

Script your **Java** apps with **Groovy**

Reinhard Pointner
郝瑞尼



Outline

- **What** is **Groovy**? **Why Groovy**?
- **Tips & Tricks** for **Groovy DSL** development **from Java**
 - ExtensionModule
 - CompilerConfiguration
 - ScriptBaseClass
 - ImportCustomizer
 - GroovyObjectSupport
 - DefaultTypeTransformation
 - DefaultGroovyMethods



Java vs Groovy

	Java	Groovy
Syntax	Strict	Lenient
Code	Verbose	“Groovy 1-Liners”
Strength	Clean & Readable & Robust Code	Developer Productivity
	Great for large products!	Great for 3rd party scripts!

“Talk is cheap. Show me the code.”

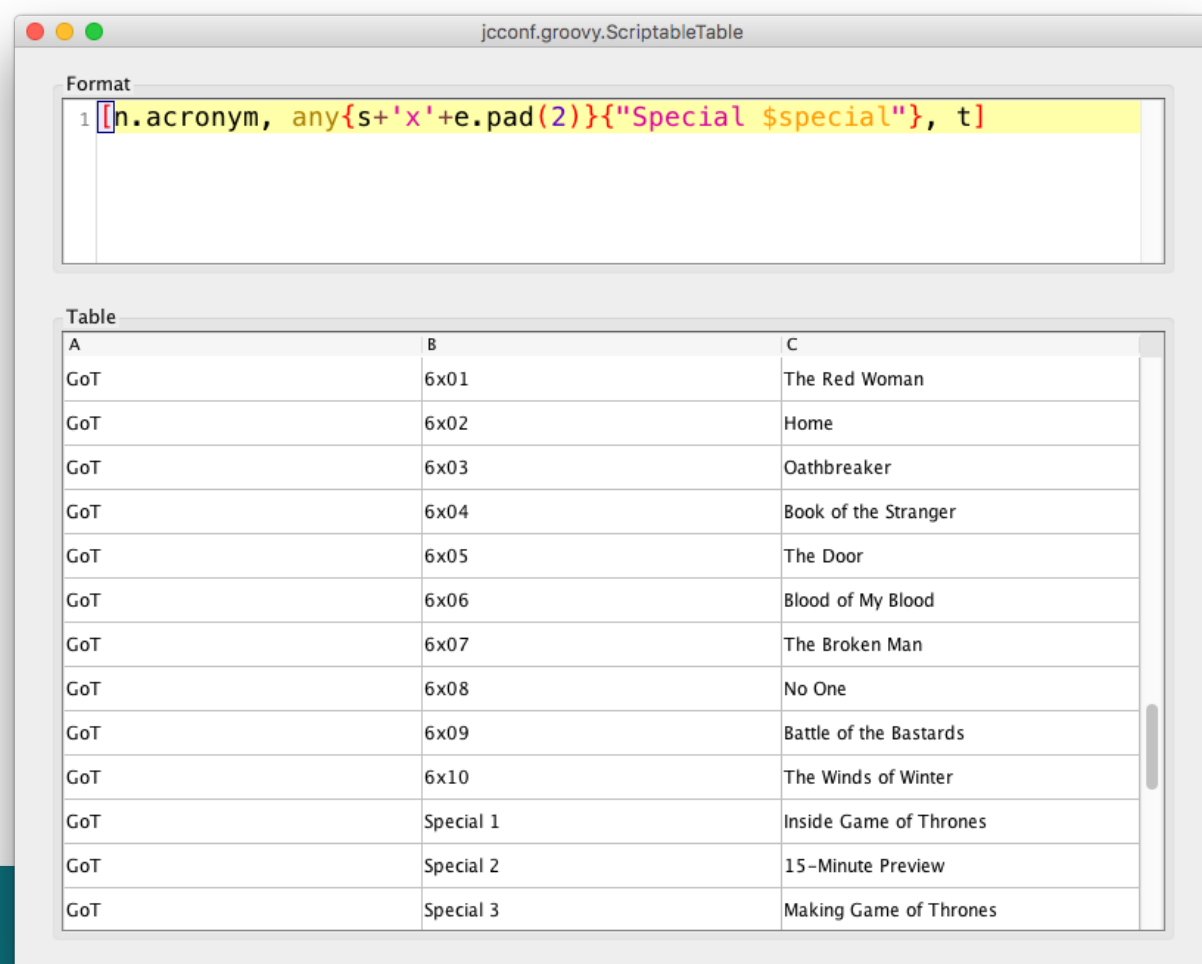
// Linus Torvalds

Example Application

on

<https://github.com/rednoah/GroovyTableFormat>

// TODO markers are your friend



The screenshot shows a Java Swing window titled "jconf.groovy.ScriptableTable". It has two main sections: "Format" and "Table".

The "Format" section contains a text area with the following Groovy script:

```
1 [n.acronym, any{s+'x'+e.pad(2)}{"Special $special"}, t]
```

The "Table" section displays a rendered table with 3 columns: A, B, and C. The table contains 15 rows of data, including standard Game of Thrones episodes and special content.

