Extensions

Spock comes with a powerful extension mechanism, which allows to hook into a spec's lifecycle to enrich or alter its behavior. In this chapter, we will first learn about Spock's built-in extensions, and then dive into writing custom extensions.

Built-In Extensions

Most of Spock's built-in extensions are *annotation-driven*. In other words, they are triggered by annotating a spec class or method with a certain annotation. You can tell such an annotation by its `@ExtensionAnnotation` meta-annotation.

Ignore

To temporarily prevent a feature method from getting executed, annotate it with `spock.lang.Ignore`:

```
@Ignore
def "my feature"() { ... }
```

For documentation purposes, a reason can be provided:

```
@Ignore(reason = "TODO")
def "my feature"() { ... }
```

To ignore a whole specification, annotate its class:

```
@Ignore
class MySpec extends Specification { ... }
```

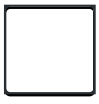
In most execution environments, ignored feature methods and specs will be reported as "skipped".

IgnoreRest

To ignore all but a (typically) small subset of methods, annotate the latter with `spock.lang.IgnoreRest`:

```
def "I'll be ignored"() { ... }

@IgnoreRest
```



```
def "I'll run"() { ... }

def "I'll also be ignored"() { ... }
```

`@IgnoreRest` is especially handy in execution environments that don't provide an (easy) way to run a subset of methods.

IgnoreIf

To ignore a feature method under certain conditions, annotate it with `spock.lang.IgnoreIf`, followed by a predicate:

```
@IgnoreIf({ System.getProperty("os.name").contains("windows") })
def "I'll run everywhere but on Windows"() { ... }
```

To make predicates easier to read and write, the following properties are available inside the closure:

- `sys` A map of all system properties
- `env` A map of all environment variables
- `os` Information about the operating system (see `spock.util.environment.OperatingSystem`)
- `jvm` Information about the JVM (see `spock.util.environment.Jvm`)

Using the `os` property, the previous example can be rewritten as:

```
@IgnoreIf({ os.windows })
def "I'll run everywhere but on Windows"() { ... }
```

Requires

To execute a feature method under certain conditions, annotate it with `spock.lang.Requires`, followed by a predicate:

```
@Requires({ os.windows })
def "I'll only run on Windows"() { ... }
```

`Requires` works exactly like `IgnoreIf`, except that the predicate is inverted. In general, it is preferable to state the conditions under which a method gets executed, rather than the conditions under which it gets ignored.

TODO More to follow.

Writing Custom Extensions

TODO