To ask Spring Data to create an implementation of SpitterRepository, you need

to add a single element to your Spring configuration. The following listing shows the

XML configuration needed to put Spring Data JPA into motion.

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:jpa="http://www.springframework.org/schema/data/jpa"

xsi:schemaLocation="http://www.springframework.org/schema/data/jpa

<http://www.springframework.org/schema/data/jpa/spring-jpa-1.0.xsd>">

<jpa:repositories base-package="com.habuma.spittr.db" />

...

</beans>

The <jpa:repositories> element holds all the magic of Spring Data JPA. Much

like the <context:component-scan> element, <jpa:repositories> is given a basepackage

to scan. But where <context:component-scan> scans a package (and its subpackages)

for classes that are annotated with @Component, <jpa:repositories> scans

its base package for any interfaces that extend Spring Data JPA’s Repository interface.

When it finds any interface extending Repository, it automatically (at application

startup time) generates an implementation of that interface.

Instead of using the <jpa:repositories> element, you can use @EnableJpa-

Repositories in your Java configuration class. What follows is a Java configuration class

annotated with @EnableJpaRepositories to scan the com.habuma.spittr.db package:

@Configuration

@EnableJpaRepositories(basePackages="com.habuma.spittr.db")

public class JpaConfiguration {

...

}

Listing 11.4 Creating a repository from an interface definition with Spring Data

**Spring In Action 11.3.0Listing** 11.5 Configuring Spring Data JPA