CS544 EA

Spring:

Introduction to Spring

Spring Container

- The Spring project **started as a replacement** for the EJB container (version 2 and before).
  - Demonstrating that objects in a Enterprise container can be Plain Old Java Objects (POJOs)
    - A POJO class doesn’t implement or extend
    - Better separation of concerns / loose coupling
  - **Modern EJB (v3) containers copy Spring**

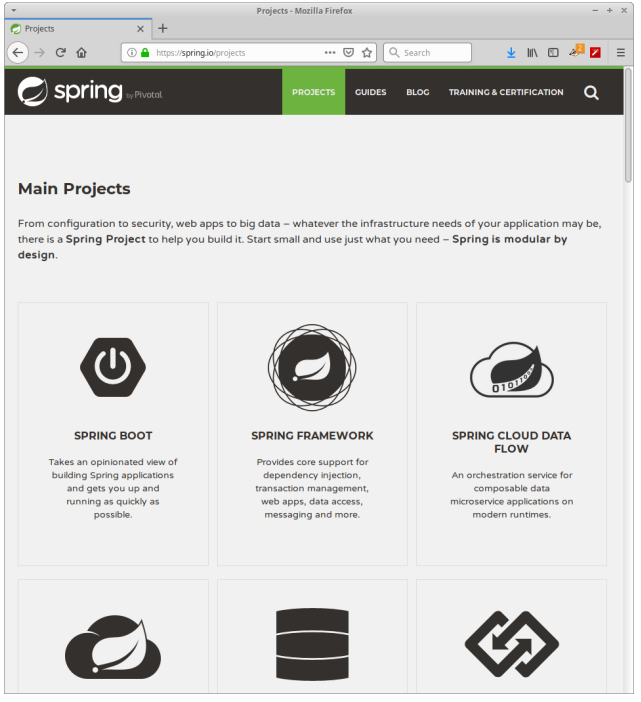
Spring Container

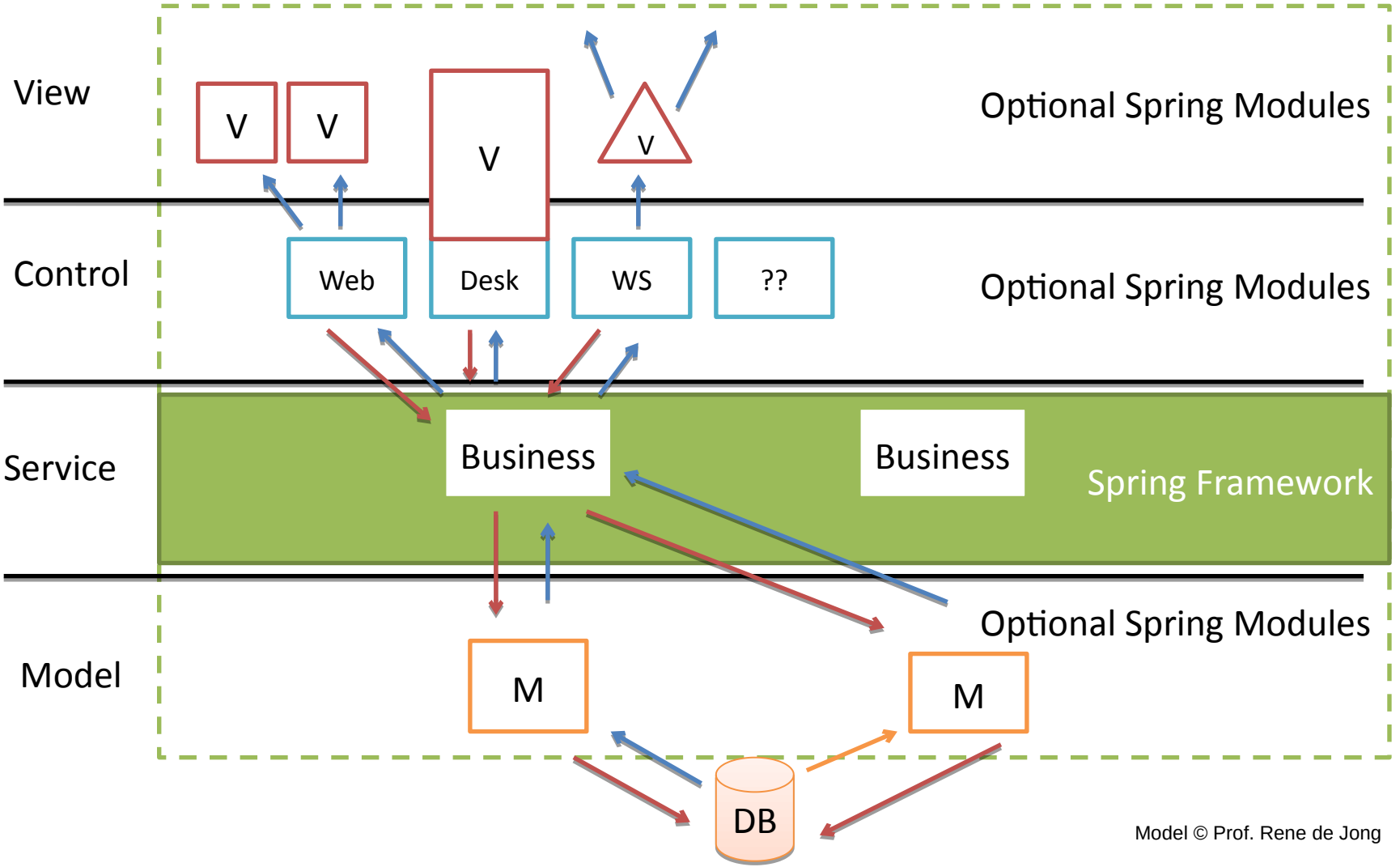
- Using IOC a container (like spring) can provide:
  - Dependency Injection
  - Aspected Oriented Programming

