Chapter

4

SECTION I: ALL ABOUT THE SPRING FRAMEWORK

Getting Started

Learning is best accomplished by doing. This chapter demonstrates a simple application that shows how application dependencies are injected through the Spring IoC container.

About The Application

This application, based on the hour of the day, indicates whether the customer to the restaurant is eligible for a happy hour meal or a standard meal.

This application is made up of the following**:**

- Domain
    - Customer
- Interfaces
    - HappyHourService

    - StandardService
- Implementations
    - HappyHourServiceImpl
    - StandardServiceImpl
- Beans
    - WelcomeUserBean
- Client
    - RunApp

All of these are wired using welcomeUser.xml.

Creating A Java Application

This book uses NetBeans as the IDE of choice. Use it to create a new Project called **WelcomeUser**.

REMINDER

Refer to Appendix A for steps on downloading and installing NetBeans IDE.

Select **File → New Project**, as shown in diagram 4.1.1, to create a new **Java Project**.

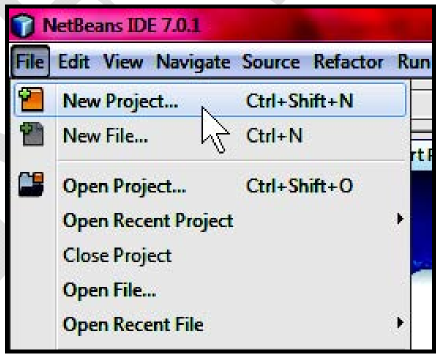

NetBeans IDE 7.0.1
File Edit View Navigate Source Refactor Run
New Project... Ctrl+Shift+N
New File... Ctrl+N
Open Project... Ctrl+Shift+O
Open Recent Project
Close Project
Open File...
Open Recent File

Diagram 4.1.1: Creating New Project

New Project dialog box appears as shown in diagram 4.1.2.

Enter the name of the Java application as **WelcomeUser**, uncheck the **Create Main Class** option and check the **Use Dedicated Folder for Storing Libraries** option as shown in the diagram 4.1.3.

Click Finish. **WelcomeUser** application is created in the NetBeans IDE.

Downloading And Extracting The Spring Framework

The Spring framework is available for download at**:**
http://www.springsource.org/download

At the time of writing this book the latest production release was**:**
Spring Framework 3.1.1.RELEASE (requires Java 1.5+)

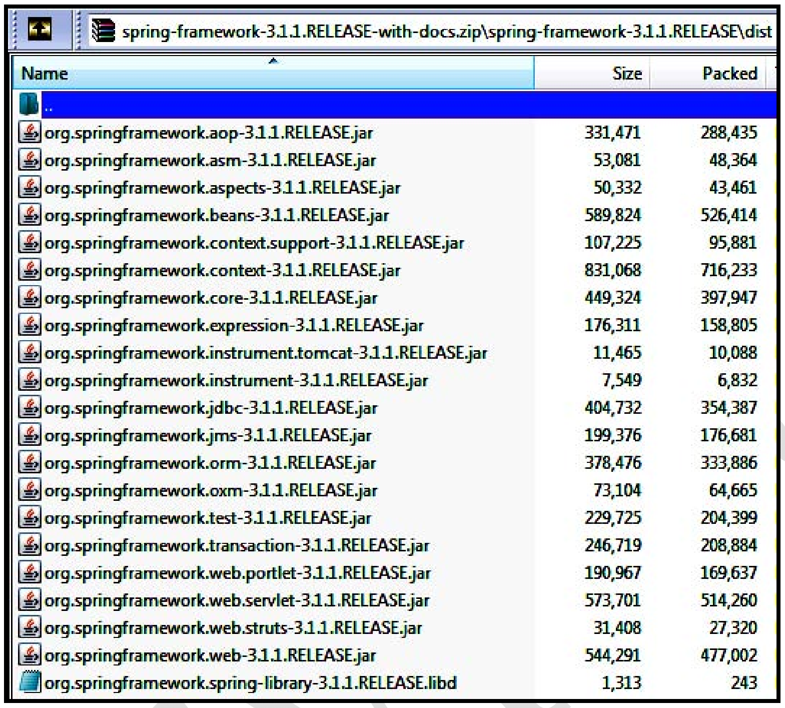
The file that was downloaded was**:**
spring-framework-3.1.1.RELEASE-with-docs.zip (sha1) 51.6 MB

This file is also available in this book's accompanying CDROM.

