SPRING FRAMEWORK

SCOPES AND INNER BEANS

APRIL, 2015

EXAMPLE

DIY

Event

```
- id: int;      // Auto-generated
- msg: String;  // Set in setter
- date: Date;   // Set in constructor
+ toString()
```

EXAMPLE

DIY

Event

```
- id: int;      // Auto-generated
- msg: String;  // Set in setter
- date: Date;   // Set in constructor
+ toString()
```

EventLogger

+ logEvent(Event event)

EXAMPLE

DIY

Event

```
- id: int;      // Auto-generated
- msg: String;  // Set in setter
- date: Date;   // Set in constructor
+ toString()
```

EventLogger

```
+ logEvent(Event event)
```

Class App now creates Event objects.

Or, maybe, it should receive them? :)

BEAN SCOPE

By default all beans are singletons (within container)

```
<bean id="..." class="...">
<bean id="..." class="..." scope="singleton">
```

Prototype – new object on every getBean() call

```
<bean id="..." class="..." scope="prototype">
```

For web applications:

- request
- session
- global-session

BEAN SCOPE

DIY

By default all beans are singletons (within container)

```
<bean id="..." class="...">
<bean id="..." class="..." scope="singleton">
```

Prototype – new object on every getBean() call

```
<bean id="..." class="..." scope="prototype">
```

For web applications:

- request
- session
- global-session

INNER BEANS

DIY

```
<bean id="event" class="..." scope="prototype">

   <constructor-arg>
      <bean class="java.util.Date"/>
   </constructor-arg>

</bean>
```

- Visible where they are defined – can’t be reused

FACTORY METHOD

DIY

```
public Event(Date date, DateFormat df) {
   this.date = date; this.df = df;
}

public String toString() {
   ... df.format(date) ...
}
```

FACTORY METHOD

DIY

```
public Event(Date date, DateFormat df) {
    this.date = date; this.df = df;
}

public String toString() {
    ... df.format(date) ...
}
```

DateFormat - abstract class

```
DateFormat.getDateInstance()
DateFormat.getTimeInstance()
DateFormat.getDateTimeInstance()
```

FACTORY METHOD

DIY

```
public Event(Date date, DateFormat df) {
   this.date = date; this.df = df;
}
public String toString() {
   ... df.format(date) ...
}
```

DateFormat - abstract class

```
DateFormat.getDateInstance()
DateFormat.getTimeInstance()
DateFormat.getDateTimeInstance()
```

```
<bean id="dateFormat"
   class="java.text.DateFormat"
      factory-method="getDateTimeInstance"/>
```

THANK YOU

YURIY_TKACH@EPAM.COM

APRIL, 2015