- The Spring Container gives the programmer **full control** over these features.
  - **Lots of features** & control compared to others

Spring Projects

- Beyond being a container there are also many other Spring related Projects
  - A Java Enterprise **Eco System**
- While JavaEE is the official Java standard
  - Spring and its related projects are the **DeFacto standard**





Model © Prof. Rene de Jong

Understanding Spring & DI

- **4 ways** connect objects together
  - Instantiate Objects Directly
  - P2I for more flexibility, but still instantiate
  - Use a factory object to instantiate
  - Use Spring and DI

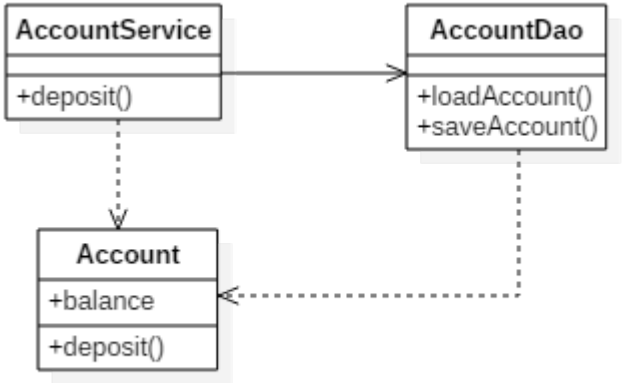
Instantiate Objects Directly

```
public class AccountService {
  private AccountDAO accountDAO;

  public AccountService() {
    accountDAO = new AccountDAO();
  }

  public void deposit(long accountNumber, double amount){
    Account account=accountDAO.loadAccount(accountNumber);
    account.deposit(amount);
    accountDAO.saveAccount(account);
  }
}
```

Hardcoded



- Relation between AccountService and AccountDao is hardcoded
  - To Change AccountDao implementation have to change the code

