Classpath scanning for managed components

Summary

Description

In most of examples before this chapter, XML was used for specifying configuration metadata to create BeanDefinition in the Spring Container. The previous section (Annotation-based configuration) demonstrated the possibility of providing a considerable amount of the configuration metadata using source-level annotations. Even in those examples however, the "base" bean definitions were explicitly defined in the XML file while the annotations were driving the dependency injection only. The current section introduces an option for implicitly detecting the *candidate components* by scanning the classpath and matching against *filters*.

@Component and further stereotype annotations

From Spring2.0, @Repository annotation is introduced to indicate repositories such as Data Access Object (DAO). In Spring 2.5, @Component, @Service and @Controller annotation are added. @Component, @Service and @Controller. @Component serves as a generic stereotype for any Spring-managed component; whereas, @Repository, @Service, and @Controller serve as specializations of @Component for more specific use cases (e.g., in the persistence, service, and presentation layers, respectively).

Auto-detection components

Spring provides the function to automatically detect 'stereotyped' classes and register BeanDefinition that corresponds with the ApplicationContext.

```
@Service
public class SimpleMovieLister {

    private MovieFinder movieFinder;

    @Autowired
    public SimpleMovieLister(MovieFinder movieFinder) {
        this.movieFinder = movieFinder;
    }
}
@Repository
public class JpaMovieFinder implements MovieFinder {
    // implementation elided for clarity
}
```

To autodetect these classes and register the corresponding beans requires the inclusion of the following element in XML where 'basePackage' would be a common parent package for the two classes (or alternatively a comma-separated list could be specified that included the parent package of each class).

```
<?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:context="http://www.springframework.org/schema/context"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
            http://www.springframework.org/schema/context
            http://www.springframework.org/schema/context/spring-context-2.5.xsd">

     <context:component-scan base-package="org.example"/>

</beans>
```

Naming autodetected components

When a component is autodetected in scanning, the bean name is generated by the BeanNameGenerator strategy known to that scanner. By default, any Spring 'stereotype' annotation (@Component, @Repository, @Service, and @Controller) that contains a name value will thereby provide that name to the corresponding bean definition. If such an annotation contains no name value or for any other detected component (such as those discovered due to custom filters), the default bean name generator will return the uncapitalized non-qualified class name. For example, if the following two components were detected, the names would be 'myMovieLister' and 'movieFinderImpl':

```
@Service("myMovieLister")
public class SimpleMovieLister {
    // ...
}
@Repository
public class MovieFinderImpl implements MovieFinder {
    // ...
}
```

Providing a scope for autodetected components

Generally, the Spring management component is a ‘singleton’. There are cases that require other scopes. Spring2.5 provides @Scope annotation.

```
@Scope("prototype")
@Repository
public class MovieFinderImpl implements MovieFinder {
    // ...
}
```

Providing qualifier metadata with annotations

This section explains how to use @Qualifier annotation to provide detailed control of searching autowiring.

```
@Component
@Qualifier("Action")
public class ActionMovieCatalog implements MovieCatalog {
    // ...
}
```

Reference

- Spring Framework - Reference Document / 3.12. Classpath scanning for managed components