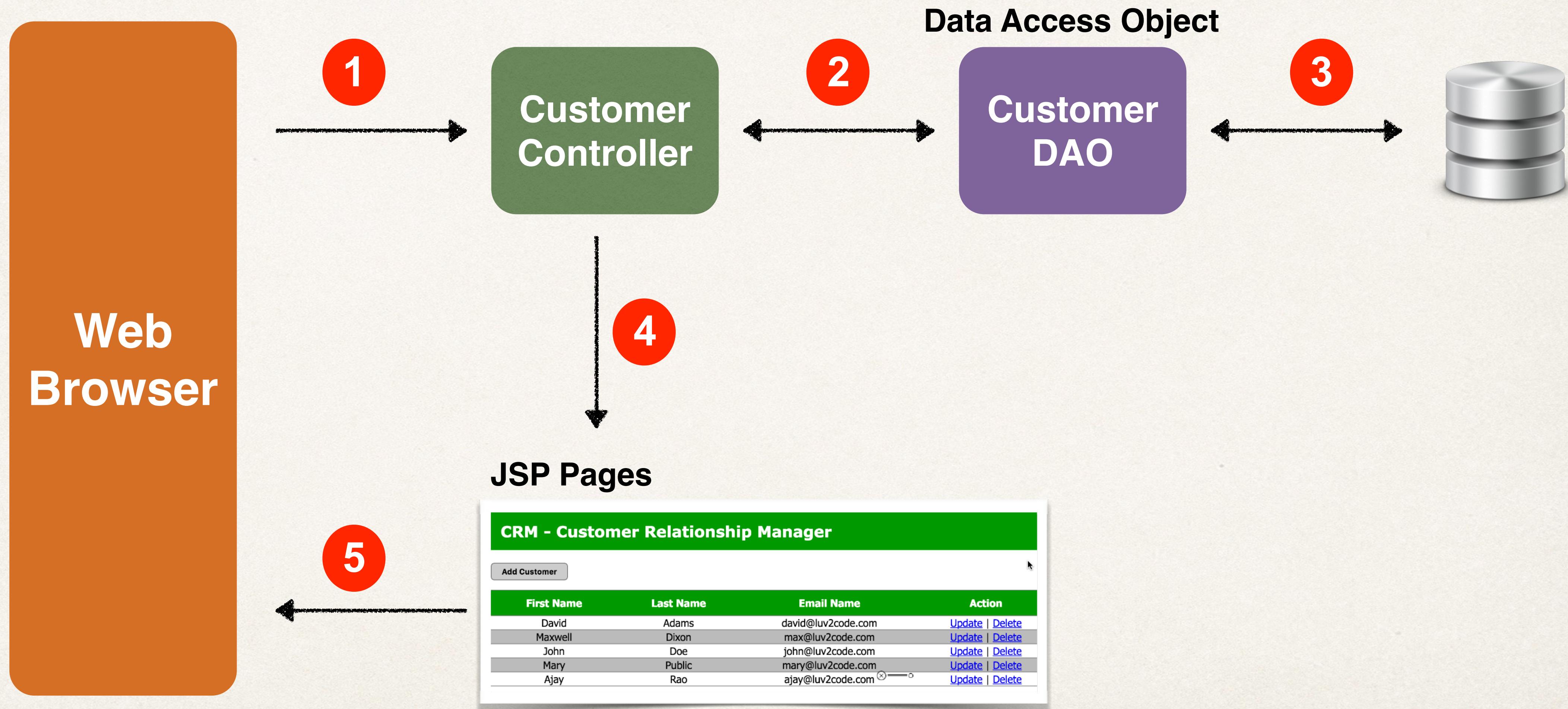


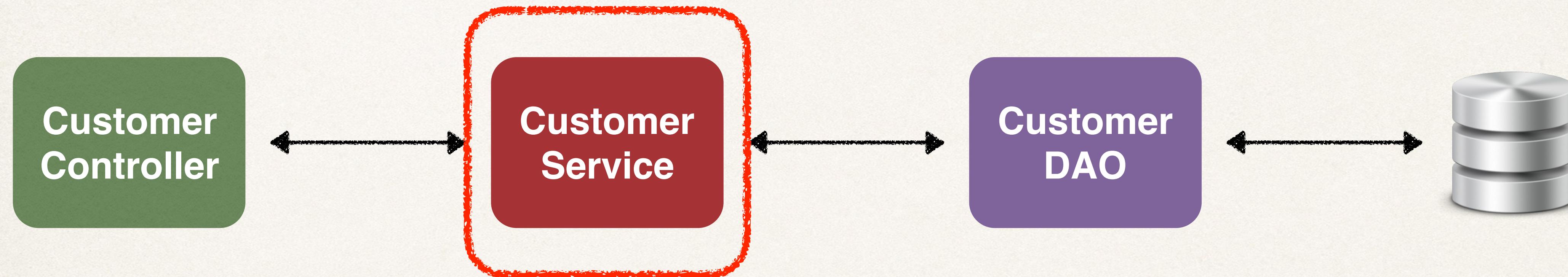
Define Services with @Service



Big Picture

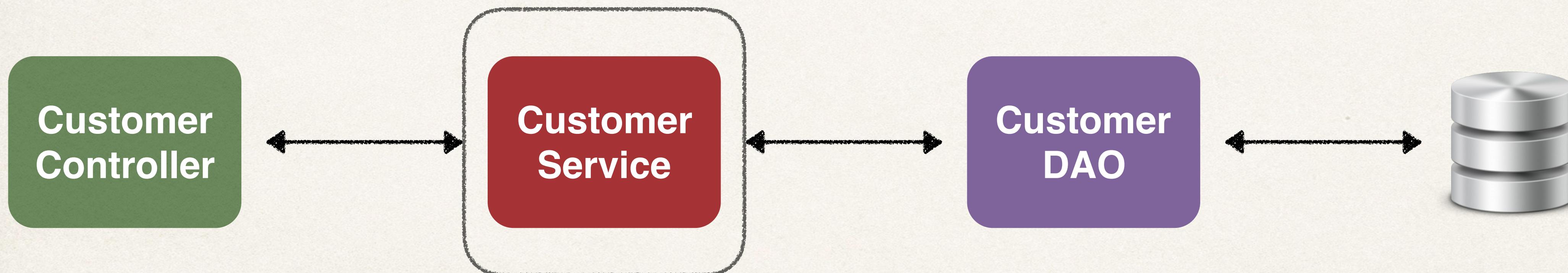


Refactor: Add a Service Layer

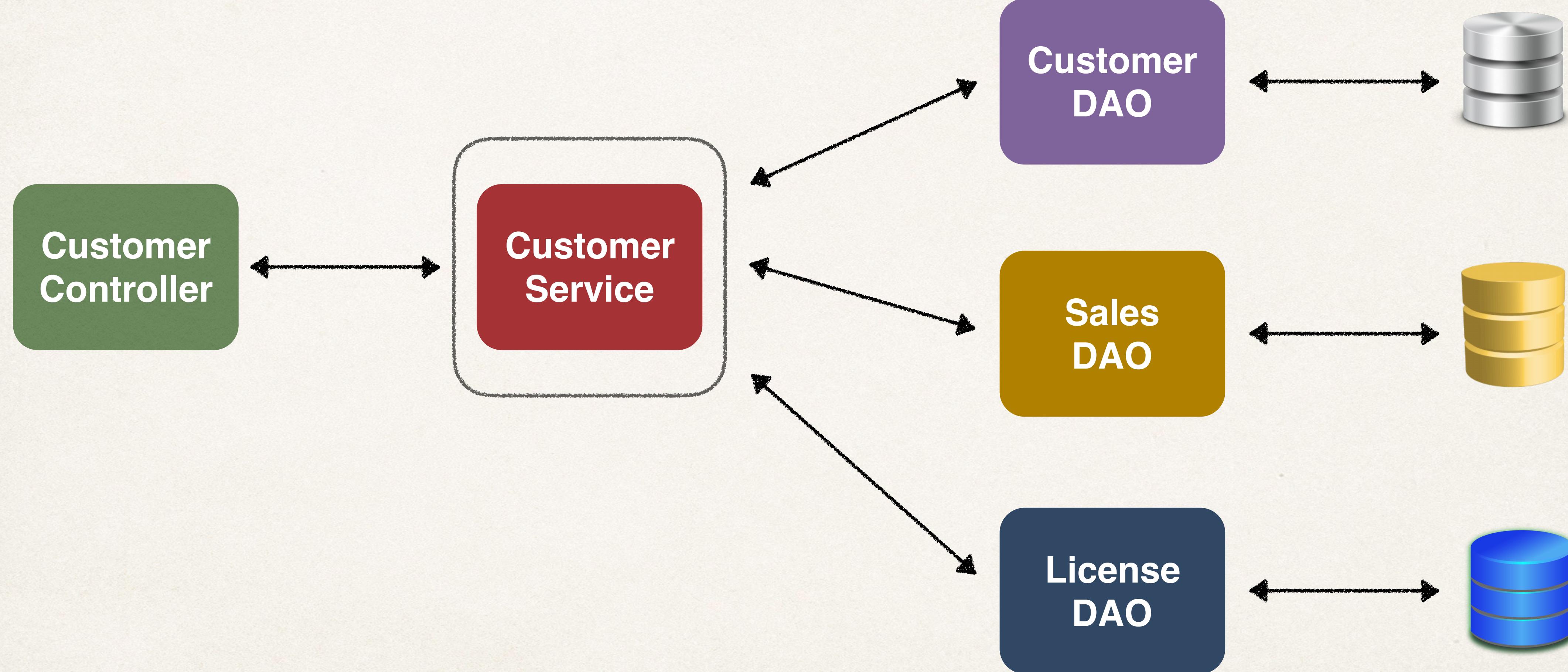