A	B	C
GoT	6x01	The Red Woman
GoT	6x02	Home
GoT	6x03	Oathbreaker
GoT	6x04	Book of the Stranger
GoT	6x05	The Door
GoT	6x06	Blood of My Blood
GoT	6x07	The Broken Man
GoT	6x08	No One
GoT	6x09	Battle of the Bastards
GoT	6x10	The Winds of Winter
GoT	Special 1	Inside Game of Thrones
GoT	Special 2	15-Minute Preview
GoT	Special 3	Making Game of Thrones

Use Groovy to generate
user-defined table views.

ExtensionModule

“Adding Properties and Methods”

① Java API

```
"Groovy".toUpperCase() // GROOVY
```

② Groovy Default Methods

```
"uname -m".execute().text // x86_64
```

③ My Default Methods

```
1.pad 2 // 02
```

```
“台灣”.pinyin // tái wān
```

ExtensionModule

“Adding Properties and Methods”

① META-INF/services/org.codehaus.groovy.runtime.ExtensionModule

```
moduleName=GroovyTableCustomExtensionMethods  
moduleVersion=1.0  
extensionClasses=jcconf.groovy.script.CustomExtensionMethods
```

② jcconf.groovy.script.CustomExtensionMethods

```
public static String pad(Number self, int length) {  
    return String.format("%0" + length + "d", self);  
}
```

```
public static String getPinyin(String self) {  
    return Transliterator.getInstance("Han-Latin").transform(self);  
}
```

ScriptBaseClass

“Variables and default values”

① jcconf.groovy.script.CustomScriptBaseClass

```
public Object getProperty(String property) {  
    try {  
        return super.getProperty(property);  
    } catch (MissingPropertyException e) {  
        return null;  
    }  
}
```

② Undefined variables now default to null

```
println asdf // null
```

Groovy with default ScriptBaseClass:

groovy.lang.MissingPropertyException: No such property: asdf

ImportCustomizer

“Adding default imports”

① GroovyScriptEngine << GroovyClassLoader << CompilerConfiguration

```
ImportCustomizer imports = new ImportCustomizer();  
imports.addStaticStars(Math.class.getName());  
compilerConfiguration.addCompilationCustomizers(imports);
```

② Use imported classes and static methods or static fields

```
sqrt(4) // 2
```

Groovy with default CompilerConfiguration:

```
import static java.lang.Math.*  
sqrt(4) // 2
```

GroovyObjectSupport

“Dynamic Properties and Methods”

① Override `invokeMethod` and `getProperty`

```
public class Undefined extends GroovyObjectSupport {  
    [...]  
    public Object getProperty(String property) {  
        return new Undefined(name + "." + property);  
    }  
    public Object invokeMethod(String method, Object args) {  
        return new Undefined(name + "." + method + "()");  
    }  
}
```

② Groovy:

```
x.y.z() // Undefined: x.y.z()
```

DefaultTypeTransformation

“Getting to the Groovy Truth”

① Groovy Truth:

```
null as boolean // false  
' ' as boolean // false  
[] as boolean // false  
0 as boolean // false
```

② Groovy Truth from Java:

```
DefaultTypeTransformation.castToBoolean(object) // Groovy Truth
```

* Also useful for casting Groovy closures as Java interfaces:

① Groovy: `def closure = { f -> f.hidden }`

② Java: `DefaultTypeTransformation.castToType(closure, FileFilter.class)`

DefaultGroovyMethods

“Using the Groovy methods from Java”

① Groovy:

```
def condition = { it % 2 == 0 }  
[...]  
return (1..10).findAll(condition).join(', ') // 2, 4, 6, 8, 10
```

② Java:

```
// Groovy Closure `condition` is a given  
Range range = new IntRange(1, 10);  
Iterable values = DefaultGroovyMethods.findAll(range, condition);  
return DefaultGroovyMethods.join(values, ", ");
```

Closure.rehydrate()

“Invoking the Groovy Builder Pattern”

① Groovy:

```
StringWriter sw = new StringWriter()
new groovy.xml.MarkupBuilder(sw).html{
    body{ p(id:1, 'Groovy') }
}
println sw.toString()
```

② Java:

```
public String XML(Closure closure) {
    StringWriter sw = new StringWriter();
    MarkupBuilder mb = new MarkupBuilder(sw);
    closure.rehydrate(closure.getDelegate(), mb, mb).call();
    return sw.toString();
}
```

③ My Groovy:

```
XML{
    html{
        body{ p(id:1, 'Groovy') }
    }
}
```

```
<html>
  <body>
    <p id='1'>Groovy</p>
  </body>
</html>
```

```
println XML{
    jcconf(year:2016){
        talk{
            name('Script your Java applications with Groovy')
            github('https://github.com/rednoah/GroovyTableFormat')
            speaker{
                name(lang:'en', 'Reinhard Pointner')
                name(lang:'zh', '郝瑞尼')
                email('ruini@me.com')
                github('https://github.com/rednoah')
            }
        }
    }
}

// Goodbye!
System.exit(0)
```