Arrays and Collections As Beans

Spring XML based configuration defined elements like <list/>, <set/>, <map/>, and <props/>. The purpose of these elements was to map XML based configuration to Java arrays and collection. Since the addition of JavaConfig, we don't require that kind of custom mappings because new configuration is itself in Java language, so we can directly define beans of Array and Collections.

Examples

Bean of Array

```
package com.piseth.java.school;

@Configuration

public class ArraysAsBeanExample {


    @Bean

    public String[] fruits() {

        return new String[]{"apple", "banana", "orange"};

    }


    @Bean

    public TestBean testBean(){

        return new TestBean();

    }
    private static class TestBean {
        @Autowired
        private String[] fruits;

        @PostConstruct
        public void postConstruct() {
            System.out.println(Arrays.toString(fruits));
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                ArraysAsBeanExample.class);
    }
}
```

Output

```
[apple, banana, orange]
```

Bean of List

```
@Configuration
public class ListAsBeanExample {

    @Bean
    public List<String> fruits() {
        return Arrays.asList("apple", "banana", "orange");
    }


    @Bean
    public TestBean testBean(){
        return new TestBean();
    }


    private static class TestBean {
        @Autowired
        private List<String> fruits;


        @PostConstruct
        public void postConstruct() {
            System.out.println(fruits);
        }
    }


    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                ListAsBeanExample.class);
    }
}
```

Output

```
[apple, banana, orange]
```

Bean of Set

```
@Configuration

public class SetAsBeanExample {


    @Bean

    public Set<String> fruits() {

        return new HashSet<>(Arrays.asList("apple", "banana", "orange"));

    }


    @Bean

    public TestBean testBean(){

        return new TestBean();

    }


    private static class TestBean {

        @Autowired

        private Set<String> fruits;


        @PostConstruct

        public void postConstruct() {

            System.out.println(fruits);
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                SetAsBeanExample.class);
    }
}
```

Output

```
[banana, orange, apple]
```

Bean of Map

```
@Configuration
public class MapAsBeanExample {

    @Bean
    public HashMap<String, String> fruits() {
        HashMap<String, String> map = new HashMap<>();
        map.put("apple", "every morning");
        map.put("banana", "after lunch");
        map.put("orange", "every evening");
        return map;
    }


    @Bean
    public TestBean testBean(){
        return new TestBean();
    }


    private static class TestBean {
        @Autowired
        private Map<String, String> fruits;


        @PostConstruct
        public void postConstruct() {
            System.out.println(fruits);
        }
    }


    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                MapAsBeanExample.class);
    }
}
```

Output

```
{banana=after lunch, orange=every evening, apple=every morning}
```

Bean of User Object List

```
@Configuration
public class ObjectListAsBeanExample {

    @Bean
    public List<Fruit> fruits() {
        return Arrays.asList(
                new Fruit("apple", "every morning"),
                new Fruit("banana", "after lunch"),
                new Fruit("orange", "every evening")
        );
    }

    @Bean
    public TestBean testBean() {
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        private List<Fruit> fruits;
        @PostConstruct
        public void postConstruct() {
            System.out.println(fruits);
        }
    }

    private static class Fruit {
        private String name;
        private String whenToEat;

        public Fruit(String name, String whenToEat) {
            this.name = name;
            this.whenToEat = whenToEat;
        }

        @Override
        public String toString() {
            return "Fruit{" +
                    "name='" + name + '\'' +
                    ", whenToEat='" + whenToEat + '\'' +
                    '}';
        }
    }


    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                ObjectListAsBeanExample.class);
    }
}
```

Output

```
[Fruit{name='apple', whenToEat='every morning'}, Fruit{name='banana',
whenToEat='after lunch'}, Fruit{name='orange', whenToEat='every evening'}]
```

2 - Injecting multiple Beans Into Arrays and Collections

This example shows how to inject multiple beans into Arrays and Collections.

