Item 183
git_comments:

1. * * Return the lines of a String as a List of Strings. * * @param self a String object * @return a list of lines

git_commits:

1. **summary:** GROOVY-644: New Groovy JDK methods - to improve consistency (minor tweak to last commit)
   message: GROOVY-644: New Groovy JDK methods - to improve consistency (minor tweak to last commit) git-svn-id: http://svn.codehaus.org/groovy/trunk/groovy/groovy-core@11601 a5544e8c-8a19-0410-ba12-f9af4593a198

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** New Groovy JDK methods - to improve consistency
   description: From my August 2004 email: I've been having a quick look at the Groovy JDK, i.e. DefaultGroovyMethods and DefaultGroovyStaticMethods As there are so many methods in these classes, it can be a little hard to take in all at once. So I've knocked up a quick script (in groovy), that collates the methods against the objects to which they become attached. hacky script -> http://javanicus.com/groovy/MungGroovySourceCode.groovy The result is a pretty table, with the methods down the left hand side, and the Classes to which they are attached along the top. Where a method has been implemented, at the crossover, I have placed a small graphic, which if you hover your cursor over, will give you a bit more detail about the method. http://javanicus.com/groovy/GroovyJDKCrossReference.html I've had a quick look for methods that I thought would be defined and from my quick inspection of abs() thru to leftShift() so far, I believe that the following candidates are available (this is by no means an exhaustive list...) * Some possible missing (non static) methods -Collection.asImmutable()- Object.asImmutable() Object.asSynchronized() -Set.count()- -Byte[].eachByte()- -File.filterLine()- -InputStream.filterLine()- -List.findIndexOf()- -Collection.flatten()- -Date.getAt()- // to get the year etc... (In conjunction with a Calendar?) -Reader.getText()- -Collection.intersect()- -BufferedWriter <<- Object[] << CharSequence << -File <<- -Map << // another Map- -Process <<- // to the process.out -Socket <<- // to the socket.out -Object[].max()- -Object[].min()- Collection.minus() Map.minus() // could compare RHS with key || value -Object[].minus()- -URL.newInputStream()- -URL.newReader()- CharSequence.padLeft() CharSequence.padRight() Object[].pop() CharSequence.putAt() Collection.putAt() // !always ordered, but we have -Collection.getAt()- -Date.putAt()- // e.g. easy access to components of Date Matcher.putAt() InputStream.readBytes() URL.readBytes() -URL.readLines()- CharSequence.reverse() -Object[].reverse()- SortedMap.reverse() SortedSet.reverse() Collection.reverseEach() // like Collection.each() this could be indeterminate... -Map.reverseEach()- // we have Map.each()... Matcher.reverseEach() Object.reverseEach() Object[].reverseEach() // perhaps foo.reverseEach() should be foo.reverse().each() Object.rightShift() // so you can do things like... foo >> log -Object[].sort()- -InputStream.splitEachLine()- -URL.splitEachLine()- OutputStream.withWriterAppend() -BufferedWriter.write()- -OutputStream.write()- File.writeLine() OutputStream.writeLine() * and I'm not quite sure what the difference between append() and << is supposed to be, perhaps File.append() will become File << * As we have {{eachByte()}}, would it not be a good idea to also have {{eachCharacter()}} for the Readers... thanks Jeremy. P.S. This is based entirely on existing methods and existing Owner Objects, would be nice to think of what other Objects and methods we could include, (current owner objects come from java.lang.*, java.util.*, java.util.regex.*, java.io.* and java.net.*)