Download the file and extract the file contents to a location of choice.

About The Downloaded Files

The Spring Framework is composed of several distinct modules. After the Spring framework is downloaded, unzip the Spring framework distribution. It holds **20 different JAR** files in the **dist** directory, as shown in diagram 4.2.



spring-framework-3.1.1.RELEASE-with-docs.zip\spring-framework-3.1.1.RELEASE\dist

Name	Size	Packed
..		
org.springframework.aop-3.1.1.RELEASE.jar	331,471	288,435
org.springframework.asm-3.1.1.RELEASE.jar	53,081	48,364
org.springframework.aspects-3.1.1.RELEASE.jar	50,332	43,461
org.springframework.beans-3.1.1.RELEASE.jar	589,824	526,414
org.springframework.context.support-3.1.1.RELEASE.jar	107,225	95,881
org.springframework.context-3.1.1.RELEASE.jar	831,068	716,233
org.springframework.core-3.1.1.RELEASE.jar	449,324	397,947
org.springframework.expression-3.1.1.RELEASE.jar	176,311	158,805
org.springframework.instrument.tomcat-3.1.1.RELEASE.jar	11,465	10,088
org.springframework.instrument-3.1.1.RELEASE.jar	7,549	6,832
org.springframework.jdbc-3.1.1.RELEASE.jar	404,732	354,387
org.springframework.jms-3.1.1.RELEASE.jar	199,376	176,681
org.springframework.orm-3.1.1.RELEASE.jar	378,476	333,886
org.springframework.oxm-3.1.1.RELEASE.jar	73,104	64,665
org.springframework.test-3.1.1.RELEASE.jar	229,725	204,399
org.springframework.transaction-3.1.1.RELEASE.jar	246,719	208,884
org.springframework.web.portlet-3.1.1.RELEASE.jar	190,967	169,637
org.springframework.web.servlet-3.1.1.RELEASE.jar	573,701	514,260
org.springframework.web.struts-3.1.1.RELEASE.jar	31,408	27,320
org.springframework.web-3.1.1.RELEASE.jar	544,291	477,002
org.springframework.spring-library-3.1.1.RELEASE.libd	1,313	243

Diagram 4.2: Spring modules (**.**jar files)

Adding The Spring Libraries To The Project

Since this application is going to be built using the Spring framework, add the Spring framework library files to this Java application.

Since while creating this Java application, the option **Use Dedicated Folder For Storing Libraries** was checked, NetBeans automatically creates the **lib** directory in the WelcomeUser application.

Now add the Spring Framework libraries in NetBeans.

Right-click on the **WelcomeUser** application, select **Properties** as shown in diagram 4.3.1.

Click **Add JAR/Folder**. This displays the dialog box that allows choosing the JAR files.

Browse to the directory that holds the downloaded files and select the following .jar files:

- From <Drive>:\spring-framework-3.1.1.RELEASE\dist:
  - org.springframework.asm-3.1.1.RELEASE.jar
  - org.springframework.beans-3.1.1.RELEASE.jar
  - org.springframework.context.support-3.1.1.RELEASE.jar
  - org.springframework.context-3.1.1.RELEASE.jar
  - org.springframework.core-3.1.1.RELEASE.jar
  - org.springframework.expression-3.1.1.RELEASE.jar
- From <Drive>:\spring-framework-3.1.1.RELEASE\projects\spring-build\lib\ivy
  - commons-logging.jar

After adding all the JAR files, click **OK**. All the libraries are added to the project.