Examples

Beans

```
public interface Account {
}


@Component
class SavingAccount implements Account {
    @Override
    public String toString() {
        return "SavingAccount";
    }
}

@Component
class CheckingAccount implements Account {
    @Override
    public String toString() {
        return "CheckInAccount";
    }
}

@Component
class FixedDepositAccount implements Account {
    @Override
    public String toString() {
        return "FixedDepositAccount";
    }
}
```

Injecting beans into Arrays

```
@Configuration
```

```
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingArrayOfBeansExample {

    @Bean
    public TestBean testBean(){
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        private Account[] accounts;

       @PostConstruct
        public void init() {
           System.out.println(Arrays.toString(accounts));
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingArrayOfBeansExample.class);
    }
}
```

Output

```
[CheckInAccount, FixedDepositAccount, SavingAccount]
```

Injecting beans into List

```
@Configuration
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingListOfBeansExample {

    @Bean
    public TestBean testBean() {
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        private List<Account> accounts;

        @PostConstruct
        public void init() {
            System.out.println(accounts);
        }
    }
    public static void main(String[] args) {

        new AnnotationConfigApplicationContext(

                InjectingListOfBeansExample.class);

    }

}
```

Output

```
[CheckInAccount, FixedDepositAccount, SavingAccount]
```

Injecting beans into Set

```
@Configuration

@ComponentScan(basePackages = "com.piseth.java.school.beans")

public class InjectingSetOfBeansExample {


    @Bean

    public TestBean testBean() {

        return new TestBean();

    }


    private static class TestBean {

        @Autowired

        private Set<Account> accounts;
        @PostConstruct
        public void init() {
            System.out.println(accounts);
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingSetOfBeansExample.class);
    }
}
```

Output

```
[CheckInAccount, FixedDepositAccount, SavingAccount]
```

Injecting beans into Map

In this case the Map's keys will contain the corresponding bean names and the Map's values will be beans instances.

```
@Configuration
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingMapOfBeansExample {

    @Bean
    public TestBean testBean() {
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        private Map<String, Account> accounts;

        @PostConstruct
        public void init() {
            System.out.println(accounts);
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingMapOfBeansExample.class);
    }
}
```

Output

```
{checkingAccount=CheckInAccount, fixedDepositAccount=FixedDepositAccount,
savingAccount=SavingAccount}
```

3 - Injecting beans into Arrays and Collections, selecting elements with @Qualifier annotation

In the last example we saw how to inject multiple beans into arrays and collections. This example shows how to use @Qualifier annotation for selection of array/collection/map elements.

Examples

Beans with @Qualifier

```
public interface Account {
}

@Component
@Qualifier("basicAccount")
class SavingAccount implements Account {
    @Override
    public String toString() {
        return "SavingAccount";
    }
}

@Component
@Qualifier("basicAccount")
```

```
class CheckingAccount implements Account {
    @Override
    public String toString() {
        return "CheckInAccount";
    }
}

@Component
class FixedDepositAccount implements Account {
    @Override
    public String toString() {
        return "FixedDepositAccount";
    }
}
```

Injecting beans into Array with @Qualifier

```
@Configuration
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingArrayOfBeansExample {

    @Bean
    public TestBean testBean(){
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        @Qualifier("basicAccount")
        private Account[] accounts;

       @PostConstruct
        public void init() {
           System.out.println(Arrays.toString(accounts));
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingArrayOfBeansExample.class);
    }
}
```

Output

```
[CheckInAccount, SavingAccount]
```

Injecting beans into List/Set with @Qualifier

```
@Configuration

@ComponentScan(basePackages = "com.piseth.java.school.beans")

public class InjectingListOfBeansExample {


    @Bean

    public TestBean testBean() {

        return new TestBean();

    }


    private static class TestBean {

        @Autowired

        @Qualifier("basicAccount")

        private List<Account> accounts;


        @PostConstruct

        public void init() {

            System.out.println(accounts);

        }

    }


    public static void main(String[] args) {

        new AnnotationConfigApplicationContext(

                InjectingListOfBeansExample.class);
    }
}
```

Output

```
[CheckInAccount, SavingAccount]
```

```
@Configuration
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingSetOfBeansExample {

    @Bean
    public TestBean testBean() {
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        @Qualifier("basicAccount")
        private Set<Account> accounts;

        @PostConstruct
        public void init() {
            System.out.println(accounts);
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingSetOfBeansExample.class);
    }
}
```