2. **summary:** New Groovy JDK methods - to improve consistency
   description: From my August 2004 email: I've been having a quick look at the Groovy JDK, i.e. DefaultGroovyMethods and DefaultGroovyStaticMethods As there are so many methods in these classes, it can be a little hard to take in all at once. So I've knocked up a quick script (in groovy), that collates the methods against the objects to which they become attached. hacky script -> http://javanicus.com/groovy/MungGroovySourceCode.groovy The result is a pretty table, with the methods down the left hand side, and the Classes to which they are attached along the top. Where a method has been implemented, at the crossover, I have placed a small graphic, which if you hover your cursor over, will give you a bit more detail about the method. http://javanicus.com/groovy/GroovyJDKCrossReference.html I've had a quick look for methods that I thought would be defined and from my quick inspection of abs() thru to leftShift() so far, I believe that the following candidates are available (this is by no means an exhaustive list...) * Some possible missing (non static) methods -Collection.asImmutable()- Object.asImmutable() Object.asSynchronized() -Set.count()- -Byte[].eachByte()- -File.filterLine()- -InputStream.filterLine()- -List.findIndexOf()- -Collection.flatten()- -Date.getAt()- // to get the year etc... (In conjunction with a Calendar?) -Reader.getText()- -Collection.intersect()- -BufferedWriter <<- Object[] << CharSequence << -File <<- -Map << // another Map- -Process <<- // to the process.out -Socket <<- // to the socket.out -Object[].max()- -Object[].min()- Collection.minus() Map.minus() // could compare RHS with key || value -Object[].minus()- -URL.newInputStream()- -URL.newReader()- CharSequence.padLeft() CharSequence.padRight() Object[].pop() CharSequence.putAt() Collection.putAt() // !always ordered, but we have -Collection.getAt()- -Date.putAt()- // e.g. easy access to components of Date Matcher.putAt() InputStream.readBytes() URL.readBytes() -URL.readLines()- CharSequence.reverse() -Object[].reverse()- SortedMap.reverse() SortedSet.reverse() Collection.reverseEach() // like Collection.each() this could be indeterminate... -Map.reverseEach()- // we have Map.each()... Matcher.reverseEach() Object.reverseEach() Object[].reverseEach() // perhaps foo.reverseEach() should be foo.reverse().each() Object.rightShift() // so you can do things like... foo >> log -Object[].sort()- -InputStream.splitEachLine()- -URL.splitEachLine()- OutputStream.withWriterAppend() -BufferedWriter.write()- -OutputStream.write()- File.writeLine() OutputStream.writeLine() * and I'm not quite sure what the difference between append() and << is supposed to be, perhaps File.append() will become File << * As we have {{eachByte()}}, would it not be a good idea to also have {{eachCharacter()}} for the Readers... thanks Jeremy. P.S. This is based entirely on existing methods and existing Owner Objects, would be nice to think of what other Objects and methods we could include, (current owner objects come from java.lang.*, java.util.*, java.util.regex.*, java.io.* and java.net.*)
3. **summary:** New Groovy JDK methods - to improve consistency
   description: From my August 2004 email: I've been having a quick look at the Groovy JDK, i.e. DefaultGroovyMethods and DefaultGroovyStaticMethods As there are so many methods in these classes, it can be a little hard to take in all at once. So I've knocked up a quick script (in groovy), that collates the methods against the objects to which they become attached. hacky script -> http://javanicus.com/groovy/MungGroovySourceCode.groovy The result is a pretty table, with the methods down the left hand side, and the Classes to which they are attached along the top. Where a method has been implemented, at the crossover, I have placed a small graphic, which if you hover your cursor over, will give you a bit more detail about the method. http://javanicus.com/groovy/GroovyJDKCrossReference.html I've had a quick look for methods that I thought would be defined and from my quick inspection of abs() thru to leftShift() so far, I believe that the following candidates are available (this is by no means an exhaustive list...) * Some possible missing (non static) methods -Collection.asImmutable()- Object.asImmutable() Object.asSynchronized() -Set.count()- -Byte[].eachByte()- -File.filterLine()- -InputStream.filterLine()- -List.findIndexOf()- -Collection.flatten()- -Date.getAt()- // to get the year etc... (In conjunction with a Calendar?) -Reader.getText()- -Collection.intersect()- -BufferedWriter <<- Object[] << CharSequence << -File <<- -Map << // another Map- -Process <<- // to the process.out -Socket <<- // to the socket.out -Object[].max()- -Object[].min()- Collection.minus() Map.minus() // could compare RHS with key || value -Object[].minus()- -URL.newInputStream()- -URL.newReader()- CharSequence.padLeft() CharSequence.padRight() Object[].pop() CharSequence.putAt() Collection.putAt() // !always ordered, but we have -Collection.getAt()- -Date.putAt()- // e.g. easy access to components of Date Matcher.putAt() InputStream.readBytes() URL.readBytes() -URL.readLines()- CharSequence.reverse() -Object[].reverse()- SortedMap.reverse() SortedSet.reverse() Collection.reverseEach() // like Collection.each() this could be indeterminate... -Map.reverseEach()- // we have Map.each()... Matcher.reverseEach() Object.reverseEach() Object[].reverseEach() // perhaps foo.reverseEach() should be foo.reverse().each() Object.rightShift() // so you can do things like... foo >> log -Object[].sort()- -

InputStream.splitEachLine()- -URL.splitEachLine()- OutputStream.withWriterAppend() - BufferedWriter.write()- -OutputStream.write()- File.writeLine() OutputStream.writeLine() * and I'm not quite sure what the difference between append() and << is supposed to be, perhaps File.append() will become File << * As we have {{eachByte()}}, would it not be a good idea to also have {{eachCharacter()}} for the Readers... thanks Jeremy. P.S. This is based entirely on existing methods and existing Owner Objects, would be nice to think of what other Objects and methods we could include, (current owner objects come from java.lang.*, java.util.*, java.util.regex.*, java.io.* and java.net.*)