Building The Code Spec

Now that the project is created by the NetBeans IDE, let's begin extending the project according to the specifications. Let's begin with the domain class.

Domain [Customer.java]

Customer.java needs to be created as a simple bean to hold information for each request.

To create this class, right-click the **WelcomeUser** project and select **New → Java Class...** as shown in diagram 4.4.1.

Spring is a container-based framework. The container needs to be configured such that it knows what beans it should contain and how to wire those beans so that they can work together. The act of creating associations between application objects is the essence of dependency injection and is commonly referred to as wiring.

As of Spring 3.0, there are three ways to configure beans in the Spring container:

- One or more XML files [Traditional]
- A Java-based configuration option [Spring 3.0+]
- Annotations [Spring 2.5+]

For simplicity, this application focuses on the traditional XML option for now. Spring's Java-based configuration and the annotation based configuration will be covered later in this book.

Create a Spring XML configuration file named welcomeUser.xml. To create the configuration file, right-click the **WelcomeUser** application and select **New → Others...** as shown in diagram 4.6.1.

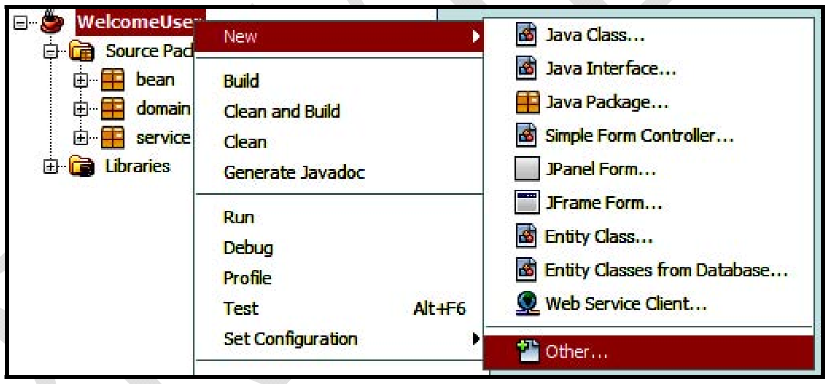


Diagram 4.6.1: Selecting the Other... option

New File dialog box appears, as shown in diagram 4.6.2. Select **Other** available under the **Categories** list and **Spring XML Configuration File** available under the **File types**, as shown in diagram 4.6.2.

A typical Spring configuration XML file looks like this:

```
<?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-3.1.xsd">

  <bean id="..." class="...">
    <!-- collaborators and configuration for this bean go here -->
  </bean>

  <!-- more bean definitions here -->

</beans>
```

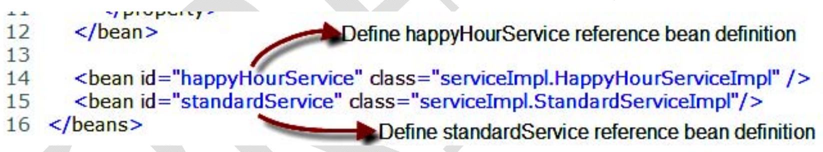
Within <beans> all of the Spring configuration is placed, including <bean> declarations.

The **id** attribute is a string used to identify the individual bean definition. The **class** attribute defines the type of the bean and uses the fully qualified class name. The value of the id attribute refers to collaborating objects.

The beans namespace is not the only Spring namespace. The core Spring Framework comes with ten configuration namespaces, all of these are explained later in this book in *Chapter 05: Using XML*.

