CS544 EA

Spring

DI Features: Collections

Collections: List

- Collections can also be injected

Also see: https://www.baeldung.com/spring-injecting-collections

```java
package cs544.spring17.di.list;

import java.util.Arrays;
import java.util.List;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

@Configuration
@ComponentScan("cs544.spring17.di.list")
public class Config {
	@Bean
	public List<String> names() {
		return Arrays.asList("John", "Jim", "Jane");
	}
}
```

```java
package cs544.spring17.di.list;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class CustomerService {
	@Autowired
	private List<String> names;

	public void sayHello() {
		names.stream().forEach(
			n -> System.out.println("Hello " + n));
	}


}
```

XML List

```xml
<bean id="customerService" class="cs544.spring17.di.list.CustomerService">
	<property name="names">
		<list>
			<value>John</value>
			<value>Jim</value>
			<value>Jane</value>
		</list>
	</property>
</bean>
```

Also possible inside <constructor-arg>

```java
package cs544.spring17.di.list;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

public class CustomerService {
	private List<String> names;

	public void sayHello() {
		names.stream().forEach(
			n -> System.out.println("Hello " + n));
	}
	public List<String> getNames() {
		return names;
	}
	public void setNames(List<String> names) {
		this.names = names;
	}
}
```

Set

```xml
<bean id="customerService" class="cs544.spring18.di.set.CustomerService">
	<property name="names">
		<set>
			<value>John</value>
			<value>Jim</value>
			<value>Jane</value>
		</set>
	</property>
</bean>
```

```java
package cs544.spring18.di.set;

import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

@Configuration
@ComponentScan("cs544.spring18.di.set")
public class Config {
	@Bean
	public Set<String> names() {
		return new HashSet<String>(Arrays.asList("John", "Jim", "Jane"));

	}
}
```

Map

```xml
<bean id="customerService" class="cs544.spring19.di.map.CustomerService">
	<property name="names">
		<map>
			<entry key="John" value="guy" />
			<entry key="Jim" value="guy" />
			<entry key="Janet" value="girl" />
		</map>
	</property>
</bean>
```

```java
package cs544.spring19.di.map;

import java.util.HashMap;
import java.util.Map;

import org.springframework.context.annotation.
import org.springframework.context.annotation.
import org.springframework.context.annotation.Configuration;

@Configuration
@ComponentScan("cs544.spring19.di.map")
public class Config {
	@Bean
	public Map<String, String> names() {
		Map<String, String> m = new HashMap<>();
		m.put("John", "guy");
		m.put("Jane", "girl");
		m.put("Janet", "girl");
		return m;
	}
}
```

References Collection XML

```xml
<bean id="shopService" class="cs544.spring20.di.refs.ShopService">
	<property name="finders">
		<list>
			<ref bean="amazon"/>
			<ref bean="ebay"/>
		</list>
	</property>
</bean>
<bean id="amazon" class="cs544.spring20.di.refs.AmazonPriceFinder" />
<bean id="ebay" class="cs544.spring20.di.refs.EbayPriceFinder" />
```

```java
package cs544.spring20.di.refs;

import java.util.List;

import org.springframework.beans.factory.annot
import org.springframework.stereotype.Service;

public class ShopService {
	private List<PriceFinder> finders;

	public void prices() {
		finders.stream().forEach(
			x -> System.out.println(x.name() + ": " + x.price()));
	}
	public List<PriceFinder> getFinders() {
		return finders;
	}
	public void setFinders(List<PriceFinder> finders) {
		this.finders = finders;
	}
}
```

References Java Conf

- If you don't have a @Bean that returns a list of references, Spring will make a list from all the bean's that return the requested type
  - Found from component scan or @Bean defs

```
package cs544.spring20.di.refs;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class Config {
	@Bean
	public ShopService shopService() {
		return new ShopService();
	}
	@Bean
	public PriceFinder amazon() {
		return new AmazonPriceFinder();
	}
	@Bean
	public PriceFinder eBay() {
		return new EbayPriceFinder();
	}
}
```

```
package cs544.spring20.di.refs;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;

public class ShopService {
	@Autowired
	private List<PriceFinder> finders;

	...
}
```

@Qualifier

- Using @Qualifier you can specify which beans of that type ‘are qualified’

```
package cs544.spring21.di.refs;

...

@Configuration
@ComponentScan("cs544.spring21.di.refs")
public class Config {
	@Bean
	@Qualifier("finders")
	public PriceFinder amazon() {
		return new AmazonPriceFinder();
	}

	@Bean
	public PriceFinder eBay() {
		return new EbayPriceFinder();
	}

	@Bean
	@Qualifier("finders")
	public PriceFinder newEgg() {
		return new NewEggPriceFinder();
	}

}
```

```
@Service
public class ShopService {
	@Autowired
	@Qualifier("finders")
	private List<PriceFinder> finders;
```

qualifier on the collection

Ebay not in list (not qualified)

@Order

```
package cs544.spring22.di.refs;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.annotation.Order;

@Configuration
@ComponentScan("cs544.spring22.di.refs")
public class Config {
	@Bean
	@Order(1)
	public PriceFinder amazon() {
		return new AmazonPriceFinder();
	}
	@Bean
	@Order(3)
	public PriceFinder eBay() {
		return new EbayPriceFinder();
	}
	@Bean
	@Order(2)
	public PriceFinder newEgg() {
		return new NewEggPriceFinder();
	}

}
```

You can use @Order to indicate in which order they should be inserted into the collection.

Does not have to be zero based, or even contiguous