4. **summary:** New Groovy JDK methods - to improve consistency
   description: From my August 2004 email: I've been having a quick look at the Groovy JDK, i.e. DefaultGroovyMethods and DefaultGroovyStaticMethods As there are so many methods in these classes, it can be a little hard to take in all at once. So I've knocked up a quick script (in groovy), that collates the methods against the objects to which they become attached. hacky script -> http://javanicus.com/groovy/MungGroovySourceCode.groovy The result is a pretty table, with the methods down the left hand side, and the Classes to which they are attached along the top. Where a method has been implemented, at the crossover, I have placed a small graphic, which if you hover your cursor over, will give you a bit more detail about the method. http://javanicus.com/groovy/GroovyJDKCrossReference.html I've had a quick look for methods that I thought would be defined and from my quick inspection of abs() thru to leftShift() so far, I believe that the following candidates are available (this is by no means an exhaustive list...) * Some possible missing (non static) methods -Collection.asImmutable()- Object.asImmutable() Object.asSynchronized() - Set.count()- -Byte[].eachByte()- -File.filterLine()- -InputStream.filterLine()- -List.findIndexOf()- - Collection.flatten()- -Date.getAt()- // to get the year etc... (In conjunction with a Calendar?) - Reader.getText()- -Collection.intersect()- -BufferedWriter <<- Object[] << CharSequence << -File <<- - Map << // another Map- -Process <<- // to the process.out -Socket <<- // to the socket.out - Object[].max()- -Object[].min()- Collection.minus() Map.minus() // could compare RHS with key || value -Object[].minus()- -URL.newInputStream()- -URL.newReader()- CharSequence.padLeft() CharSequence.padRight() Object[].pop() CharSequence.putAt() Collection.putAt() // !always ordered, but we have -Collection.getAt()- -Date.putAt()- // e.g. easy access to components of Date Matcher.putAt() InputStream.readBytes() URL.readBytes() -URL.readLines()- CharSequence.reverse() - Object[].reverse()- SortedMap.reverse() SortedSet.reverse() Collection.reverseEach() // like Collection.each() this could be indeterminate... -Map.reverseEach()- // we have Map.each()... Matcher.reverseEach() Object.reverseEach() Object[].reverseEach() // perhaps foo.reverseEach() should be foo.reverse().each() Object.rightShift() // so you can do things like... foo >> log -Object[].sort()- - InputStream.splitEachLine()- -URL.splitEachLine()- OutputStream.withWriterAppend() - BufferedWriter.write()- -OutputStream.write()- File.writeLine() OutputStream.writeLine() * and I'm not quite sure what the difference between append() and << is supposed to be, perhaps File.append() will become File << * As we have {{eachByte()}}, would it not be a good idea to also have {{eachCharacter()}} for the Readers... thanks Jeremy. P.S. This is based entirely on existing methods and existing Owner Objects, would be nice to think of what other Objects and methods we could include, (current owner objects come from java.lang.*, java.util.*, java.util.regex.*, java.io.* and java.net.*)
   label: code-design
5. **summary:** New Groovy JDK methods - to improve consistency
   description: From my August 2004 email: I've been having a quick look at the Groovy JDK, i.e. DefaultGroovyMethods and DefaultGroovyStaticMethods As there are so many methods in these classes, it can be a little hard to take in all at once. So I've knocked up a quick script (in groovy), that collates the methods against the objects to which they become attached. hacky script -> http://javanicus.com/groovy/MungGroovySourceCode.groovy The result is a pretty table, with the methods down the left hand side, and the Classes to which they are attached along the top. Where a method has been implemented, at the crossover, I have placed a small graphic, which if you hover your cursor over, will give you a bit more detail about the method. http://javanicus.com/groovy/GroovyJDKCrossReference.html I've had a quick look for methods that I thought would be defined and from my quick inspection of abs() thru to leftShift() so far, I believe that the following candidates are available (this is by no means an exhaustive list...) * Some possible missing (non static) methods -Collection.asImmutable()- Object.asImmutable() Object.asSynchronized() - Set.count()- -Byte[].eachByte()- -File.filterLine()- -InputStream.filterLine()- -List.findIndexOf()- - Collection.flatten()- -Date.getAt()- // to get the year etc... (In conjunction with a Calendar?) - Reader.getText()- -Collection.intersect()- -BufferedWriter <<- Object[] << CharSequence << -File <<- - Map << // another Map- -Process <<- // to the process.out -Socket <<- // to the socket.out - Object[].max()- -Object[].min()- Collection.minus() Map.minus() // could compare RHS with key || value

-Object[].minus()- -URL.newInputStream()- -URL.newReader()- CharSequence.padLeft() CharSequence.padRight() Object[].pop() CharSequence.putAt() Collection.putAt() // !always ordered, but we have -Collection.getAt()- -Date.putAt()- // e.g. easy access to components of Date Matcher.putAt() InputStream.readBytes() URL.readBytes() -URL.readLines()- CharSequence.reverse() -Object[].reverse()- SortedMap.reverse() SortedSet.reverse() Collection.reverseEach() // like Collection.each() this could be indeterminate... -Map.reverseEach()- // we have Map.each()... Matcher.reverseEach() Object.reverseEach() Object[].reverseEach() // perhaps foo.reverseEach() should be foo.reverse().each() Object.rightShift() // so you can do things like... foo >> log -Object[].sort()- -InputStream.splitEachLine()- -URL.splitEachLine()- OutputStream.withWriterAppend() -BufferedWriter.write()- -OutputStream.write()- File.writeLine() OutputStream.writeLine() * and I'm not quite sure what the difference between append() and << is supposed to be, perhaps File.append() will become File << * As we have {{eachByte()}}, would it not be a good idea to also have {{eachCharacter()}} for the Readers... thanks Jeremy. P.S. This is based entirely on existing methods and existing Owner Objects, would be nice to think of what other Objects and methods we could include, (current owner objects come from java.lang.*, java.util.*, java.util.regex.*, java.io.* and java.net.*)

