luv2code

Testing using Reflection

Special Cases during Testing

- Need to access non-public fields
  - Read the field's value
  - Set the field's value

```
@Id
@GeneratedValue(...)
private Long productId;
```

```
@Autowired
private ApplicationService service;
```

- Invoke non-public methods

```
private String generateTrackingNumber() { ... }
```

- Testing legacy code ... can't change code due to corporate policy
- In general, testing non-public fields and methods is controversial ... use sparingly

Spring: ReflectionTestUtils

- Spring provides a utility class: `ReflectionTestUtils`
- Allows you to get/set non-public fields directly
- Can also invoke non-public methods
- JavaDocs provide additional use cases and examples

www.luv2code.com/reflection-test-utils-javadoc

Reading Private Fields

CollegeStudent.java

```
public class CollegeStudent implements Student {
    private int id;
   ...

}
```

Private field: id

DemoTest.java

```
CollegeStudent theStudent = (CollegeStudent) context.getBean("collegeStudent");

ReflectionTestUtils.getField(theStudent, "id");
```

From the Spring Framework

target object

field name

Setting Private Fields

CollegeStudent.java

```
public class CollegeStudent implements Student {
    private int id;
   ...

}
```

DemoTest.java

```
CollegeStudent theStudent = (CollegeStudent) context.getBean("collegeStudent");

ReflectionTestUtils.setField(theStudent, "id", 1);
```

value

target object

field name

Invoking Private Methods

CollegeStudent.java

```
public class CollegeStudent implements Student {
   ...
   private String getFirstNameAndId() {
     return getFirstname() + " " + getId();
   }

}
```

DemoTest.java

```
CollegeStudent theStudent = (CollegeStudent) context.getBean("collegeStudent");

ReflectionTestUtils.invokeMethod(theStudent, "getFirstNameAndId");
```

target object

method name