1.2. 다양한 빈 컨테이너 설정

Collection 설정방법

Example.java

```
package com.example.demo.collection;

import java.util.List;
import java.util.Map;
import java.util.Properties;
import java.util.Set;

import lombok.Data;

@Data
public class Example {
    private Set<Object> set;
    private Map<String, Object> map;
    private List<Object> list;
    private Properties prop;
}
```

collection-config.xml

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

    <!-- String who = new String("홍길동"); -->
    <bean id="who" class="java.lang.String">
        <constructor-arg value="홍길동"/>
    </bean>

    <util:list id="myList" list-class="java.util.ArrayList">
        <value>111</value>
        <value>222</value>
        <value>111</value>
        <value>333</value>
        <ref bean="who"/>
    </util:list>
```

```
    <bean id="example" class="com.example.demo.collection.Example">
        <property name="list" ref="myList">
            <!-- <list>
                <value>111</value>
                <value>222</value>
                <value>111</value>
                <ref bean="who"/>
            </list> -->
        </property>
        <property name="map">
            <map>
                <entry key="봄">
                    <value>Spring</value>
                </entry>
                <entry key="여름">
                    <value>Summer</value>
                </entry>
                <entry key="who">
                    <ref bean="who"/>
                </entry>
            </map>
        </property>
        <property name="prop">
            <props>
                <prop key="봄">Spring</prop>
                <prop key="여름">Summer</prop>
            </props>
        </property>
        <property name="set">
            <set>
                <value>111</value>
                <value>222</value>
                <value>111</value>
                <ref bean="who"/>
            </set>
        </property>
    </bean>

</beans>
```

Test.java

```
package com.example.demo.collection;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test {

    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext(
                "com/example/demo/collection/collection-config.xml");

        Example example = context.getBean(Example.class);

        example.getList().forEach(System.out::println);
        System.out.println();

        example.getMap().forEach((key, value) -> {
            System.out.println(key + ":" + value);
        });
```

```
        System.out.println();

        example.getProp().forEach((key, value) -> {
            System.out.println(key + ":" + value);
        });
        System.out.println();

        example.getSet().forEach(System.out::println);
    }

}
```

SpEL을 이용한 설정방법

Spring 3.X에서 추가된 기능으로 SpEL을 이용하면 동적으로 표현식을 해석하고 그 결과를 ApplicationContext에서 사용할 수 있다. 결국 동적으로 생성된 값을 다른 자바 빈에 주입할 수 있다. #{빈아이디.멤버변수} 구문에 의해 getter가 호출되고 그 값이 주입된다.

User.java

```
package com.example.demo.etc;

import lombok.Data;

@Data
public class User {
    private String name;
    private int age;
}
```

Member.java

```
package com.example.demo.etc;

import lombok.Data;

@Data
public class Member {
    private String name;
    private int age;
}
```

etc-config.xml

```
<?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:c="http://www.springframework.org/schema/c"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:util="http://www.springframework.org/schema/util"
    xsi:schemaLocation="
    http://www.springframework.org/schema/beans
```

```
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/util
    http://www.springframework.org/schema/util/spring-util.xsd">

    <bean id="user" class="com.example.demo.etc.User">
        <property name="name" value="일지매">
            <!-- <value>홍길동</value> -->
        </property>
        <property name="age">
            <value>19</value>
        </property>
    </bean>

    <bean id="member" class="com.example.demo.etc.Member">
        <property name="name">
            <value>#{user.name + "님"}</value>
        </property>
        <property name="age">
            <value>#{user.age + 1}</value>
        </property>
    </bean>

    <context:component-scan base-package="com.example.demo.etc"/>
    <!-- <context:property-placeholder location="my.properties"/> -->

</beans>
```

`@Value` 애노테이션은 Spring의 빈의 필드에 값 자체를 주입하는 데 사용할 수 있으며 멤버필드 또는 생성자, 메소드, 매개 변수 수준에서 적용 할 수 있다.

`@PropertySource` 애노테이션은 Spring 3.1 부터 생기기 시작한 통합 프로퍼티 관리 시스템으로 시스템 프로퍼티, 환경변수, JNDI 등을 모두 하나의 공간에 넣고 그 값을 읽고 설정할 수 있게 해준다. <context:property-placeholder location="classpath:app.properties"/> 태그로 대체할 수 있다.

Person.java

```
package com.example.demo.etc;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.PropertySource;
import org.springframework.stereotype.Component;

import lombok.Data;

//<context:property-placeholder location="my.properties"/>
@PropertySource(value = { "my.properties" })

@Component
@Data
public class Person {

    @Value("#{member.name}")
    private String name;
    @Value("#{member.age}")
    private int age;

    @Value("${car.default.name:null}")
    private String carName;
```

```
    @Value("${car.default.doors:0}")
    private int carDoors;

    // Run As > Run Configuration > VM arguments > -Duser.region=KR
    @Value("#{systemProperties['user.region'] == null ? 'US' : systemProperties['user.region']}")
    private String defaultLocale;
}
```

Test.java

```
package com.example.demo.etc;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test {

    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext(
                "com/example/demo/etc/etc-config.xml");

        User user = context.getBean("user", User.class);
        System.out.println(user);

        Member member = context.getBean("member", Member.class);
        System.out.println(member);

        Person person = context.getBean("person", Person.class);
        System.out.println(person);
    }

}
```