6. **summary:** New Groovy JDK methods - to improve consistency
   description: From my August 2004 email: I've been having a quick look at the Groovy JDK, i.e. DefaultGroovyMethods and DefaultGroovyStaticMethods As there are so many methods in these classes, it can be a little hard to take in all at once. So I've knocked up a quick script (in groovy), that collates the methods against the objects to which they become attached. hacky script -> http://javanicus.com/groovy/MungGroovySourceCode.groovy The result is a pretty table, with the methods down the left hand side, and the Classes to which they are attached along the top. Where a method has been implemented, at the crossover, I have placed a small graphic, which if you hover your cursor over, will give you a bit more detail about the method. http://javanicus.com/groovy/GroovyJDKCrossReference.html I've had a quick look for methods that I thought would be defined and from my quick inspection of abs() thru to leftShift() so far, I believe that the following candidates are available (this is by no means an exhaustive list...) * Some possible missing (non static) methods -Collection.asImmutable()- Object.asImmutable() Object.asSynchronized() -Set.count()- -Byte[].eachByte()- -File.filterLine()- -InputStream.filterLine()- -List.findIndexOf()- -Collection.flatten()- -Date.getAt()- // to get the year etc... (In conjunction with a Calendar?) -Reader.getText()- -Collection.intersect()- -BufferedWriter <<- Object[] << CharSequence << -File <<- -Map << // another Map- -Process <<- // to the process.out -Socket <<- // to the socket.out -Object[].max()- -Object[].min()- Collection.minus() Map.minus() // could compare RHS with key || value -Object[].minus()- -URL.newInputStream()- -URL.newReader()- CharSequence.padLeft() CharSequence.padRight() Object[].pop() CharSequence.putAt() Collection.putAt() // !always ordered, but we have -Collection.getAt()- -Date.putAt()- // e.g. easy access to components of Date Matcher.putAt() InputStream.readBytes() URL.readBytes() -URL.readLines()- CharSequence.reverse() -Object[].reverse()- SortedMap.reverse() SortedSet.reverse() Collection.reverseEach() // like Collection.each() this could be indeterminate... -Map.reverseEach()- // we have Map.each()... Matcher.reverseEach() Object.reverseEach() Object[].reverseEach() // perhaps foo.reverseEach() should be foo.reverse().each() Object.rightShift() // so you can do things like... foo >> log -Object[].sort()- -InputStream.splitEachLine()- -URL.splitEachLine()- OutputStream.withWriterAppend() -BufferedWriter.write()- -OutputStream.write()- File.writeLine() OutputStream.writeLine() * and I'm not quite sure what the difference between append() and << is supposed to be, perhaps File.append() will become File << * As we have {{eachByte()}}, would it not be a good idea to also have {{eachCharacter()}} for the Readers... thanks Jeremy. P.S. This is based entirely on existing methods and existing Owner Objects, would be nice to think of what other Objects and methods we could include, (current owner objects come from java.lang.*, java.util.*, java.util.regex.*, java.io.* and java.net.*)
7. **summary:** New Groovy JDK methods - to improve consistency
   description: From my August 2004 email: I've been having a quick look at the Groovy JDK, i.e. DefaultGroovyMethods and DefaultGroovyStaticMethods As there are so many methods in these classes, it can be a little hard to take in all at once. So I've knocked up a quick script (in groovy), that collates the methods against the objects to which they become attached. hacky script -> http://javanicus.com/groovy/MungGroovySourceCode.groovy The result is a pretty table, with the methods down the left hand side, and the Classes to which they are attached along the top. Where a method has been implemented, at the crossover, I have placed a small graphic, which if you hover your cursor over, will give you a bit more detail about the method. http://javanicus.com/groovy/GroovyJDKCrossReference.html I've had a quick look for methods that I thought would be defined and from my quick inspection of abs() thru to leftShift() so far, I believe that

the following candidates are available (this is by no means an exhaustive list...) * Some possible missing (non static) methods -Collection.asImmutable()- Object.asImmutable() Object.asSynchronized() -Set.count()- -Byte[].eachByte()- -File.filterLine()- -InputStream.filterLine()- -List.findIndexOf()- -Collection.flatten()- -Date.getAt()- // to get the year etc... (In conjunction with a Calendar?) -Reader.getText()- -Collection.intersect()- -BufferedWriter <<- Object[] << CharSequence << -File <<- -Map << // another Map- -Process <<- // to the process.out -Socket <<- // to the socket.out -Object[].max()- -Object[].min()- Collection.minus() Map.minus() // could compare RHS with key || value -Object[].minus()- -URL.newInputStream()- -URL.newReader()- CharSequence.padLeft() CharSequence.padRight() Object[].pop() CharSequence.putAt() Collection.putAt() // !always ordered, but we have -Collection.getAt()- -Date.putAt()- // e.g. easy access to components of Date Matcher.putAt() InputStream.readBytes() URL.readBytes() -URL.readLines()- CharSequence.reverse() -Object[].reverse()- SortedMap.reverse() SortedSet.reverse() Collection.reverseEach() // like Collection.each() this could be indeterminate... -Map.reverseEach()- // we have Map.each()... Matcher.reverseEach() Object.reverseEach() Object[].reverseEach() // perhaps foo.reverseEach() should be foo.reverse().each() Object.rightShift() // so you can do things like... foo >> log -Object[].sort()- -InputStream.splitEachLine()- -URL.splitEachLine()- OutputStream.withWriterAppend() -BufferedWriter.write()- -OutputStream.write()- File.writeLine() OutputStream.writeLine() * and I'm not quite sure what the difference between append() and << is supposed to be, perhaps File.append() will become File << * As we have {{eachByte()}}, would it not be a good idea to also have {{eachCharacter()}} for the Readers... thanks Jeremy. P.S. This is based entirely on existing methods and existing Owner Objects, would be nice to think of what other Objects and methods we could include, (current owner objects come from java.lang.*, java.util.*, java.util.regex.*, java.io.* and java.net.*)
label: code-design

