SKILLS MATTER

Grails plugin examples

2007

GRAILS eXCHANGE

http://www.grails-exchange.com | http://www.grails.org | http://skillsmatter.com

ExpandoMetaClass

© Steven Devijver

ExpandoMetaClass is everywhere:

```
assert String.metaClass instanceof ExpandoMetaClass

assert Class.metaClass instanceof ExpandoMetaClass
```

© Steven Devijver

This meta class method definition:

```
import org.springframework.util.StringUtils
String.metaClass.hasText = {->
   StringUtils.hasText(delegate)
}
```

Let's you do this:

```
assert "someText".hasText()
assert !"".hasText()
assert !"  ".hasText()
```

© Steven Devijver

This meta class method definition:

```
String.metaClass.newReader = {->
    new StringReader(delegate)
}
```

Let's you do this:

```
String fileContents = new File("data.log").text

Reader reader = fileContents.newReader()
```

© Steven Devijver

Where to define methods

In the BootStrap class:

```
class BootStrap {
   def init = { servletContext ->
      String.metaClass.newReader = {->
         new StringReader(delegate)
      }
   }
   def destroy = {}
}
```

© Steven Devijver

Where to define methods (2)

In plugin classes:

```
class MypluginGrailsPlugin {
    def version = 0.1
    def dependsOn = [:]
    def doWithDynamicMethods = {
        ctx ->
        String.metaClass.newReader = {->
            new StringReader(delegate)
        }
    }
}
```

© Steven Devijver

Extending Spring

© Steven Devijver

Getting beans

```
def doWithDynamicMethods = {
   applicationContext ->
   String.metaClass.bean = {->
      applicationContext.getBean(delegate)
   }
}
```

Let's you do this:

```
def sessionFactory = "sessionFactory".bean()
```

Spring

© Steven Devijver

Getting all beans of a type

```
def doWithDynamicMethods = {
   ctx ->
   Class.metaClass.beansOfType = {->
      ctx.getBeansOfType(delegate).values()
   }
}
```

Let's you do this:

```
def dataSources = DataSource.beansOfType()
```

Spring

Autowiring objects

```
import o.sfw...AutowireCapableBeanFactory as ACBF
def doWithDynamicMethods = {
   ctx ->
   Object.metaClass.autowireByType = {->
      ctx.beanFactory.autowireBeanProperties(delegate,
   ACBF.AUTOWIRE_BY_TYPE, false)
      return delegate
   }
}
```

Let's you do this:

```
def myObject = new MyObject().autowireByType()
```

Spring

© Steven Devijver

Autowiring objects (2)

```
import o.sfw...AutowireCapableBeanFactory as ACBF
def doWithDynamicMethods = {
   ctx ->
   Object.metaClass.autowireByName = {->
      ctx.beanFactory.autowireBeanProperties(delegate,
   ACBF.AUTOWIRE_BY_NAME, false)
      return delegate
   }
}
```

Let's you do this:

```
def myObject = new MyObject().autowireByName()
```

Spring

© Steven Devijver

Loading resources

```
def doWithDynamicMethods = {
    ctx ->
    String.metaClass.asResource = {->
        ctx.getResource(delegate)
    }
}
```

Let's you do this:

```
"http://www.nytimes.com/services/xml/rss/nyt/SundayBookReview.xml"
   .asResource().inputStream
```

Spring

© Steven Devijver

Dispatching events

```
import org.springframework.context.ApplicationEvent
def doWithDynamicMethods = {
   ctx ->
   Object.metaClass.dispatchEvent = {->
      ctx.publish(
         new ApplicationEvent(delegate))
   }
   Collection.metaClass.dispatchEvents = {->
      delegate.each { it.dispatchEvent() }
   }
}
```

Let's you do this:

```
eventObject.dispatchEvent()
[1,2,3].dispatchEvents()
```

Spring

© Steven Devijver

Extending Swing

© Steven Devijver

Extending Grails

© Steven Devijver

Adding methods to controllers

```
def doWithDynamicMethods = {
   ctx ->
   ctx.controllers.metaClass.each { metaClass ->
      metaClass.js = { text ->
         render(contentType:"text/javascript", text: text)
      }
   }
}
```

Let's you do this:

```
class MyController {
   def index = { js("alert('Hello, World!');") }
}
```

© Steven Devijver

Adding methods to controllers (2)

```
import org.springframework.web.util.JavaScriptUtils as JSU
def doWithDynamicMethods = {
   ctx ->
   ctx.controllers.metaClass.each { metaClass ->
      metaClass.alert = { text ->
         js("alert('${JSU.javaScriptEscape(s)}');")
      }
   }
}
```

Let's you do this:

```
class MyController {
   def index = { alert('Hello, World!') }
}
```

© Steven Devijver

Easy RSS reader (1)

```
// Using Project Rome
import com.sun.syndication.io.SyndFeedInput
import com.sun.syndication.io.XmlReader
def doWithDynamicMethods = {
   ctx ->
   InputStream.metaClass.asFeed = {->
      new SyndFeedInput()
         .build(new XmlReader(delegate))
   }
}
```

© Steven Devijver

Easy RSS reader (2)

Controller:

```
class MyFeedController {
   def index = {
     def feed =
     "http://www.nytimes.com/services/xml/rss/nyt/SundayBookReview.xml"
        .asResource().inputStream.asFeed()
     [feed: feed]
   }
}
```

© Steven Devijver

Easy RSS reader (3)

index.gsp

```
<ul>
   <g:each in="${feed}" var="entry">
      <li>
         <a href="${entry.link}">${entry.title}</a>
      </li>
   </g:each>
</ul>
```

© Steven Devijver

Creating a ThreadPool plugin

© Steven Devijver

First things first

```
grails create-plugin ThreadPool
```

© Steven Devijver

Spring config & meta class

ThreadPoolGrailsPlugin.groovy:

```
class ThreadPoolGrailsPlugin {
  def version = 0.1
  def dependsOn = [:]
  def doWithSpring = {
    taskExecutor(o.sfw...ThreadPoolTaskExecutor) {
      maxPoolSize = 100
    }
  }
  def doWithDynamicMethods = { ctx ->
    Runnable.metaClass.start = {
       ctx.getBean("taskExecutor").execute(delegate)
    }
  }
}
```

© Steven Devijver

Package & install

Package

```
grails package-plugin
```

Install

```
grails install-plugin ..\grails-thread-pool-0.1.zip
```

© Steven Devijver

A little test

BootStrap.groovy:

```
class BootStrap {
  def init = {
    println "#### ${Thread.currentThread().name}"

    {->
       println "#### ${Thread.currentThread().name}"
    }.start()
  }
}
```

© Steven Devijver

Run the little test

Run

```
grails run-app
```

Console

```
#### main
#### pool-1-thread-1
```

© Steven Devijver

Plugin requests

- Blog
- Forum
- Wiki
- Issue tracker
- Webmail
- Calendar
- Project management
- Social network
- …

© Steven Devijver

Summary

- Grails plugins are easy and fun
- Meta programming is sexy
- Go home and write plugins

© Steven Devijver