Purpose of Service Layer

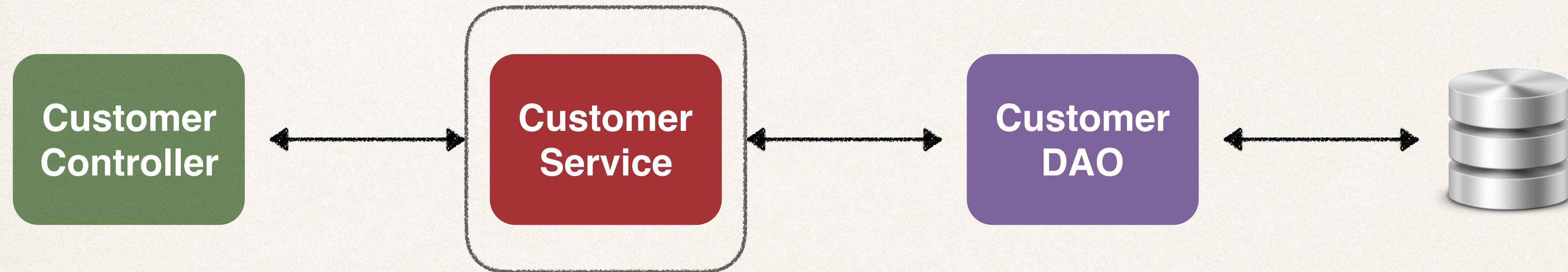
- ❖ **Service Facade** design pattern
- ❖ Intermediate layer for custom business logic
- ❖ Integrate data from multiple sources (DAO/repositories)



Integrate Multiple Data Sources