8. **summary:** New Groovy JDK methods - to improve consistency
   description: From my August 2004 email: I've been having a quick look at the Groovy JDK, i.e. DefaultGroovyMethods and DefaultGroovyStaticMethods As there are so many methods in these classes, it can be a little hard to take in all at once. So I've knocked up a quick script (in groovy), that collates the methods against the objects to which they become attached. hacky script -> http://javanicus.com/groovy/MungGroovySourceCode.groovy The result is a pretty table, with the methods down the left hand side, and the Classes to which they are attached along the top. Where a method has been implemented, at the crossover, I have placed a small graphic, which if you hover your cursor over, will give you a bit more detail about the method. http://javanicus.com/groovy/GroovyJDKCrossReference.html I've had a quick look for methods that I thought would be defined and from my quick inspection of abs() thru to leftShift() so far, I believe that the following candidates are available (this is by no means an exhaustive list...) * Some possible missing (non static) methods -Collection.asImmutable()- Object.asImmutable() Object.asSynchronized() -Set.count()- -Byte[].eachByte()- -File.filterLine()- -InputStream.filterLine()- -List.findIndexOf()- -Collection.flatten()- -Date.getAt()- // to get the year etc... (In conjunction with a Calendar?) -Reader.getText()- -Collection.intersect()- -BufferedWriter <<- Object[] << CharSequence << -File <<- -Map << // another Map- -Process <<- // to the process.out -Socket <<- // to the socket.out -Object[].max()- -Object[].min()- Collection.minus() Map.minus() // could compare RHS with key || value -Object[].minus()- -URL.newInputStream()- -URL.newReader()- CharSequence.padLeft() CharSequence.padRight() Object[].pop() CharSequence.putAt() Collection.putAt() // !always ordered, but we have -Collection.getAt()- -Date.putAt()- // e.g. easy access to components of Date Matcher.putAt() InputStream.readBytes() URL.readBytes() -URL.readLines()- CharSequence.reverse() -Object[].reverse()- SortedMap.reverse() SortedSet.reverse() Collection.reverseEach() // like Collection.each() this could be indeterminate... -Map.reverseEach()- // we have Map.each()... Matcher.reverseEach() Object.reverseEach() Object[].reverseEach() // perhaps foo.reverseEach() should be foo.reverse().each() Object.rightShift() // so you can do things like... foo >> log -Object[].sort()- -InputStream.splitEachLine()- -URL.splitEachLine()- OutputStream.withWriterAppend() -BufferedWriter.write()- -OutputStream.write()- File.writeLine() OutputStream.writeLine() * and I'm not quite sure what the difference between append() and << is supposed to be, perhaps File.append() will become File << * As we have {{eachByte()}}, would it not be a good idea to also have {{eachCharacter()}} for the Readers... thanks Jeremy. P.S. This is based entirely on existing methods and existing Owner Objects, would be nice to think of what other Objects and methods we could include, (current owner objects come from java.lang.*, java.util.*, java.util.regex.*, java.io.* and java.net.*)
9. **summary:** New Groovy JDK methods - to improve consistency
   description: From my August 2004 email: I've been having a quick look at the Groovy JDK, i.e. DefaultGroovyMethods and DefaultGroovyStaticMethods As there are so many methods in these classes,

it can be a little hard to take in all at once. So I've knocked up a quick script (in groovy), that collates the methods against the objects to which they become attached. hacky script -> http://javanicus.com/groovy/MungGroovySourceCode.groovy The result is a pretty table, with the methods down the left hand side, and the Classes to which they are attached along the top. Where a method has been implemented, at the crossover, I have placed a small graphic, which if you hover your cursor over, will give you a bit more detail about the method. http://javanicus.com/groovy/GroovyJDKCrossReference.html I've had a quick look for methods that I thought would be defined and from my quick inspection of abs() thru to leftShift() so far, I believe that the following candidates are available (this is by no means an exhaustive list...) * Some possible missing (non static) methods -Collection.asImmutable()- Object.asImmutable() Object.asSynchronized() -Set.count()- -Byte[].eachByte()- -File.filterLine()- -InputStream.filterLine()- -List.findIndexOf()- -Collection.flatten()- -Date.getAt()- // to get the year etc... (In conjunction with a Calendar?) -Reader.getText()- -Collection.intersect()- -BufferedWriter <<- Object[] << CharSequence << -File <<- -Map << // another Map- -Process <<- // to the process.out -Socket <<- // to the socket.out -Object[].max()- -Object[].min()- Collection.minus() Map.minus() // could compare RHS with key || value -Object[].minus()- -URL.newInputStream()- -URL.newReader()- CharSequence.padLeft() CharSequence.padRight() Object[].pop() CharSequence.putAt() Collection.putAt() // !always ordered, but we have -Collection.getAt()- -Date.putAt()- // e.g. easy access to components of Date Matcher.putAt() InputStream.readBytes() URL.readBytes() -URL.readLines()- CharSequence.reverse() -Object[].reverse()- SortedMap.reverse() SortedSet.reverse() Collection.reverseEach() // like Collection.each() this could be indeterminate... -Map.reverseEach()- // we have Map.each()... Matcher.reverseEach() Object.reverseEach() Object[].reverseEach() // perhaps foo.reverseEach() should be foo.reverse().each() Object.rightShift() // so you can do things like... foo >> log -Object[].sort()- -InputStream.splitEachLine()- -URL.splitEachLine()- OutputStream.withWriterAppend() -BufferedWriter.write()- -OutputStream.write()- File.writeLine() OutputStream.writeLine() * and I'm not quite sure what the difference between append() and << is supposed to be, perhaps File.append() will become File << * As we have {{eachByte()}}, would it not be a good idea to also have {{eachCharacter()}} for the Readers... thanks Jeremy. P.S. This is based entirely on existing methods and existing Owner Objects, would be nice to think of what other Objects and methods we could include, (current owner objects come from java.lang.*, java.util.*, java.util.regex.*, java.io.* and java.net.*)