In this application:

- ❑ All the setter methods are implemented by Spring configuration beans



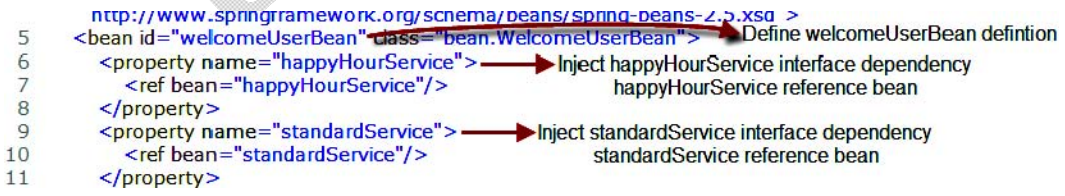
```
    </bean>

    <bean id="happyHourService" class="serviceImpl.HappyHourServiceImpl" />
    <bean id="standardService" class="serviceImpl.StandardServiceImpl"/>
</beans>
```

Define happyHourService reference bean definition
Define standardService reference bean definition

- ❑ In Spring, objects are not responsible for finding or creating the other objects that they need to do their jobs. Instead, they are given references to the objects that they collaborate with by the container

  All the dependencies i.e. the two interfaces, are injected by the Spring framework using these beans.



```
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">
    <bean id="welcomeUserBean" class="bean.WelcomeUserBean">
       <property name="happyHourService">
          <ref bean="happyHourService"/>
       </property>
       <property name="standardService">
          <ref bean="standardService"/>
       </property>
```

Define welcomeUserBean defintion
Inject happyHourService interface dependency happyHourService reference bean
Inject standardService interface dependency standardService reference bean

```
WelcomeUserBean welcomeUserBean = (WelcomeUserBean)
appContext.getBean("welcomeUserBean");
```

With a reference to the **WelcomeUserBean** object, **welcomeUser()** is invoked to welcome the customer.

```
Customer cust = new Customer();
cust.setFirstName("Sharanam");
cust.setLastName("Shah");
String str = welcomeUserBean.welcomeUser(cust);
```

REMINDER

Note that this class knows nothing about which type of welcome [**happy hour** or **standard**] is required. All of this is taken care of by **welcomeUser.xml** which knows what the implementations are.

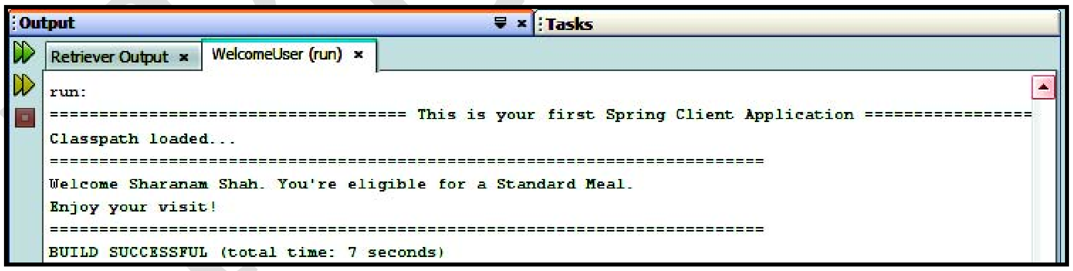
Running The Application

Compiling And Running The Project

Compile and build the application by right clicking and choosing **Clean and Build**.

Right click the **WelcomeUser** project and select **Run**.

Once the **Run** processing completes in NetBeans IDE, the WelcomeUser application is executed, in the console, as shown in diagram 4.7.1.



```
run:
================================== This is your first Spring Client Application ================
Classpath loaded...
====================================================================
Welcome Sharanam Shah. You're eligible for a Standard Meal.
Enjoy your visit!
====================================================================
BUILD SUCCESSFUL (total time: 7 seconds)
```

Diagram 4.7.1: The WelcomeUser application

The output shows that the customer is welcomed with a standard meal message.

Summary

This chapter has shown how to develop an application using the Spring framework. It demonstrated the complete application development using NetBeans IDE.

It looked into wiring beans together within the Spring container. Wiring is typically performed within a Spring container using an XML file. This XML file contains configuration information for all of the components of an application, along with information that helps the container perform DI to associate beans with other beans that they depend on.

The next section deals with the details of configuring beans using XML, annotations and Java based configuration.

The Book CDROM holds the complete application source code along with the required libraries built using the NetBeans IDE for the following examples. The application simply needs to be opened in NetBeans and then compiled, build and executed.

- Code/Section 1/Chapter04_Cds/**WelcomeUser**