Use an Interface

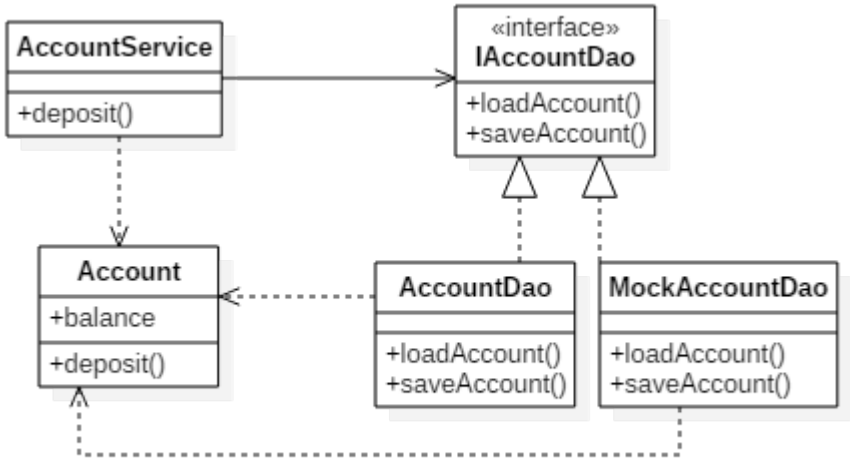
Flexibility

```
public class AccountService {
  private IAccountDAO accountDAO;

  public AccountService() {
    accountDAO = new AccountDAO();
  }

  public void deposit(long accountNumber, double amount){
    Account account=accountDAO.loadAccount(accountNumber);
    account.deposit(amount);
    accountDAO.saveAccount(account);
  }
}
```

Hardcoded



- By using an interface (P2I) we've gained the flexibility (two implementations)
  - But the relationship is still hard coded
  - To switch, we still have to change code

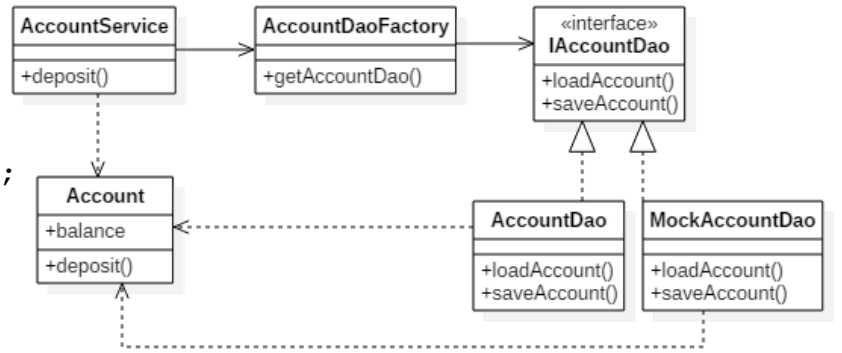
Use a Factory

```
public class AccountService {
  private IAccountDAO accountDAO;

  public AccountService() {
    AccountDAOFactory daoFactory = new AccountDAOFactory();
    accountDAO = daoFactory.getAccountDAO();
  }

  public void deposit(long accountNumber, double amount){
    Account account=accountDAO.loadAccount(accountNumber);
    account.deposit(amount);
    accountDAO.saveAccount(account);
  }
}
```

Hardcoded



- The relation between AccountService and AccountDao is still hardcoded
  - We have more flexibility, but if you want to change the AccountDao implementation you have to change the code in the AccountDaoFactory

Spring Dependency Injection

```
public class AccountService {
  private IAccountDAO accountDAO;

  public void setAccountDAO(IAccountDAO accountDAO) {
    this.accountDAO = accountDAO;
  }

  public void deposit(long accountNumber, double amount){
    Account account=accountDAO.loadAccount(accountNumber);
    account.deposit(amount);
    accountDAO.saveAccount(account);
  }
}
```

Setter for Injection

Config for creating and Injecting

```
<bean id="accountService" class="AccountService">
  <property name="accountDAO" ref="accountDAO" />
</bean>
<bean id="accountDAO" class="AccountDAO" />
<bean id="mockAccountDAO" class="MockAccountDAO" />
```

- What if the Factory created both Account Service and the AccountDao?
- What if the Factory could read a config file?
- That's in essence what Spring does

Summary

- Spring is a container for the service layer
  - Started as a replacement EJB container
  - Focuses on POJO based (flexible, best practices)

- In essence spring is a **fancy factory** that:
  - reads a config file, creates objects, and connects them

Science of Conciousness

- The whole is greater than the sum of the parts, since it matters just as much how the parts are connected