10. **summary:** New Groovy JDK methods - to improve consistency
description: From my August 2004 email: I've been having a quick look at the Groovy JDK, i.e. DefaultGroovyMethods and DefaultGroovyStaticMethods As there are so many methods in these classes, it can be a little hard to take in all at once. So I've knocked up a quick script (in groovy), that collates the methods against the objects to which they become attached. hacky script -> http://javanicus.com/groovy/MungGroovySourceCode.groovy The result is a pretty table, with the methods down the left hand side, and the Classes to which they are attached along the top. Where a method has been implemented, at the crossover, I have placed a small graphic, which if you hover your cursor over, will give you a bit more detail about the method. http://javanicus.com/groovy/GroovyJDKCrossReference.html I've had a quick look for methods that I thought would be defined and from my quick inspection of abs() thru to leftShift() so far, I believe that the following candidates are available (this is by no means an exhaustive list...) * Some possible missing (non static) methods -Collection.asImmutable()- Object.asImmutable() Object.asSynchronized() -Set.count()- -Byte[].eachByte()- -File.filterLine()- -InputStream.filterLine()- -List.findIndexOf()- -Collection.flatten()- -Date.getAt()- // to get the year etc... (In conjunction with a Calendar?) -Reader.getText()- -Collection.intersect()- -BufferedWriter <<- Object[] << CharSequence << -File <<- -Map << // another Map- -Process <<- // to the process.out -Socket <<- // to the socket.out -Object[].max()- -Object[].min()- Collection.minus() Map.minus() // could compare RHS with key || value -Object[].minus()- -URL.newInputStream()- -URL.newReader()- CharSequence.padLeft() CharSequence.padRight() Object[].pop() CharSequence.putAt() Collection.putAt() // !always ordered, but we have -Collection.getAt()- -Date.putAt()- // e.g. easy access to components of Date Matcher.putAt() InputStream.readBytes() URL.readBytes() -URL.readLines()- CharSequence.reverse() -Object[].reverse()- SortedMap.reverse() SortedSet.reverse() Collection.reverseEach() // like Collection.each() this could be indeterminate... -Map.reverseEach()- // we have Map.each()... Matcher.reverseEach() Object.reverseEach() Object[].reverseEach() // perhaps foo.reverseEach() should be foo.reverse().each() Object.rightShift() // so you can do things like... foo >> log -Object[].sort()- -InputStream.splitEachLine()- -URL.splitEachLine()- OutputStream.withWriterAppend() -BufferedWriter.write()- -OutputStream.write()- File.writeLine() OutputStream.writeLine() * and I'm not quite sure what the difference between append() and << is supposed to be, perhaps File.append() will

become File << * As we have {{eachByte()}}, would it not be a good idea to also have {{eachCharacter()}} for the Readers... thanks Jeremy. P.S. This is based entirely on existing methods and existing Owner Objects, would be nice to think of what other Objects and methods we could include, (current owner objects come from java.lang.*, java.util.*, java.util.regex.*, java.io.* and java.net.*)

11. **summary:** New Groovy JDK methods - to improve consistency
    description: From my August 2004 email: I've been having a quick look at the Groovy JDK, i.e. DefaultGroovyMethods and DefaultGroovyStaticMethods As there are so many methods in these classes, it can be a little hard to take in all at once. So I've knocked up a quick script (in groovy), that collates the methods against the objects to which they become attached. hacky script -> http://javanicus.com/groovy/MungGroovySourceCode.groovy The result is a pretty table, with the methods down the left hand side, and the Classes to which they are attached along the top. Where a method has been implemented, at the crossover, I have placed a small graphic, which if you hover your cursor over, will give you a bit more detail about the method. http://javanicus.com/groovy/GroovyJDKCrossReference.html I've had a quick look for methods that I thought would be defined and from my quick inspection of abs() thru to leftShift() so far, I believe that the following candidates are available (this is by no means an exhaustive list...) * Some possible missing (non static) methods -Collection.asImmutable()- Object.asImmutable() Object.asSynchronized() -Set.count()- -Byte[].eachByte()- -File.filterLine()- -InputStream.filterLine()- -List.findIndexOf()- -Collection.flatten()- -Date.getAt()- // to get the year etc... (In conjunction with a Calendar?) -Reader.getText()- -Collection.intersect()- -BufferedWriter <<- Object[] << CharSequence << -File <<- -Map << // another Map- -Process <<- // to the process.out -Socket <<- // to the socket.out -Object[].max()- -Object[].min()- Collection.minus() Map.minus() // could compare RHS with key || value -Object[].minus()- -URL.newInputStream()- -URL.newReader()- CharSequence.padLeft() CharSequence.padRight() Object[].pop() CharSequence.putAt() Collection.putAt() // !always ordered, but we have -Collection.getAt()- -Date.putAt()- // e.g. easy access to components of Date Matcher.putAt() InputStream.readBytes() URL.readBytes() -URL.readLines()- CharSequence.reverse() -Object[].reverse()- SortedMap.reverse() SortedSet.reverse() Collection.reverseEach() // like Collection.each() this could be indeterminate... -Map.reverseEach()- // we have Map.each()... Matcher.reverseEach() Object.reverseEach() Object[].reverseEach() // perhaps foo.reverseEach() should be foo.reverse().each() Object.rightShift() // so you can do things like... foo >> log -Object[].sort()- -InputStream.splitEachLine()- -URL.splitEachLine()- OutputStream.withWriterAppend() -BufferedWriter.write()- -OutputStream.write()- File.writeLine() OutputStream.writeLine() * and I'm not quite sure what the difference between append() and << is supposed to be, perhaps File.append() will become File << * As we have {{eachByte()}}, would it not be a good idea to also have {{eachCharacter()}} for the Readers... thanks Jeremy. P.S. This is based entirely on existing methods and existing Owner Objects, would be nice to think of what other Objects and methods we could include, (current owner objects come from java.lang.*, java.util.*, java.util.regex.*, java.io.* and java.net.*)
12. **summary:** New Groovy JDK methods - to improve consistency
    description: From my August 2004 email: I've been having a quick look at the Groovy JDK, i.e. DefaultGroovyMethods and DefaultGroovyStaticMethods As there are so many methods in these classes, it can be a little hard to take in all at once. So I've knocked up a quick script (in groovy), that collates the methods against the objects to which they become attached. hacky script -> http://javanicus.com/groovy/MungGroovySourceCode.groovy The result is a pretty table, with the methods down the left hand side, and the Classes to which they are attached along the top. Where a method has been implemented, at the crossover, I have placed a small graphic, which if you hover your cursor over, will give you a bit more detail about the method. http://javanicus.com/groovy/GroovyJDKCrossReference.html I've had a quick look for methods that I thought would be defined and from my quick inspection of abs() thru to leftShift() so far, I believe that the following candidates are available (this is by no means an exhaustive list...) * Some possible missing (non static) methods -Collection.asImmutable()- Object.asImmutable() Object.asSynchronized() -Set.count()- -Byte[].eachByte()- -File.filterLine()- -InputStream.filterLine()- -List.findIndexOf()- -Collection.flatten()- -Date.getAt()- // to get the year etc... (In conjunction with a Calendar?) -Reader.getText()- -Collection.intersect()- -BufferedWriter <<- Object[] << CharSequence << -File <<- -Map << // another Map- -Process <<- // to the process.out -Socket <<- // to the socket.out -Object[].max()- -Object[].min()- Collection.minus() Map.minus() // could compare RHS with key || value -Object[].minus()- -URL.newInputStream()- -URL.newReader()- CharSequence.padLeft() CharSequence.padRight() Object[].pop() CharSequence.putAt() Collection.putAt() // !always ordered, but we have -Collection.getAt()- -Date.putAt()- // e.g. easy access to components of Date Matcher.putAt() InputStream.readBytes() URL.readBytes() -URL.readLines()- CharSequence.reverse() -