Output

```
[CheckInAccount, SavingAccount]
```

Injecting beans into Map with @Qualifier

```
@Configuration
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingMapOfBeansExample {

    @Bean
    public TestBean testBean() {
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        @Qualifier("basicAccount")
        private Map<String, Account> accounts;

        @PostConstruct
        public void init() {
            System.out.println(accounts);
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingMapOfBeansExample.class);
    }
}
```

Output

```
{checkingAccount=CheckInAccount, savingAccount=SavingAccount}
```

4 - Injecting beans Into Arrays and Lists, ordering with @Ordered annotation

While injecting beans into Array and List, the elements can be ordered by using @Order annotation.

Definition of @Order annotation

```
package org.springframework.core.annotation;

 .......

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.TYPE, ElementType.METHOD, ElementType.FIELD})

@Documented

public @interface Order {

    int value() default 2147483647;

}
```

Examples

Beans using @Order annotation

```
package com.piseth.java.school.beans;



public interface Account {

}


@Component
```

```
@Order(1)

class SavingAccount implements Account {

    @Override

    public String toString() {

        return "SavingAccount";

    }

}


@Component

@Order(3)

class CheckingAccount implements Account {

    @Override

    public String toString() {

        return "CheckInAccount";

    }

}


@Component

@Order(2)

class FixedDepositAccount implements Account {

    @Override

    public String toString() {

        return "FixedDepositAccount";
    }
}
```

Injecting Ordered elements into Array

```
@Configuration
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingArrayOfBeansExample {

    @Bean
    public TestBean testBean(){
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        private Account[] accounts;

       @PostConstruct
        public void init() {
           System.out.println(Arrays.toString(accounts));
        }
    }
    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingArrayOfBeansExample.class);
    }
}
```

Output

```
[SavingAccount, FixedDepositAccount, CheckInAccount]
```

Injecting Ordered elements into List

```
package com.piseth.java.school;



@Configuration
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingListOfBeansExample {

    @Bean
    public TestBean testBean() {
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        private List<Account> accounts;
        @PostConstruct
        public void init() {
            System.out.println(accounts);
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingListOfBeansExample.class);
    }
}
```

Output

```
[SavingAccount, FixedDepositAccount, CheckInAccount]
```

Ordering specified by @Order annotations is ignored by Set and Map

```
@Configuration
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingSetOfBeansExample {

    @Bean
    public TestBean testBean() {
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        private Set<Account> accounts;

        @PostConstruct
        public void init() {
            System.out.println(accounts);
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingSetOfBeansExample.class);
    }
}
```

Output

```
[CheckInAccount, FixedDepositAccount, SavingAccount]
```

```
@Configuration
```

```
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingMapOfBeansExample {

    @Bean
    public TestBean testBean() {
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        private Map<String, Account> accounts;

        @PostConstruct
        public void init() {
            System.out.println(accounts);
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingMapOfBeansExample.class);
    }
}
```

Output

```
{checkingAccount=CheckInAccount, fixedDepositAccount=FixedDepositAccount,
savingAccount=SavingAccount}
```

5 - Injecting Beans Into Arrays And Collections, ordering with Ordered Interface

Another way of order beans while injecting them into Array and List, is by using interface Ordered.

Definition of Ordered interface

```
package org.springframework.core;


public interface Ordered {
    int getOrder();
}
```

Examples

Beans implementing Ordered interface

```
public interface Account{
}
```

```
@Component
class SavingAccount implements Account, Ordered {
    @Override
    public String toString() {
        return "SavingAccount";
    }

    @Override
    public int getOrder() {
        return 2;
    }
}

@Component
class CheckingAccount implements Account, Ordered {
    @Override
    public String toString() {
        return "CheckInAccount";
    }

    @Override
    public int getOrder() {
        return 3;
    }
}

@Component
class FixedDepositAccount implements Account, Ordered {
    @Override
    public String toString() {
        return "FixedDepositAccount";
    }

    @Override
    public int getOrder() {
        return 1;
    }
}
```

