Spring

Beans

Beans are just another word for a Java object.

But, there's a convention that goes with it.

```
public class Customer {
    private String userId;
    private String password;

    public String getUserId() {
        return userId;
    }

    public void setUserId(String userId) {
        this.userId = userId;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

Spring Beans

Spring beans are just beans that Spring manages.

Spring does many things, but at it's core, it's all about managing beans.

Spring Context

This is an object that returns objects for Spring.

```
ApplicationContext context = new ClassPathXmlApplicationContext("context.xml");

Customer alycit = (Customer) context.getBean("alycit");
Customer guyroyse = (Customer) context.getBean("guyroyse");
```

Defining Beans

Configured via an XML file.

```
<?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-3.0.xsd">

     <bean id="alycit" class="com.bfbi.Customer" />
     <bean id="guyroyse" class="com.bfbi.Customer" />

</beans>
```

Empty Properties

Properties are not set.

```
ApplicationContext context = new ClassPathXmlApplicationContext("context.xml");

Customer alycit = (Customer) context.getBean("alycit");
Customer guyroyse = (Customer) context.getBean("guyroyse");

assertNull(alycit.getUserId());       // true
assertNull(alycit.getPassword());     // true

assertNull(guyroyse.getUserId());     // true
assertNull(guyroyse.getPassword());   // true
```

Adding Properties

```
<beans>

      <bean id="alycit" class="com.bfbi.Customer">
            <property name="userId" value="adiaz" />
            <property name="password" value="s3cr3t" />
      </bean>

      <bean id="guyroyse" class="com.bfbi.Customer">
            <property name="userId" value="groyse" />
            <property name="password" value="sup3r_s3cr3t" />
      </bean>

</beans>
```

Properties Added

Properties now have values.

```
ApplicationContext context = new ClassPathXmlApplicationContext("context.xml");

Customer alycit = (Customer) context.getBean("alycit");
Customer guyroyse = (Customer) context.getBean("guyroyse");

assertEquals("adiaz", alycit.getUserId());                  // true
assertEquals("s3cr3t", alycit.getPassword());               // true

assertEquals("groyse", guyroyse.getUserId());               // true
assertEquals("sup3r_s3cr3t", guyroyse.getPassword());       // true
```

You Can Also Add Objects

```
<beans>

      <bean id="jointAccount" class="com.bfbi.CheckingAccount" />

      <bean id="alycit" class="com.bfbi.Customer">
            <property name="userId" value="adiaz" />
            <property name="password" value="s3cr3t" />
            <property name="account" ref="jointAccount" />
      </bean>

      <bean id="guyroyse" class="com.bfbi.Customer">
            <property name="userId" value="groyse" />
            <property name="password" value="sup3r_s3cr3t" />
            <property name="account" ref="jointAccount" />
      </bean>

</beans>
```

Objects Added

The same object has been added to both beans.

```
ApplicationContext context = new ClassPathXmlApplicationContext("context.xml");

Customer alycit = (Customer) context.getBean("alycit");
Customer guyroyse = (Customer) context.getBean("guyroyse");

assertSame(alycit.getAccount(), guyroyse.getAccount());     // true
```

You Can Also Add Collections

```
<beans>

      <bean id="jointAccount" class="com.bfbi.CheckingAccount" />
      <bean id="secretAccount" class="com.bfbi.SavingsAccount" />

      <bean id="alycit" class="com.bfbi.Customer">
            <property name="accounts"><list>
                  <ref bean="jointAccount" />
            </list></property>
      </bean>

      <bean id="guyroyse" class="com.bfbi.Customer">
            <property name="accounts"><list>
                  <ref bean="jointAccount" />
                  <ref bean="secretAccount" />
            </list></property>
      </bean>

</beans>
```

Collections Added

Works for Lists, Sets, and Maps.

```
ApplicationContext context = new ClassPathXmlApplicationContext("context.xml");

Customer alycit = (Customer) context.getBean("alycit");
Customer guyroyse = (Customer) context.getBean("guyroyse");

assertSame(alycit.getAccounts().get(0), guyroyse.getAccounts().get(0));     // true
assertNotNull(guyroyse.getAccounts().get(1));                               // true
```

Works for Lists, Sets, and Maps

```
<beans>

      <bean id="jointAccount" class="com.bfbi.CheckingAccount" />
      <bean id="secretAccount" class="com.bfbi.SavingsAccount" />

      <bean id="alycit" class="com.bfbi.Customer">
            <property name="accounts"><set>
                  <ref bean="jointAccount" />
            </set></property>
      </bean>

      <bean id="guyroyse" class="com.bfbi.Customer">
            <property name="accounts"><map>
                  <entry key="shared" value-ref="jointAccount" />
                  <entry key="secret" value-ref="secretAccount" />
            </map></property>
      </bean>

</beans>
```

So what?

Why Spring?

Provides a layer of isolation between a class and it's dependencies.

This makes testing it easier.

Vending Machine Demo