Object[].reverse()- SortedMap.reverse() SortedSet.reverse() Collection.reverseEach() // like Collection.each() this could be indeterminate... -Map.reverseEach()- // we have Map.each()... Matcher.reverseEach() Object.reverseEach() Object[].reverseEach() // perhaps foo.reverseEach() should be foo.reverse().each() Object.rightShift() // so you can do things like... foo >> log -Object[].sort()- -InputStream.splitEachLine()- -URL.splitEachLine()- OutputStream.withWriterAppend() -BufferedWriter.write()- -OutputStream.write()- File.writeLine() OutputStream.writeLine() * and I'm not quite sure what the difference between append() and << is supposed to be, perhaps File.append() will become File << * As we have {{eachByte()}}, would it not be a good idea to also have {{eachCharacter()}} for the Readers... thanks Jeremy. P.S. This is based entirely on existing methods and existing Owner Objects, would be nice to think of what other Objects and methods we could include, (current owner objects come from java.lang.*, java.util.*, java.util.regex.*, java.io.* and java.net.*)

13. **summary:** New Groovy JDK methods - to improve consistency
    description: From my August 2004 email: I've been having a quick look at the Groovy JDK, i.e. DefaultGroovyMethods and DefaultGroovyStaticMethods As there are so many methods in these classes, it can be a little hard to take in all at once. So I've knocked up a quick script (in groovy), that collates the methods against the objects to which they become attached. hacky script -> http://javanicus.com/groovy/MungGroovySourceCode.groovy The result is a pretty table, with the methods down the left hand side, and the Classes to which they are attached along the top. Where a method has been implemented, at the crossover, I have placed a small graphic, which if you hover your cursor over, will give you a bit more detail about the method. http://javanicus.com/groovy/GroovyJDKCrossReference.html I've had a quick look for methods that I thought would be defined and from my quick inspection of abs() thru to leftShift() so far, I believe that the following candidates are available (this is by no means an exhaustive list...) * Some possible missing (non static) methods -Collection.asImmutable()- Object.asImmutable() Object.asSynchronized() -Set.count()- -Byte[].eachByte()- -File.filterLine()- -InputStream.filterLine()- -List.findIndexOf()- -Collection.flatten()- -Date.getAt()- // to get the year etc... (In conjunction with a Calendar?) -Reader.getText()- -Collection.intersect()- -BufferedWriter <<- Object[] << CharSequence << -File <<- -Map << // another Map- -Process <<- // to the process.out -Socket <<- // to the socket.out -Object[].max()- -Object[].min()- Collection.minus() Map.minus() // could compare RHS with key || value -Object[].minus()- -URL.newInputStream()- -URL.newReader()- CharSequence.padLeft() CharSequence.padRight() Object[].pop() CharSequence.putAt() Collection.putAt() // !always ordered, but we have -Collection.getAt()- -Date.putAt()- // e.g. easy access to components of Date Matcher.putAt() InputStream.readBytes() URL.readBytes() -URL.readLines()- CharSequence.reverse() -Object[].reverse()- SortedMap.reverse() SortedSet.reverse() Collection.reverseEach() // like Collection.each() this could be indeterminate... -Map.reverseEach()- // we have Map.each()... Matcher.reverseEach() Object.reverseEach() Object[].reverseEach() // perhaps foo.reverseEach() should be foo.reverse().each() Object.rightShift() // so you can do things like... foo >> log -Object[].sort()- -InputStream.splitEachLine()- -URL.splitEachLine()- OutputStream.withWriterAppend() -BufferedWriter.write()- -OutputStream.write()- File.writeLine() OutputStream.writeLine() * and I'm not quite sure what the difference between append() and << is supposed to be, perhaps File.append() will become File << * As we have {{eachByte()}}, would it not be a good idea to also have {{eachCharacter()}} for the Readers... thanks Jeremy. P.S. This is based entirely on existing methods and existing Owner Objects, would be nice to think of what other Objects and methods we could include, (current owner objects come from java.lang.*, java.util.*, java.util.regex.*, java.io.* and java.net.*)
    label: code-design
14. **summary:** New Groovy JDK methods - to improve consistency
    description: From my August 2004 email: I've been having a quick look at the Groovy JDK, i.e. DefaultGroovyMethods and DefaultGroovyStaticMethods As there are so many methods in these classes, it can be a little hard to take in all at once. So I've knocked up a quick script (in groovy), that collates the methods against the objects to which they become attached. hacky script -> http://javanicus.com/groovy/MungGroovySourceCode.groovy The result is a pretty table, with the methods down the left hand side, and the Classes to which they are attached along the top. Where a method has been implemented, at the crossover, I have placed a small graphic, which if you hover your cursor over, will give you a bit more detail about the method. http://javanicus.com/groovy/GroovyJDKCrossReference.html I've had a quick look for methods that I thought would be defined and from my quick inspection of abs() thru to leftShift() so far, I believe that the following candidates are available (this is by no means an exhaustive list...) * Some possible missing (non static) methods -Collection.asImmutable()- Object.asImmutable() Object.asSynchronized() -Set.count()- -Byte[].eachByte()- -File.filterLine()- -InputStream.filterLine()- -List.findIndexOf()- -