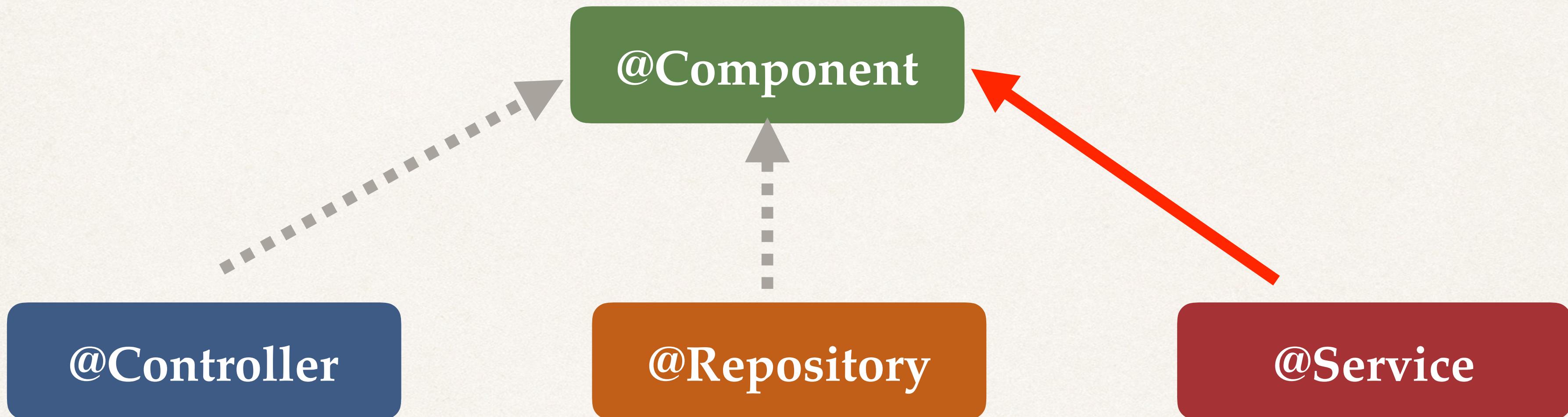


Most Times - Delegate Calls



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

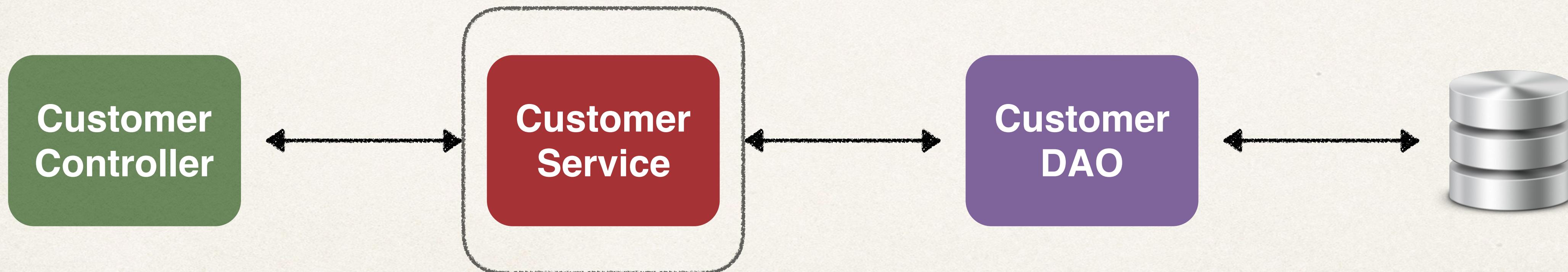
Customer Service

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