Injecting beans into Arrays

```
@Configuration
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingArrayOfBeansExample {

    @Bean
    public TestBean testBean(){
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        private Account[] accounts;

       @PostConstruct
        public void init() {
          System.out.println(Arrays.toString(accounts));
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingArrayOfBeansExample.class);
    }
}
```

Output

```
[FixedDepositAccount, SavingAccount, CheckInAccount]
```

Injecting beans into List

```
@Configuration
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingListOfBeansExample {

    @Bean
    public TestBean testBean() {
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        private List<Account> accounts;

        @PostConstruct
        public void init() {
            System.out.println(accounts);
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingListOfBeansExample.class);
    }
}
```

Output

```
[FixedDepositAccount, SavingAccount, CheckInAccount]
```

Ordering specified by Ordered implementations is ignored by Set and Map

```
@Configuration
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingSetOfBeansExample {

    @Bean
    public TestBean testBean() {
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        private Set<Account> accounts;

        @PostConstruct
        public void init() {
            System.out.println(accounts);
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingSetOfBeansExample.class);
    }
}
```

Output

```
[CheckInAccount, FixedDepositAccount, SavingAccount]
```

```
@Configuration
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingMapOfBeansExample {

    @Bean
    public TestBean testBean() {
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        private Map<String, Account> accounts;
        @PostConstruct
        public void init() {
            System.out.println(accounts);
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingMapOfBeansExample.class);
    }
}
```

Output

```
{checkingAccount=CheckInAccount, fixedDepositAccount=FixedDepositAccount,
savingAccount=SavingAccount}
```

6 - Injecting beans into Arrays/Collections, Using @Qualifiers And Specifying the Ordering

This example uses both @Qualifier and @Order at a time.

Example

Beans

```
public interface Account {
}

@Component
@Order(1)
@Qualifier("basicAccount")
class SavingAccount implements Account {
    @Override
    public String toString() {
        return "SavingAccount";
    }
}

@Component
@Order(3)
@Qualifier("basicAccount")
class CheckingAccount implements Account {
    @Override
    public String toString() {
        return "CheckInAccount";
    }
}
```

```
@Component
@Order(2)
class FixedDepositAccount implements Account {
    @Override
    public String toString() {
        return "FixedDepositAccount";
    }
}
```

Injecting beans into Arrays

```
@Configuration
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingArrayOfBeansExample {

    @Bean
    public TestBean testBean(){
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        @Qualifier("basicAccount")
        private Account[] accounts;
```

```
       @PostConstruct
        public void init() {
           System.out.println(Arrays.toString(accounts));
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingArrayOfBeansExample.class);
    }
}
```

Output

```
[SavingAccount, CheckInAccount]
```

Injecting beans into List

```
@Configuration
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingMapOfBeansExample {

    @Bean
    public TestBean testBean() {
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        @Qualifier("basicAccount")
        private Map<String, Account> accounts;

        @PostConstruct
        public void init() {
            System.out.println(accounts);
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingMapOfBeansExample.class);
    }
}
```

Output

```
{checkingAccount=CheckInAccount, savingAccount=SavingAccount}
```

Injecting beans into Set

Sets ignore the ordering specified by @Ordered annotations.

```
@Configuration
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingSetOfBeansExample {

    @Bean
    public TestBean testBean() {
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        @Qualifier("basicAccount")
        private Set<Account> accounts;

        @PostConstruct
        public void init() {
            System.out.println(accounts);
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingSetOfBeansExample.class);
    }
}
```

Output

```
[CheckInAccount, SavingAccount]
```

Injecting beans into Map

Maps also ignore ordering.

```
@Configuration
@ComponentScan(basePackages = "com.piseth.java.school.beans")
public class InjectingMapOfBeansExample {

    @Bean
    public TestBean testBean() {
        return new TestBean();
    }

    private static class TestBean {
        @Autowired
        @Qualifier("basicAccount")
        private Map<String, Account> accounts;
        @PostConstruct
        public void init() {
            System.out.println(accounts);
        }
    }

    public static void main(String[] args) {
        new AnnotationConfigApplicationContext(
                InjectingMapOfBeansExample.class);
    }
}
```

Output

```
{checkingAccount=CheckInAccount, savingAccount=SavingAccount}
```

Summary