Collection.flatten()- -Date.getAt()- // to get the year etc... (In conjunction with a Calendar?) - Reader.getText()- -Collection.intersect()- -BufferedWriter <<- Object[] << CharSequence << -File <<- - Map << // another Map- -Process <<- // to the process.out -Socket <<- // to the socket.out - Object[].max()- -Object[].min()- Collection.minus() Map.minus() // could compare RHS with key || value -Object[].minus()- -URL.newInputStream()- -URL.newReader()- CharSequence.padLeft() CharSequence.padRight() Object[].pop() CharSequence.putAt() Collection.putAt() // !always ordered, but we have -Collection.getAt()- -Date.putAt()- // e.g. easy access to components of Date Matcher.putAt() InputStream.readBytes() URL.readBytes() -URL.readLines()- CharSequence.reverse() - Object[].reverse()- SortedMap.reverse() SortedSet.reverse() Collection.reverseEach() // like Collection.each() this could be indeterminate... -Map.reverseEach()- // we have Map.each()... Matcher.reverseEach() Object.reverseEach() Object[].reverseEach() // perhaps foo.reverseEach() should be foo.reverse().each() Object.rightShift() // so you can do things like... foo >> log -Object[].sort()- - InputStream.splitEachLine()- -URL.splitEachLine()- OutputStream.withWriterAppend() - BufferedWriter.write()- -OutputStream.write()- File.writeLine() OutputStream.writeLine() * and I'm not quite sure what the difference between append() and << is supposed to be, perhaps File.append() will become File << * As we have {{eachByte()}}, would it not be a good idea to also have {{eachCharacter()}} for the Readers... thanks Jeremy. P.S. This is based entirely on existing methods and existing Owner Objects, would be nice to think of what other Objects and methods we could include, (current owner objects come from java.lang.*, java.util.*, java.util.regex.*, java.io.* and java.net.*)
label: code-design

jira_issues_comments:

1. flatten() and minus() have been added for Set
2. crossed off ones that are added up to 1.5.4
3. Probably this has been discussed in the past, but I've been starting to wonder if it wouldn't make sense to split the DefaultGroovy(Static)Methods into multiple smaller units. To me the DefaultGroovy(Static)Methods look like default wired categories, and I think it may help to have smaller, fine grained units for this. What do you think? ./alex -- .w(the_mindstorm)p.
4. **body:** Perhaps this is what Paul means by variations, but specifically splitEachLine() should support the split(String regex, int limit) version of split, like splitEachLine(regex,limit){} I was recently much embarrassed when I told a colleague that his tab-delimited file had an inconsistent number of columns based on the output of splitEachLine(), which by default (unknown to me) strips off trailing empty fields. Once I realized the source of the problem, there was no way to fix it with splitEachLine. At least with split there is the ugly split(regex,-1) to instruct split not to strip trailing fields. I find the default strip ending fields behavior of split very counterintuitive in both Java and Groovy, so I actually long for a better solution, but at least adding the limit field will allow the ugly fix.
   label: code-design
5. cross off Date#putAt
6. Most of the major DGM methods avec already been added. We can add some others on a case by case basis, when users ask for specific ones.
7. **body:** Script needs updating to 1.0, I think there was one def I needed to add and a handful of '|' to '->' closure symbols to change.
   label: code-design
8. added URL.newReader()
9. include minus() for Object[]
10. Round out eachLine() and splitEachLine() variations plus add lines() for String and File.
11. Crossed a few more that have been covered in other issues.
12. Potential patch for Date.putAt() case attached. Does Calendar and Date and an additional set method too.