Лекция 23

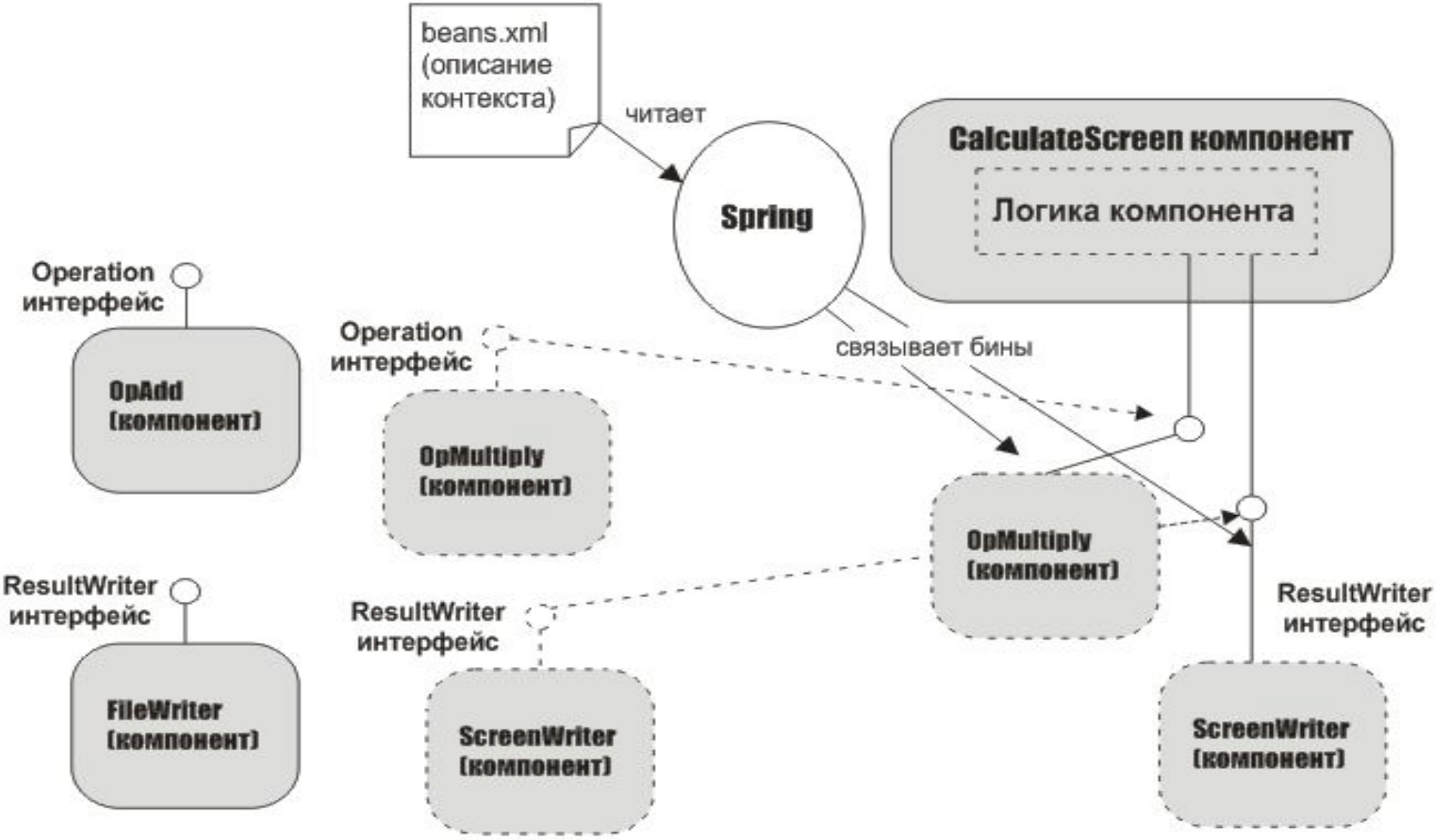
Spring

Рассмотрим создание приложения, используя Spring Framework для сборки многоразовых компонентов, затем соединяя их в одно приложение.

Это действие по соединению компонентов называется связывание (wiring).

Рассмотрим простое приложение по сложению двух чисел и выводу результата с использованием Spring.

Рисунок графически показывает, как Spring может гибко менять реализацию математической операции:



CalculateSpring – это основной класс, который на прямую не инициализирует поля класса.

Вместо этого эта задача выполняется Spring контейнером.

Spring контейнер считывает конфигурационный файл beans.xml, инициализирует бины и затем связывает их соответственно конфигурационной информации, содержащейся в beans.xml.

Класс CalculateSpring имеет вид:

```
import org.springframework.beans.factory.BeanFactory;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class CalculateSpring {
  private Operation ops;
  private ResultWriter wtr;
  public void setOps(Operation ops) { this.ops = ops; }
  public void setWriter(ResultWriter writer) { this.wtr = writer; }
  public static void main(String[] args) {
    ApplicationContext context =
              new ClassPathXmlApplicationContext("beans.xml");
    BeanFactory factory = (BeanFactory) context;
    CalculateSpring calc =(CalculateSpring) factory.getBean("opsbean");
    calc.execute(args);
  }
  public void execute(String [] args) {
    long op1 = Long.parseLong(args[0]);
    long op2 = Long.parseLong(args[1]);
    wtr.showResult("The result of " + op1 + ops.getOpsName() + op2 + " is " +
                                                    ops.operate(op1, op2) + "!");
  }
}
```

Суть в том, что CalculateSpring не сам работает с экземплярами Operation или ResultWriter, а делегирует эту задачу Spring контейнеру.

Spring контейнер, в свою очередь, читает конфигурационный файл и вызывает бин файл дескриптор.

Для этого необходимо создать контекст, затем фабрику и при помощи фабрики достать бин.

ApplicationContext в Spring это тип BeanFactory.

BeanFactory дает возможность доступа к JavaBeans которые инициализированы, связанные и управляемые Spring контейнером.

Хотя есть и другие BeanFactory классы в Spring, ApplicationContext класс намного чаще используется, так как он снабжает нас несколькими ценными особенностями – включение поддержки для интернационализации, загрузка ресурсов, интеграция с внешними иерархиями контекстов и много чего еще.

Конструктор ClassPathXmlApplicationContext берет в качестве аргумента файл описания контекста или bean wiring файл.

Этот файл называется beans.xml.

Файл beans.xml – это конфигурационный файл, описывающий, как связывать объекты вместе. Здесь показан beans.xml файл:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">
  <bean id="screen" class="ScreenWriter" />
  <bean id="multiply" class="OpMultiply" />
  <bean id="add" class="OpAdd" />
  <bean id="opsbean" class="CalculateSpring">
    <property name="ops" ref="multiply" />
    <property name="writer" ref="screen"/>
  </bean>
</beans>
```

ClassPathXmlApplicationContext – это часть Spring контейнера, и ищет контекстное описание (beans.xml) в Java VM CLASSPATH и создает из него экземпляр ApplicationContext.

Spring контейнер связывает бины в ходе инициализации ApplicationContext.

Связывание Java Beans

- Создаем экземпляр ScreenWriter и именуем bean, как screen
- Создаем экземпляр OpMultiply именуем bean, как multiply
- Создаем экземпляр OpAdd именуем bean, как add
- Создаем экземпляр CalculateSpring именуем bean, как opsbean
- Устанавливаем ссылку свойства ops opsbean бина, к бину с именем multiply
- Устанавливаем ссылку свойства writer opsbean бина, к бину с именем screen